

Notes

This course presents SPAD (Structured Prompt-Driven Engineering), Fernando García Varela's methodology for building software with the help of artificial intelligence in a controlled way. It is not a collection of instructions for requesting code: it is a way of working.

The course is designed for a full classroom session and also as outreach material. It has ten modules (0 to 9) and a closing session with cases for discussion. Each module opens with its objective and closes with an exercise or a self-check. The notes on each slide are the teacher's script.

No programming knowledge is needed. For an executive, the key question is simple: if your company already builds software with AI, who takes the design decisions, who reviews and where is the evidence kept?

- SPAD does not promise improvement figures: each organisation measures with its own data.
- SPAD does not certify: conformity is checked with an internal self-assessment.
- The examples in the course are fictitious or illustrative.

SPAD is published under a CC BY 4.0 licence for the content and MIT for the code. The authorship must be cited: SPAD · Fernando García Varela.

Question to open the session: how many of you already use AI to write code, reports or formulas in your work?

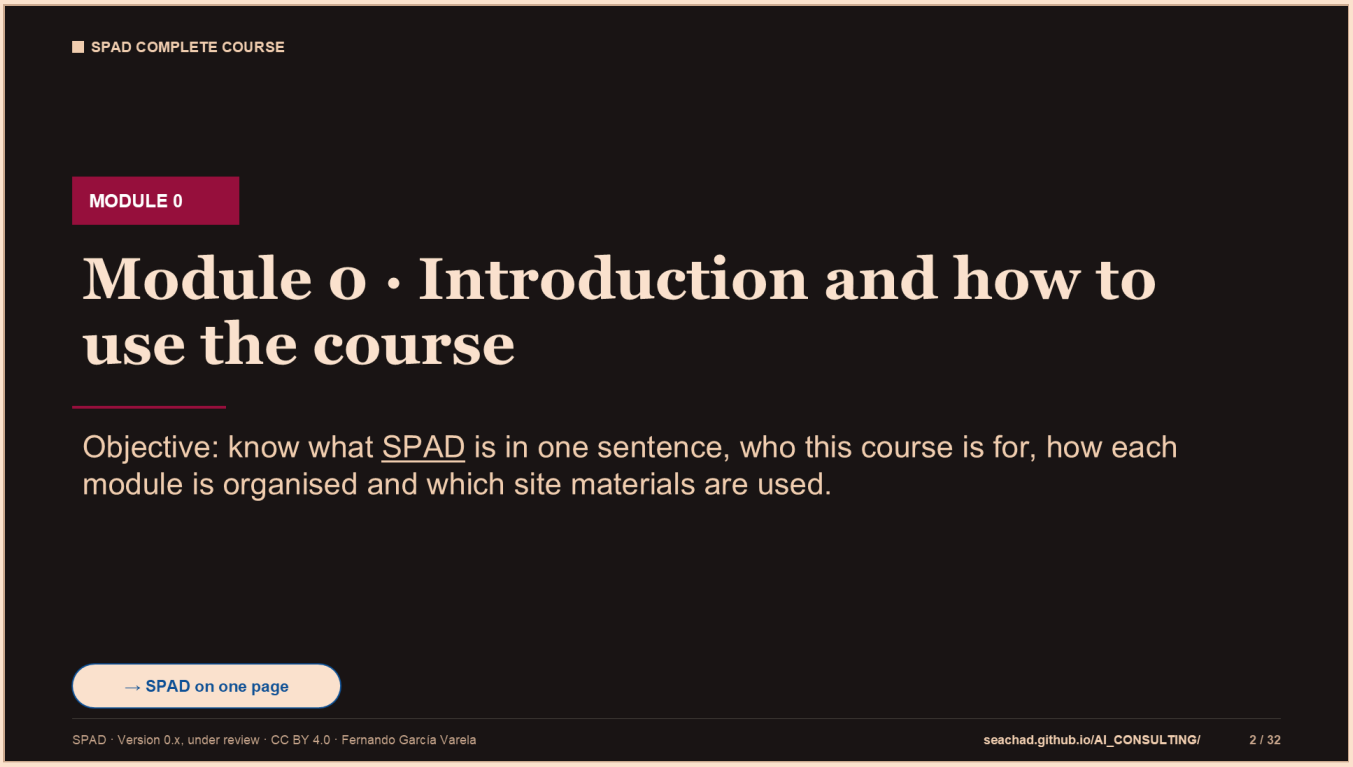
TERMS ON THIS SLIDE

- **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**: 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 · 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
- **SPAD library** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/index.html



Notes

Objective of module 0: the student knows what they are going to learn and how to use the material.

SPAD answers a very specific question: how to build software with AI without losing control. AI writes plausible code at great speed. Without structure, that speed produces design decisions that nobody has taken consciously, insufficient tests and vulnerabilities that slip in silently.

SPAD has three properties that we will see in module 2: it is sequential (the phases follow a fixed order), blocking (no phase starts without reviewing the previous one) and auditable (each phase leaves a stored document).

Why it matters to an executive: they do not need to know how an architecture plan is reviewed, but they do need to know that there is a method that does it and that leaves evidence. That is what the board, the customer or the auditor will ask for.

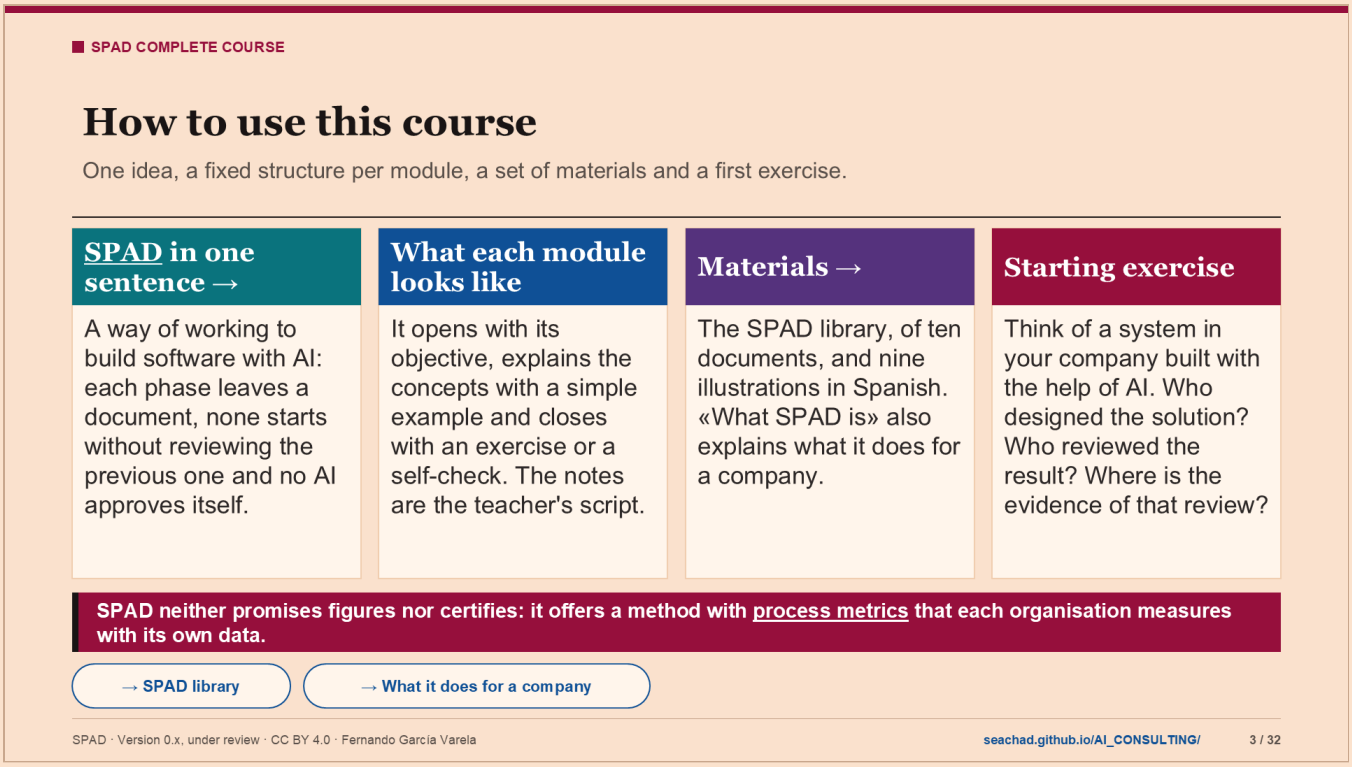
Suggestion for the teacher: spend only a few minutes on this module and move quickly to the problem (module 1), which is what most interests a non-technical audience.

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 · What SPAD is** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is
- **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 on one page** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is



Notes

This slide explains how the course is built and proposes the first exercise.

SPAD in one sentence: each phase leaves a document (in SPAD it is called an artefact), no phase starts without reviewing the previous one and no AI approves itself. A person always validates.

Each module follows the same structure: objective, concepts with a simple example and an exercise or self-check. The answers to the exercises are in the notes.

Materials: the SPAD library has ten documents. Document 00 presents the method; 01 is the operating guide with the phases, roles and verdicts; 02 explains contexts, topics and records; 03 is the validation policy; 04, the complementary cycles; 05, systems that include AI; 06 and 07, the instructions and contracts for each phase; 08, self-assessment and metrics. Document 00 also explains what SPAD does for a company: building with method and making demands of its suppliers (module 9). The illustrations are in Spanish.

Starting exercise (five minutes, in pairs): think of a system in your company built with the help of AI, even if it is a spreadsheet with generated formulas or a small program. Answer: who designed the solution?, who reviewed the result?, where is the evidence of that review? Keep the answers: we will come back to them at the close.

Why it matters: many participants will discover that they cannot answer the third question. That is the gap SPAD covers.

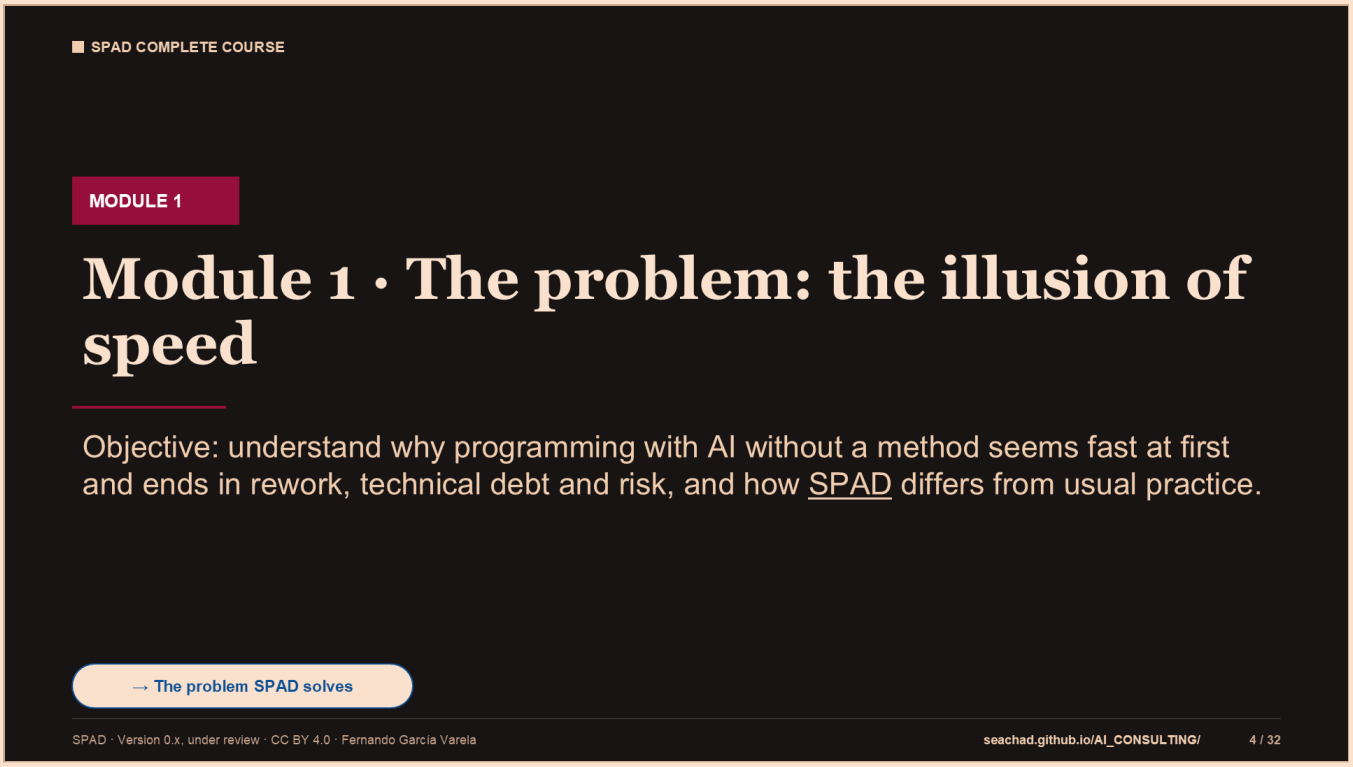
TERMS ON THIS SLIDE

- **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
- **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 00 · Status, licence and next steps (library map)** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#status-licence-and-next-steps
- **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 library** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/index.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



Notes

Objective of module 1: understand the problem before the solution.

AI has changed the economics of writing code: what used to take days can now take minutes. That is real and valuable. The problem is a different one: the speed of writing is not the speed of delivering a system that works in production, that can be maintained and that someone can explain.

SPAD document 00 summarises the typical symptoms:

- The code works in the demonstration and fails in production, because nobody designed the solution.
- Nobody knows why the system is built that way, because the decisions were taken by the AI inside the code.
- Tests are scarce, because they are written at the end, if there is time left.
- Vulnerabilities appear, because the same AI that wrote the code reviewed it, or nobody did.
- An audit asks for evidence and there is none, because everything stayed in conversations with the AI.

Key idea for the classroom: none of these problems belongs to the AI. They are method problems that AI amplifies.

Question for the classroom: has anyone lived through a project that went very fast at the start and got stuck at the end? What was missing?

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

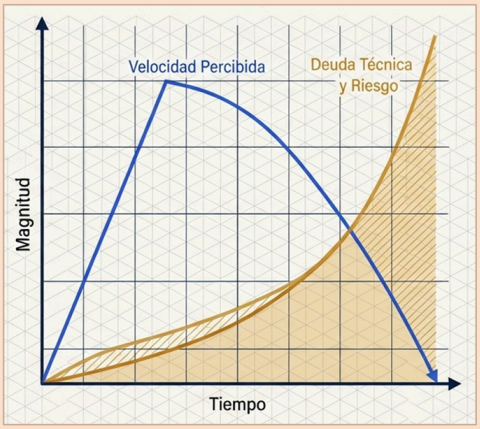
SOURCES AND LINKS

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

SPAD COMPLETE COURSE

The illusion of speed

Conceptual chart: perceived speed rises and then falls; technical debt and risk grow.



- **What you see** At first, AI generates plausible code very quickly. The feeling of progress is huge.
- **What you do not see** Design decisions nobody took, missing tests, vulnerabilities and technical debt that grow silently.
- **Why it matters** It is not an AI failure: it is a method problem that AI amplifies. It is paid later, in incidents and rework.
- **Illustration** SPAD illustration, in Spanish. It is conceptual: it contains no measured data.

→ The problem SPAD solves

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 5 / 32

Notes

The illustration is a conceptual chart, with no measured data. It shows two curves over time.

The blue line is perceived speed. It rises very quickly at first: AI generates screens, functions and apparent tests in minutes. Then it falls, because each new change collides with decisions that nobody documented and work has to be redone.

The orange curve is technical debt and risk. It grows slowly at first and then shoots up. Technical debt is the future cost of today's shortcuts: code that is hard to change, untested or unexplained.

Fictitious example: a team builds an ordering portal with AI in two weeks. In the demonstration everything works. By the third month, each change breaks something else, nobody knows why a table has the name it has and a security review finds a password written in the code.

Why it matters for decisions: the cost does not appear in the project budget, but later, in incidents, rework and loss of trust. SPAD keeps the speed where it adds value and puts control on decisions, reviews and evidence.

An honest warning: SPAD does not claim any improvement figure. At first it makes work slower while the team learns the phases.

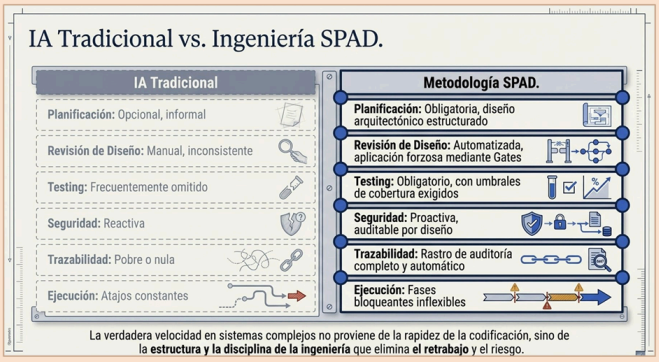
SOURCES AND LINKS

- **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 00 · When to use SPAD (no improvement figures)** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#when-to-use-spad
- **The problem SPAD solves** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves

SPAD COMPLETE COURSE

Self-check: AI without a method versus SPAD

Six aspects: planning, design review, testing, security, traceability and execution.



- **Question 1** In your team, is design mandatory or is code requested until it 'works'?
- **Question 2** Does the same AI that wrote the code review it, another AI or a person?
- **Question 3** If an audit arrives tomorrow, can you show the plan, the tests and who validated each step?
- **Illustration** Illustration in Spanish. Where it says 'automated' or 'Gates', in SPAD a person always accepts the verdict.

If you answer 'I do not know' to any question, quality depends on specific people, not on a process.

→ Symptoms and SPAD's response

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 6 / 32

Notes

The illustration compares six aspects of using AI without a method with SPAD.

- Planning: without a method it is optional and informal; in SPAD it is mandatory and explicit, before writing code.
- Design review: without a method it is irregular; in SPAD it is done by an AI other than the one that designed, and a person validates it.
- Testing: without a method it is often skipped; in SPAD the minimum coverage is set before building.
- Security: without a method it is reactive; in SPAD sensitive code undergoes a review that can block deployment.
- Traceability: without a method it is poor; in SPAD each phase leaves an artefact with a name, topic and version.
- Execution: without a method there are constant shortcuts; in SPAD the phases are blocking.

Clarification: the illustration speaks of 'automated' review and of 'Gates'. In SPAD the technical review is done by an AI Reviewer, but a person always accepts or rejects the verdict. The 'Gates' in the illustration are those blocking points between phases, not the company's investment decisions.

Guide answers: if design is not mandatory, if the same AI reviews or if you cannot show the evidence, your team is in the left-hand column for that aspect. It is not a judgement: it is the starting point.

Why it matters: with SPAD, 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.

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 · 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 · Verdicts** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#verdicts
- **Symptoms and SPAD's response** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#the-problem-it-solves



Notes

Objective of module 2: know the ideas that support the whole method.

SPAD turns AI from an informal programming assistant into a governed engineering tool, whose work can be reviewed, reproduced and audited.

The three properties:

- Sequential: work advances through phases in a fixed order.
- Blocking: no phase starts until the output of the previous one has been produced and evaluated.
- Auditable: each phase produces an explicit artefact that is stored.

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

Example for the classroom: a student hands in a perfect exam, but copied. The result looks good, but it does not count, because the process was not followed. In SPAD the same happens with an AI response that skips its phase: it is discarded in full and the phase is repeated.

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

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
- **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

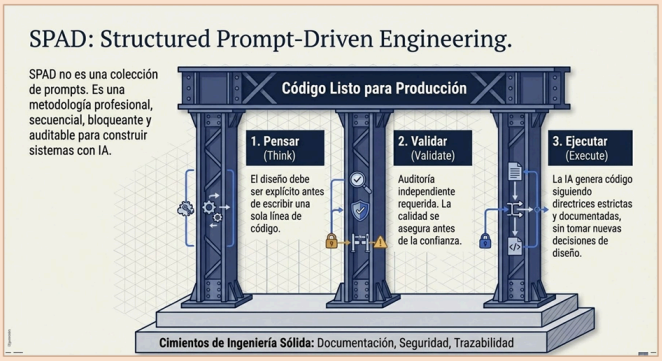
SOURCES AND LINKS

- **SPAD 00 · What SPAD is** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is
- **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 COMPLETE COURSE

Think, validate, execute

Three pillars on foundations of documentation, security and traceability.



- **Sequential** First the problem is understood; then it is designed, reviewed, tests are defined, it is built, reviewed and versioned.
- **Blocking** Under the blocking execution principle, no phase starts until a person validates the previous one.
- **Auditable** Each phase leaves an explicit document that is the mandatory input to the next one and is stored.
- **Illustration** SPAD illustration, in Spanish: think, validate and execute.

In SPAD, complying with the flow is more important than the apparent quality of the result.

→ What SPAD is | → Blocking execution principle

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 8 / 32

Notes

The illustration shows three pillars that support production-ready code, on foundations of documentation, security and traceability.

Think: the design must be explicit before a single line of code is written. The AI Planner does it in the PLAN phase, and another AI reviews it.

Validate: an independent review assures quality before trusting the result. The AI that reviews is never the one that builds, and a person accepts or rejects its verdict.

Execute: the AI generates code following strict, documented rules, without taking new design decisions.

The three properties of SPAD:

- Sequential: first understand; then design and review, define the tests, build, review and version.
- Blocking: coding does not start ‘while the plan is being finished’. At that moment the plan would cease to govern anything.
- Auditable: plan, review reports, test strategy and release record are stored.

Example: on a building site, no wall goes up before the plans are approved. SPAD applies the same logic to software built with AI.

Question for the classroom: in which other processes in your company is it normal that whoever does something does not approve it themselves? For example, in accounting, purchasing or risk.

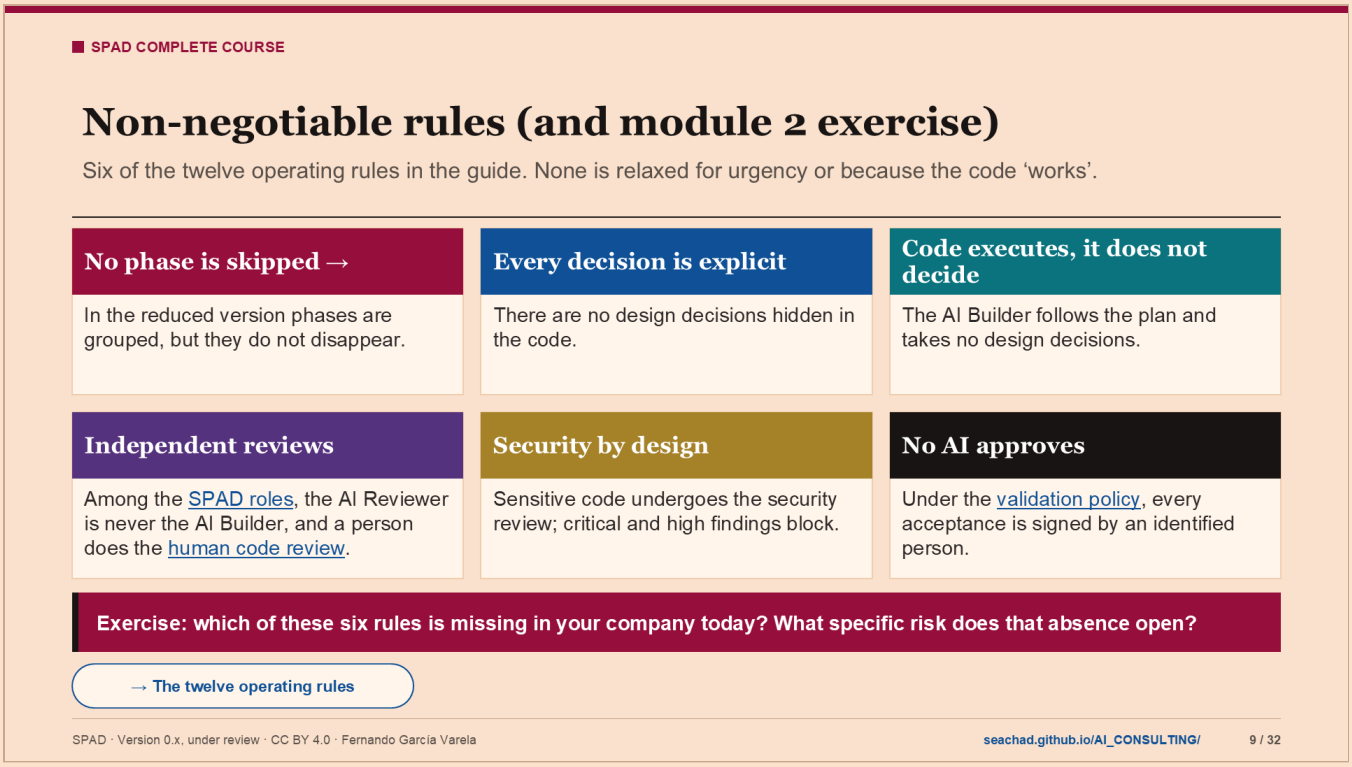
TERMS ON THIS SLIDE

- **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**: 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 · What SPAD is** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is
- **What SPAD is** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#what-spad-is



Notes

The SPAD operating guide (document 01, section 7) sets twelve operating rules that are not negotiable. The slide shows six; the other six are: every output is auditable, minimal fixes (surgical, never covert refactoring), mandatory tests with minimum coverage set before building, contexts loaded before any phase, one topic per piece of work and the model recorded in every artefact.

Each rule protects something specific:

- No phase is skipped: protects blocking.
- Every decision is explicit: protects explainability. If in a year someone asks why the system does something, the answer is in the plan, not in one person's memory.
- Code executes, it does not decide: protects the plan.
- Independent reviews: protect quality. Nobody marks their own exam well.
- Security by design: protects data and people.
- No AI approves: protects accountability. Behind each acceptance there is a name.

Module 2 exercise (ten minutes, in groups): choose a rule that is not met in your company today and describe a plausible failure that its absence would allow. Example answer: 'No AI approves is not met: the team accepts whatever the assistant generates. Risk: a change in the commission calculation goes into production without anyone having read it'.

Why it matters for an executive: these rules are the questions you can ask in a committee without knowing how to program.

TERMS ON THIS SLIDE

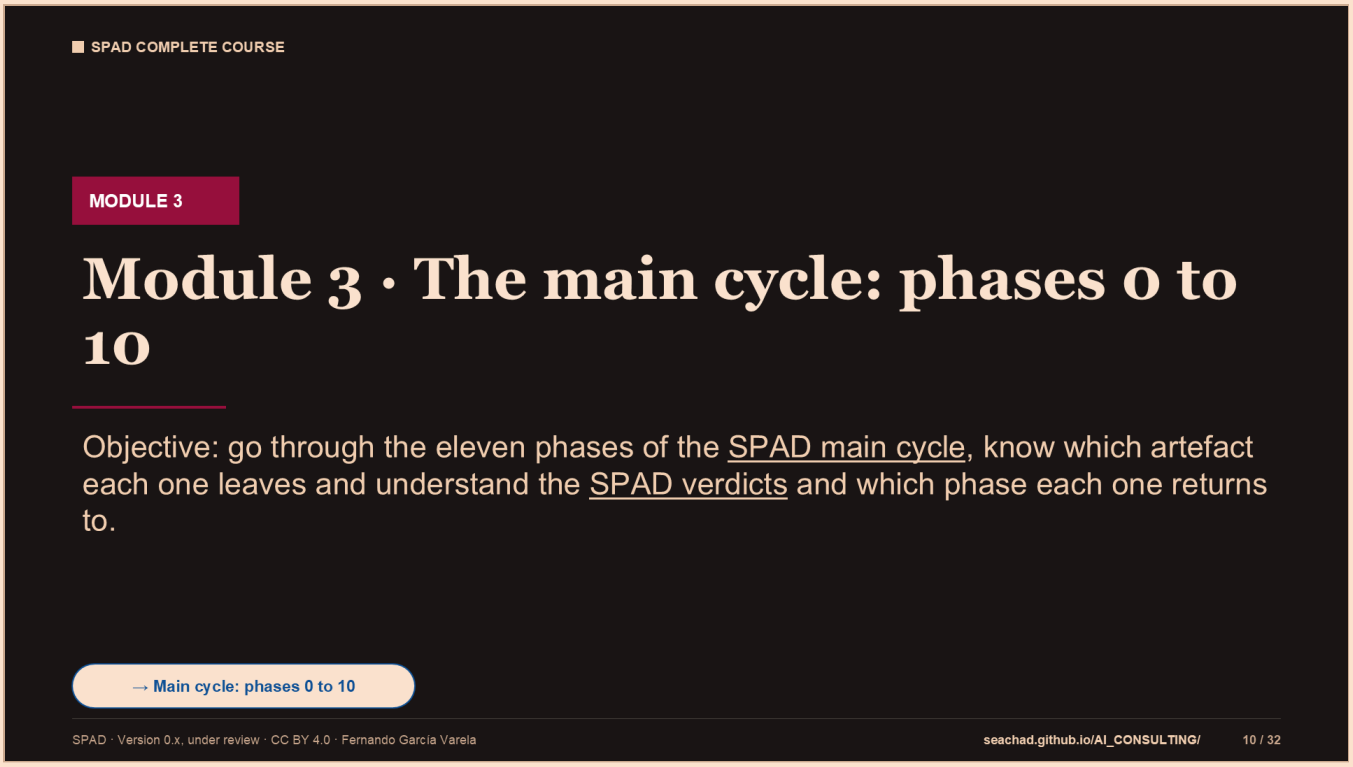
- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what

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

- **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 01 · Non-negotiable operating rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules
- **The twelve operating rules** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#non-negotiable-operating-rules



Notes

Objective of module 3: know the full path of a piece of work in SPAD.

The main cycle has eleven phases, from 0 to 10. They can be grouped into two blocks:

- Think before building (phases 0 to 4): context and objective, plan, plan review, code primer and test strategy.
- Build and review (phases 5 to 10): implementation, tests, test review, code review, fixes and version management.

An idea that surprises a business audience: almost half of the phases happen before any code is written. It is deliberate. A design error found in the plan review costs a conversation; the same error found in production costs an incident.

Example that will accompany the whole module (fictitious): a team must add payment with a new gateway to a shop's website. The work topic will be called payment_gateway.

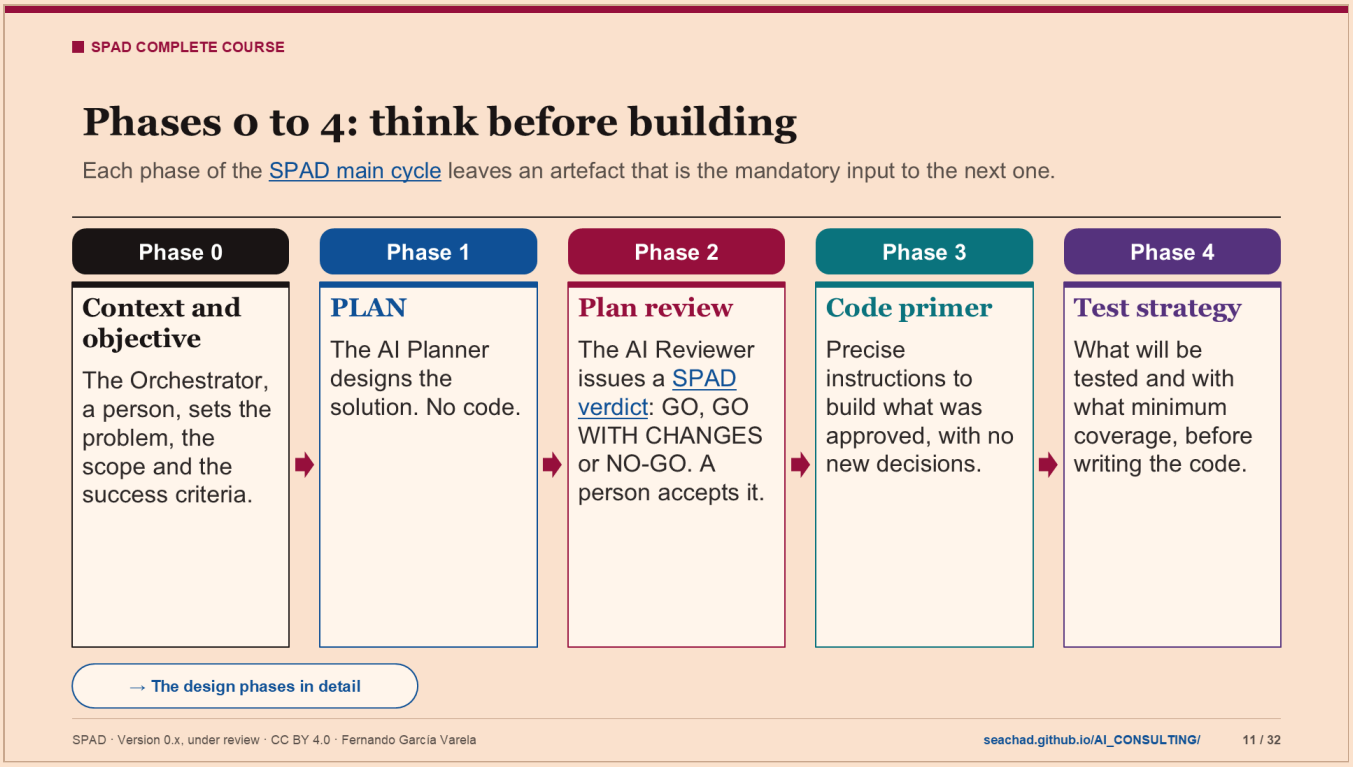
Question for the classroom: in your projects, what percentage of time goes on thinking before building? And on redoing afterwards?

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

SOURCES AND LINKS

- **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



Notes

Phases 0 to 4 are the thinking block.

- Phase 0 · Context and objective: a person, the Orchestrator, does it. It defines which problem is solved, what is out of scope and how it will be known that the work has succeeded. In the payment_gateway example: 'the customer can pay with the new gateway; failed payments do not create orders; nothing changes in the existing payment methods'.
- Phase 1 · PLAN: the AI Planner proposes the architecture, the components, the risks and the design decisions. It writes no code.
- Phase 2 · Plan review: a different AI reviews the plan and issues a justified verdict. A person accepts or rejects it. If the plan does not handle the case of a gateway that does not respond, this is where it is detected.
- Phase 3 · Code primer: turns the approved plan into instructions for building. It adds no decisions.
- Phase 4 · Test strategy: sets what will be tested and the minimum coverage before building. That way, tests do not depend on time being left over at the end.

Why it matters: each phase produces an artefact with a name, number and topic. If phase 2 rejects the plan, the rejected plan is not deleted: it is kept next to the new one. That makes it possible to explain later why the design changed.

Question for the classroom: who should be the Orchestrator in a project in your company: the product manager, the technical lead or the business owner?

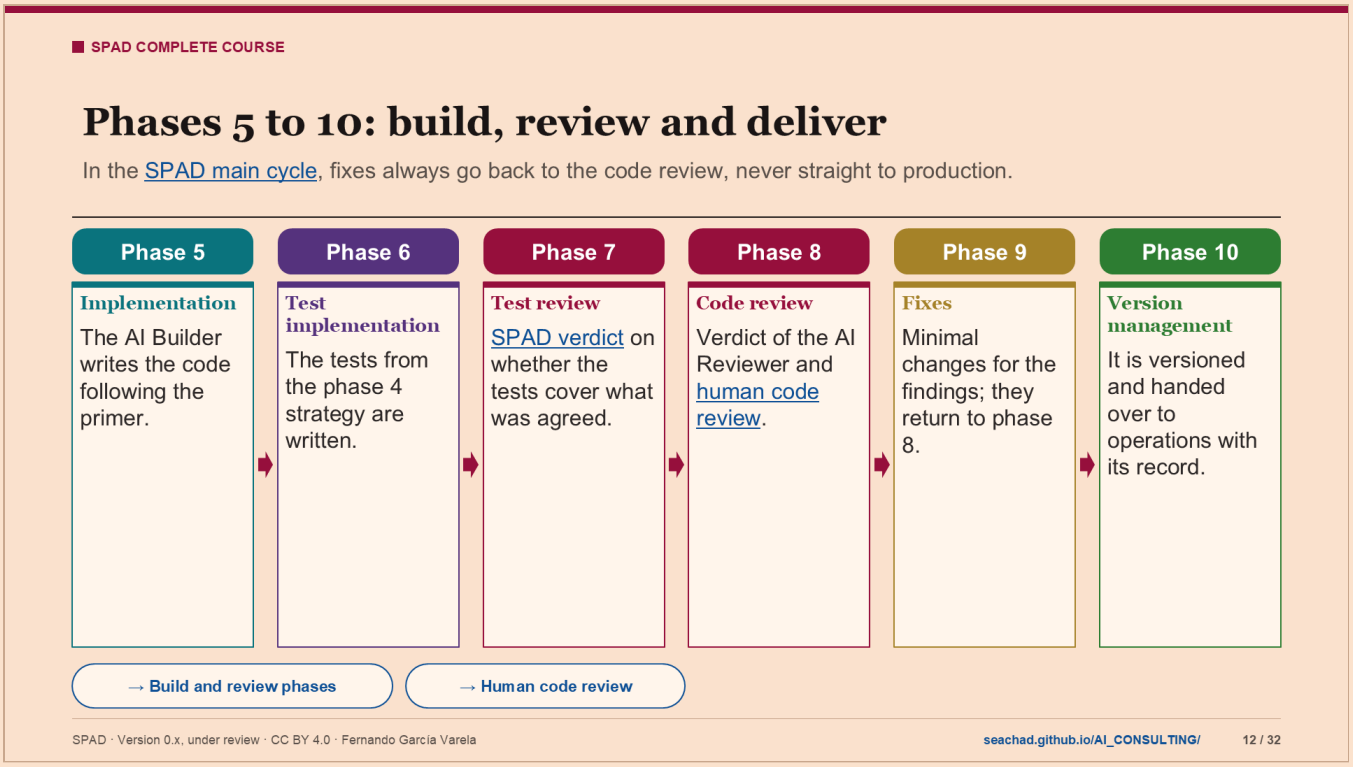
TERMS ON THIS SLIDE

- **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

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 06 · Instructions per phase** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/06_SPAD_Instrucciones_por_fase.html#how-to-use-this-document
- **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



Notes

Phases 5 to 10 are the build and review block.

- Phase 5 · Implementation: the AI Builder writes the code following the phase 3 primer. If it finds something the plan does not resolve, it does not improvise: it flags it.
- Phase 6 · Test implementation: the tests defined in phase 4 are written.
- Phase 7 · Test review: an AI Reviewer checks that the tests cover what was agreed and issues its verdict.
- Phase 8 · Code review: the AI Reviewer issues its technical verdict. In addition, a person always reviews and it is recorded. For sensitive code (authentication, authorisation, payments, personal data, agent limits, decisions about people or critical production) a second human review is needed.
- Phase 9 · Fixes: minimal, justified changes for each finding. Then work goes back to phase 8. A fix never goes straight to production.
- Phase 10 · Version management: the change is versioned and handed over to operations with its record.

In the payment_gateway example, the payment involves money: the code also goes through the security cycle (module 6) and the second human review.

Why it matters: the loop between phases 8 and 9 avoids the usual trap of the ‘quick unreviewed fix’, which is where most errors get into production.

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

- **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 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
- **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

SPAD COMPLETE COURSE

Verdicts and which phase work returns to

The SPAD main cycle uses one vocabulary of SPAD verdicts in every review. The person accepts or rejects the verdict.

Review	GO	GO WITH CHANGES	NO-GO
Phase 2 · Plan review	Goes to phase 3	Returns to phase 1 to adjust the plan	Returns to phase 0: rethink the objective
Phase 7 · Test review	Goes to phase 8	Goes to phase 9 to fix	Returns to phase 6: redo the tests
Phase 8 · Code review	Goes to phase 10	Goes to phase 9; then back to phase 8	Returns to phase 1: the design fails

Self-check: the code review gives NO-GO because the design does not work. Which phase does work return to? (Answer: phase 1.)

→ Verdicts

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 13 / 32

Notes

All SPAD reviews use the same vocabulary of three verdicts:

- GO: work can move on.
- GO WITH CHANGES: work can move on after fixing what is indicated.
- NO-GO: work cannot move on; it has to go back.

The table says which phase work returns to in each case. Note the logic: the more serious the problem, the further back work goes. A NO-GO in the code review is not fixed with patches: if the design fails, work goes back to the plan.

Three rules for verdicts:

- A verdict without justification is incomplete.
- A GO with open critical or high findings is self-approval, and the response is invalidated (module 4).
- A well-founded NO-GO is not a failure: it is the process working.

The AI Reviewer issues the verdict, but it does not decide: a person accepts or rejects it and it is recorded.

Module 3 self-check (answers):

- The plan review gives GO WITH CHANGES. Where to? Phase 1.
- The test review gives NO-GO. Where to? Phase 6.
- After the phase 9 fixes, where does work go? To phase 8, never to phase 10.

Why it matters: an executive committee also knows how to stop a project. SPAD does the same in every technical review, with written rules and not at the discretion of the moment.

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

SOURCES AND LINKS

- **SPAD 03 · Self-approval** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#3-5-self-approval



Notes

Objective of module 4: understand who does what and how it is controlled.

SPAD divides work into eight roles. One is a person: the Orchestrator, who directs the process, chooses which AI fills each role and validates each artefact. The other seven are AI roles: Planner, Reviewer, Builder, Fixer, Analyst, Diagnostic AI and Security Reviewer.

Two rules order the roles:

- There are never two roles in the same phase.
- Whoever reviews is never whoever produced what is being reviewed.

The second part of the module covers the validation policy: what the person does when an AI response breaks the rules of the process.

Everyday example: in a financial audit, the auditor does not prepare the accounts being audited. SPAD applies the same separation to AI. That is why the name ‘AI Reviewer’ is used and not ‘auditor’: an auditor is an independent person with a different function; the AI Reviewer is a tool and a person always accepts its verdict.

Question for the classroom: why is it not enough to ask the same AI to ‘review its work carefully’?

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
- **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

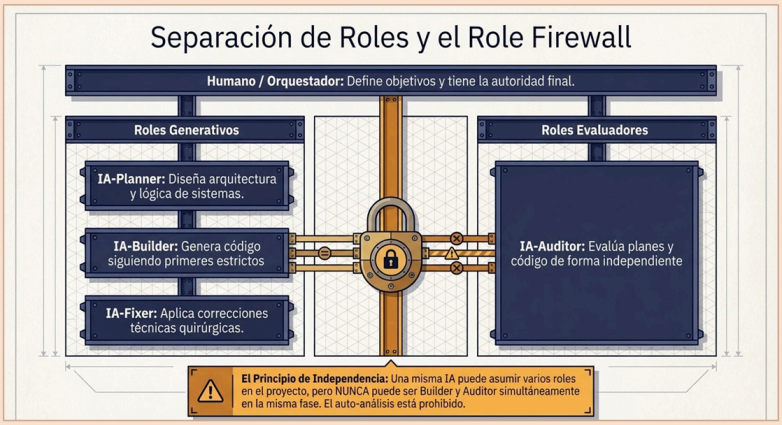
SOURCES AND LINKS

- **SPAD 01 · Roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.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** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles

SPAD COMPLETE COURSE

Whoever produces does not review

Role separation: the AI that builds and the AI that reviews are different, and a person decides.



- **Eight roles** A human Orchestrator and seven AI SPAD roles: Planner, Reviewer, Builder, Fixer, Analyst, Diagnostic AI and Security Reviewer.
- **Human review** A person always does the human code review; for sensitive code, two.
- **Illustration** Illustration in Spanish. Where it says 'IA-Auditor', read 'AI Reviewer'.

→ SPAD roles → Human code review

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 15 / 32

Notes

The illustration shows role separation with a padlock: the AI that builds and the AI that reviews do not mix. The illustration uses an old name, 'IA-Auditor'; the current name is 'AI Reviewer'.

The eight roles:

- Orchestrator (person): directs, chooses the models and validates each artefact.
- AI Planner: designs the plan and the test strategy.
- AI Reviewer: reviews plans, tests and code, and issues verdicts.
- AI Builder: implements the code following the primer.
- AI Fixer: applies the minimal fixes.
- AI Analyst: documents existing code (legacy cycle).
- Diagnostic AI: looks for the root cause of a failure; it does not write the solution.
- AI Security Reviewer: reviews sensitive code.

The same AI model can fill different roles, but never in the same phase or reviewing what it produced.

Human review: in addition to the AI Reviewer, a person always reviews the code and it is recorded. For sensitive code (authentication, authorisation, payments, personal data, agent limits, decisions about people or critical production) two human reviews are needed.

Example: in payment_gateway, one AI builds the charge, another reviews it, a developer reviews the code and, because it involves payments, a second human reviewer looks at it again.

Why it matters: separation prevents one model's errors from confirming themselves.

TERMS ON THIS SLIDE

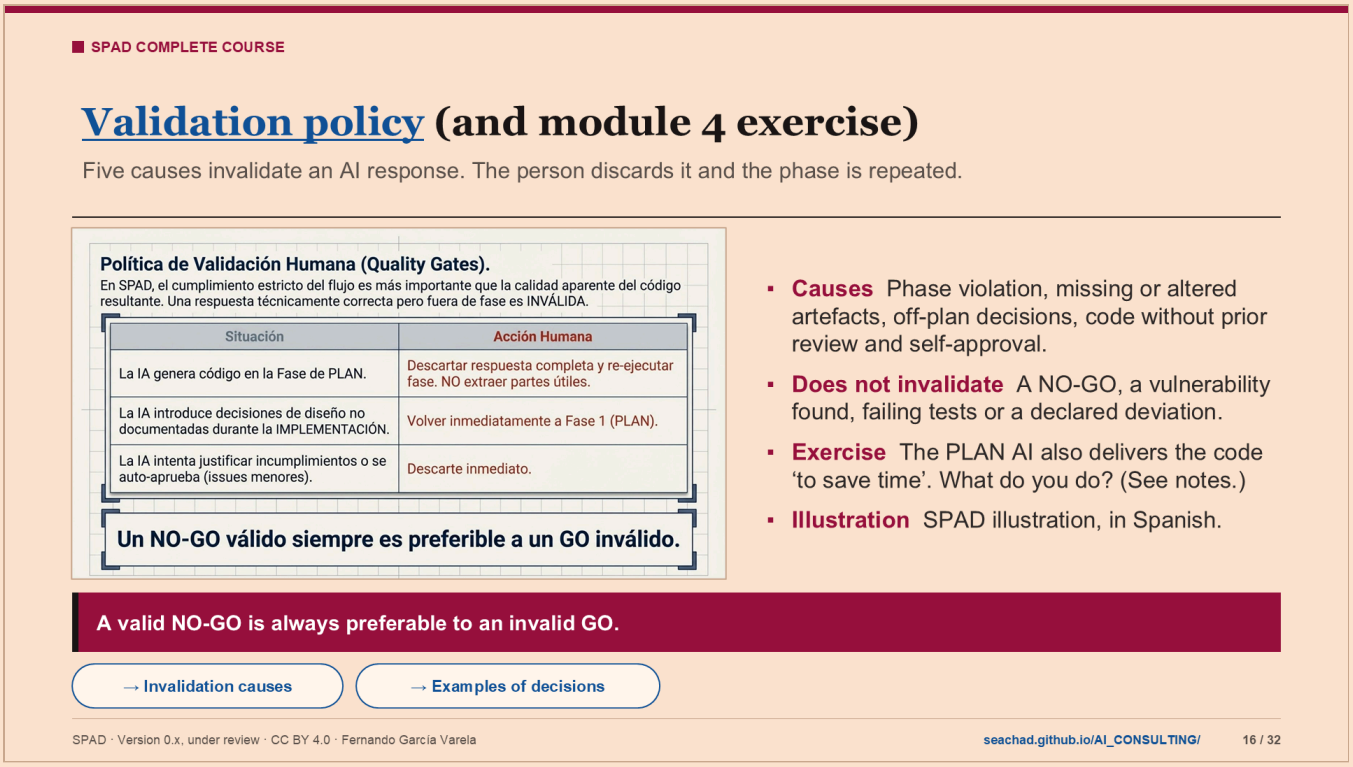
- **SPAD roles**: The roles in SPAD: the orchestrator (a person who leads the cycle and validates) and several AIs with separate functions: planner, reviewer, builder, fixer, analyst and diagnostic. Whoever builds does not review what

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

- **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 01 · Roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#roles
- **SPAD roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#roles



Notes

The validation policy (document 03) says what the person does when an AI response breaks the process. A person always applies it.

Five causes invalidate a response:

- Phase violation: it does something that does not belong to its phase, for example writing code in the PLAN.
- Missing or altered artefacts: a mandatory document is missing or an approved one has been changed.
- Off-plan decisions: the Builder changes the design on its own.
- Implementation without prior review: something is built on a plan that has not been reviewed.
- Self-approval: the AI approves itself or gives GO with open critical or high findings. It is the most serious cause.

What does not invalidate: a NO-GO, finding vulnerabilities, failing tests or a declared deviation. All of that is useful information.

Procedure for an invalid response: recognise the cause, do not fix the response, discard it in full, repeat the phase, reinforce the instructions and escalate if it happens again. The maximum number of attempts is a parameter (starting value: 3).

Answer to the exercise: the response is discarded in full, even if the code looks good. It is a phase violation. Phase 1 is repeated and it is entered in the violation log. 'The good part' is not kept.

Why it matters: if it is accepted once, the rule ceases to exist.

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

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
- **SPAD 03 · Procedure for an invalid response** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#procedure-when-a-response-is-invalid
- **SPAD 03 · Examples of decisions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#examples-of-decisions
- **Invalidation causes** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#invalidation-causes
- **Examples of decisions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#examples-of-decisions



Notes

Objective of module 5: understand SPAD's 'memory'.

An AI does not remember your company's rules from one piece of work to the next. If each person explains the rules to it in their own way, each piece of work comes out different. SPAD solves this with three pieces:

- Contexts: documents with the rules the AI receives before any phase. There is one global context, common to the whole organisation, and one per project.
- The topic: each feature, fix or incident has a short name and a folder where all its artefacts are stored.
- Records: which model produced each artefact, who validated it and which violations there were.

Everyday example: a professional firm has a common style manual (global context), instructions per client (project context) and a file per matter (topic). Nobody starts a matter without opening its file.

Why it matters: without contexts, quality depends on whoever writes the instructions. Without a topic, work gets lost in loose conversations. Without records, there is no way to answer an audit.

Question for the classroom: where are your company's technical rules today? Does the AI your teams use receive them?

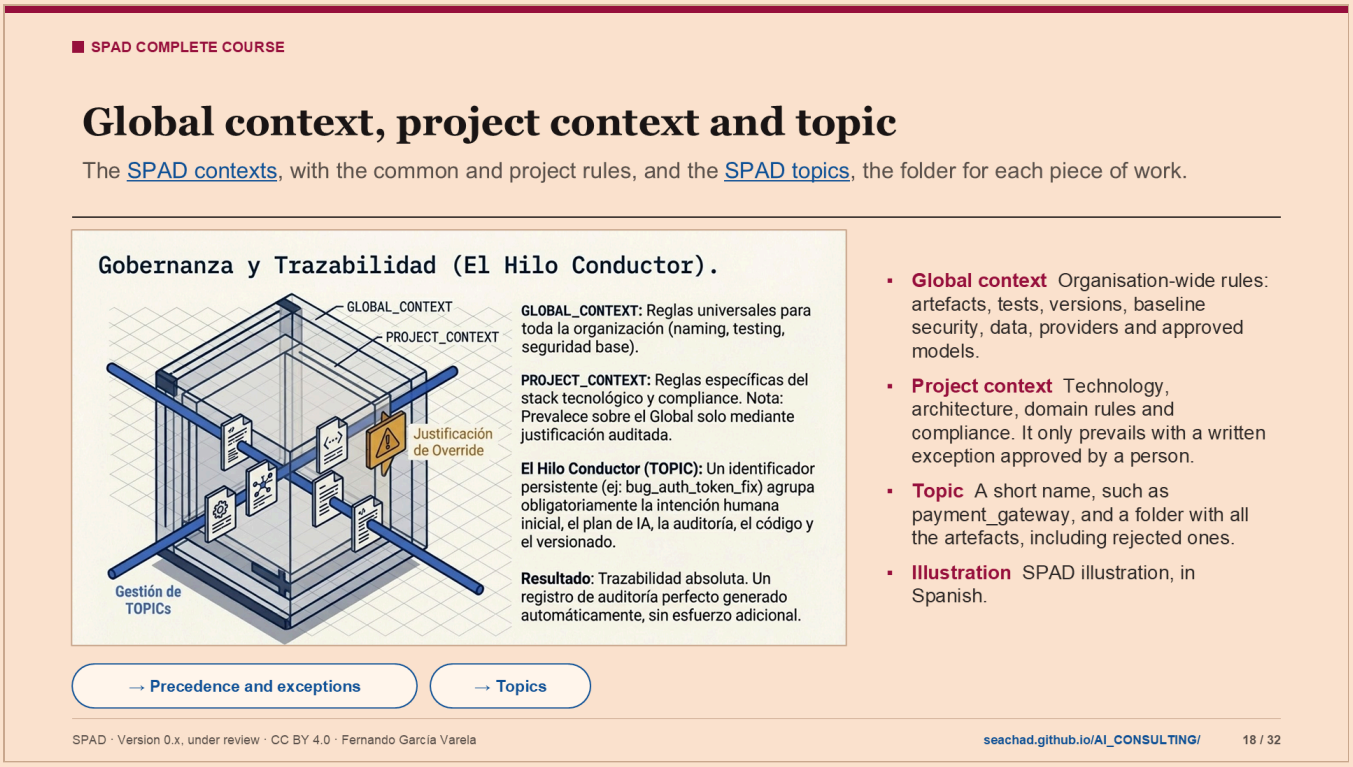
TERMS ON THIS SLIDE

- **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 00 · Contexts and topics** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#contexts
- **Contexts, topics and records** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html



Notes

The illustration shows the three pieces that the AI receives or uses in each piece of work.

Global context: the organisation sets it and it applies to all projects. It includes how artefacts are named, the minimum test coverage (the document proposes 80 % as a starting value, which each organisation adjusts), versioning, baseline security (with the OWASP Top 10 as a reference), the rules on data and model providers, the prohibitions, the parameters of the complementary cycles and the approved models.

Project context: adds what is specific to the project: technology, architecture, business rules, compliance, conventions and deployment. If it contradicts the global context, it only prevails with a written, justified exception approved by a person, never by an AI. Example from the document: lowering coverage from 80 % to 70 % during a migration, with a reason and a date.

Topic: a short name in lower case, with underscores and up to 20 characters, such as user_auth or payment_gateway. Each topic has its folder. Rejected attempts are kept: if the first plan was rejected, the folder has the rejected plan 01 and the approved plan 02.

Why it matters: the written exception turns an informal decision (‘this time we will skip the tests’) into a decision with an owner, a reason and a date.

Question for the classroom: who should approve an exception to the global context in your company?

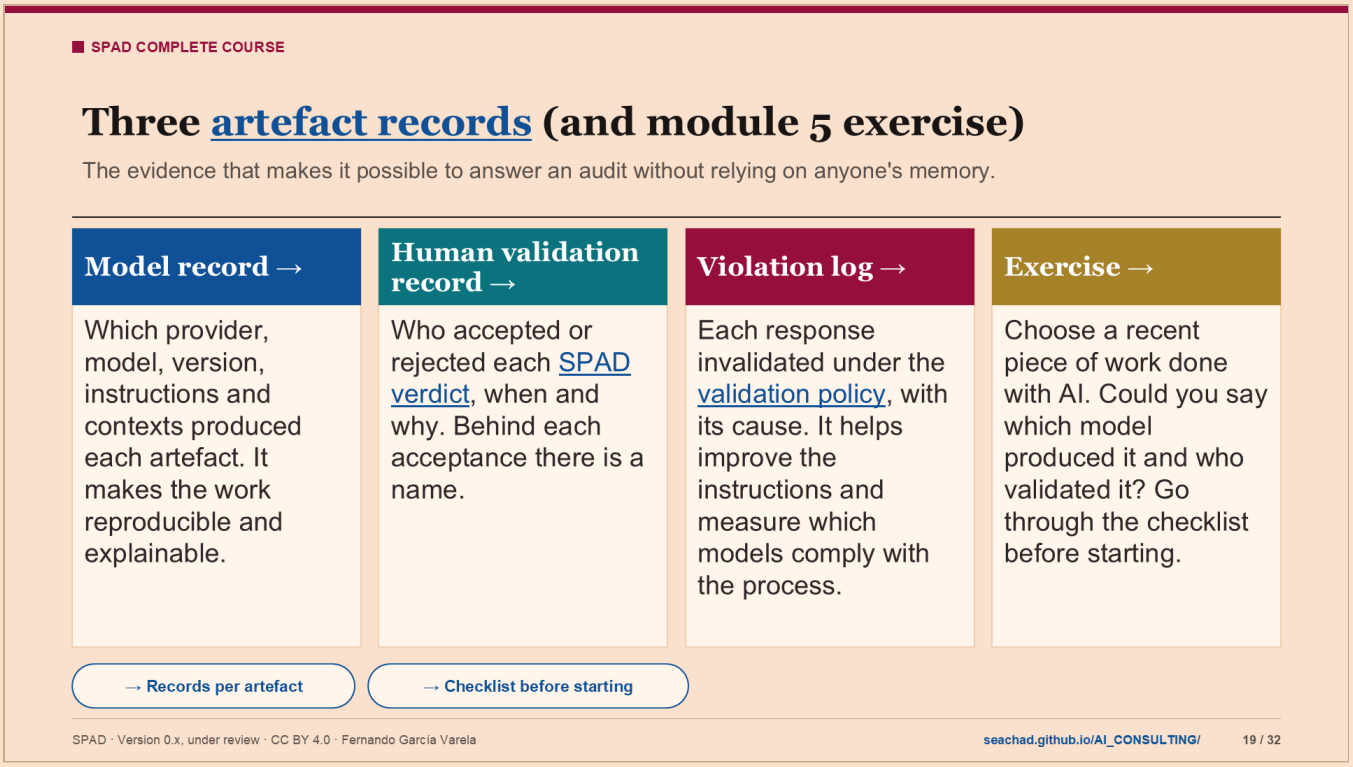
TERMS ON THIS SLIDE

- **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

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 02 · Project context** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#2-2-project-context
- **SPAD 02 · Organisation of artefacts** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#3-1-organisation-of-artefacts
- **OWASP Top 10 (official source)** https://top10.owasp.org/2025/
- **Precedence and exceptions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#2-3-precedence-and-exceptions



Notes

Each topic keeps three records.

Model record: each artefact states which provider, model and version produced it, with which instructions and with which contexts. Models change often. If a defect appears in six months, this record makes it possible to know what it was built with and to repeat it.

Human validation record: each accepted or rejected verdict carries the person's name, the date and the reason. It is the rule 'no AI approves' turned into evidence.

Violation log: each response discarded under the validation policy, with its cause. It is not a log of blame: it helps improve the instructions and compare which models comply better with the process, as we will see in the module 8 metrics.

Module 5 exercise (five minutes, individual): choose a recent piece of work done with AI in your team and answer: which model produced it?, with which instructions?, who validated it?, were there discarded attempts? Then go through the checklist in document 02, which summarises what must be ready before starting: contexts loaded, topic created, roles assigned and models approved.

Expected answer: almost nobody will be able to answer all four questions. That is the value of the records.

Why it matters for a technology leader: records cost little if they are kept from the start and a lot if they are rebuilt for an audit.

TERMS ON THIS SLIDE

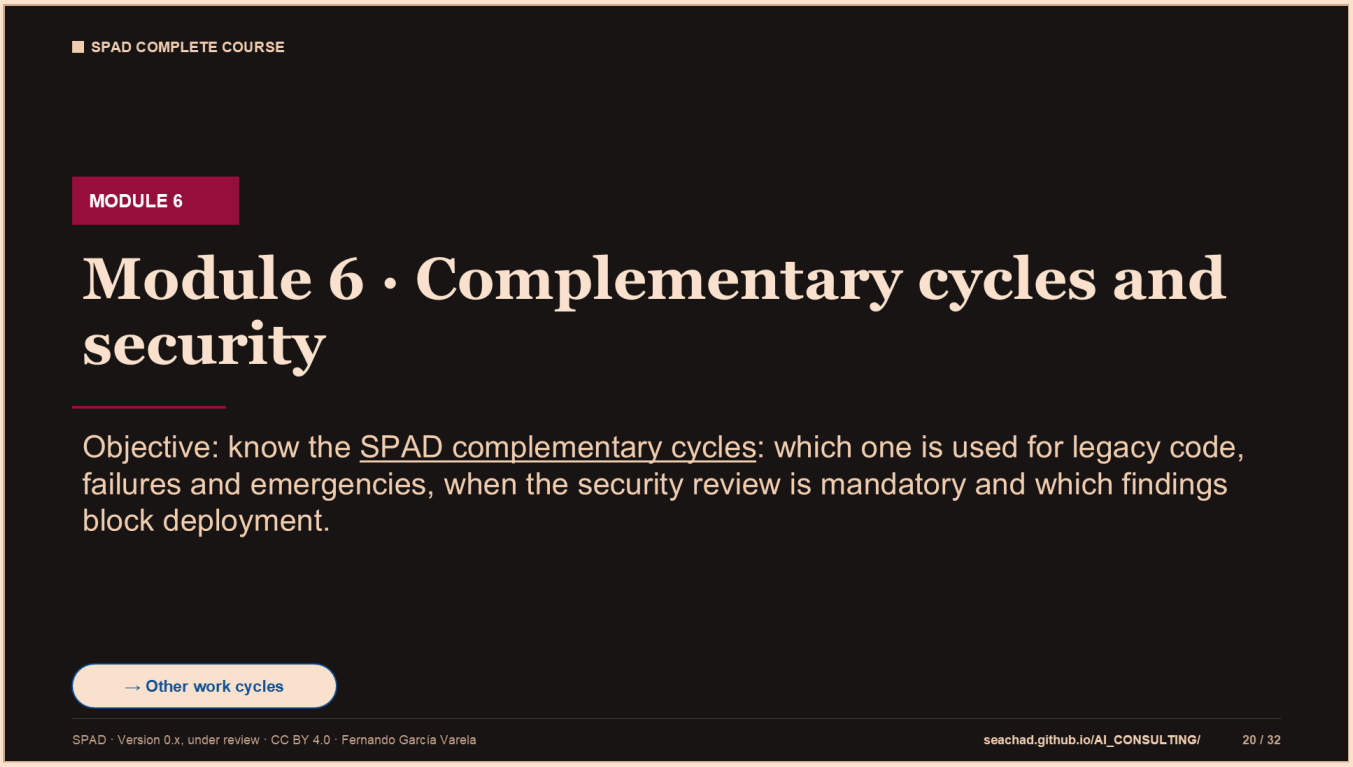
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. 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

- **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 02 · Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact
- **SPAD 03 · Validation record and violation log** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#validation-record-and-violation-log
- **SPAD 02 · Checklist** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting
- **Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact
- **Checklist before starting** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#checklist-before-starting



Notes

Objective of module 6: know the cycles that accompany the main one.

Not all work starts from scratch. Sometimes a legacy system that nobody documented has to be changed, someone has to investigate why something fails or an urgent problem in production has to be fixed. SPAD does not invent a different method for each case: it defines complementary cycles that prepare or accelerate the main cycle, without skipping its rules.

The module also covers security. In SPAD, security is not an optional review at the end: for sensitive code it is a mandatory phase, with clear rules on which findings block deployment and who may accept a risk.

Everyday example: a hospital has an emergency protocol. It goes faster, but it does not skip patient identification or the report afterwards. SPAD's urgent fix works in the same way.

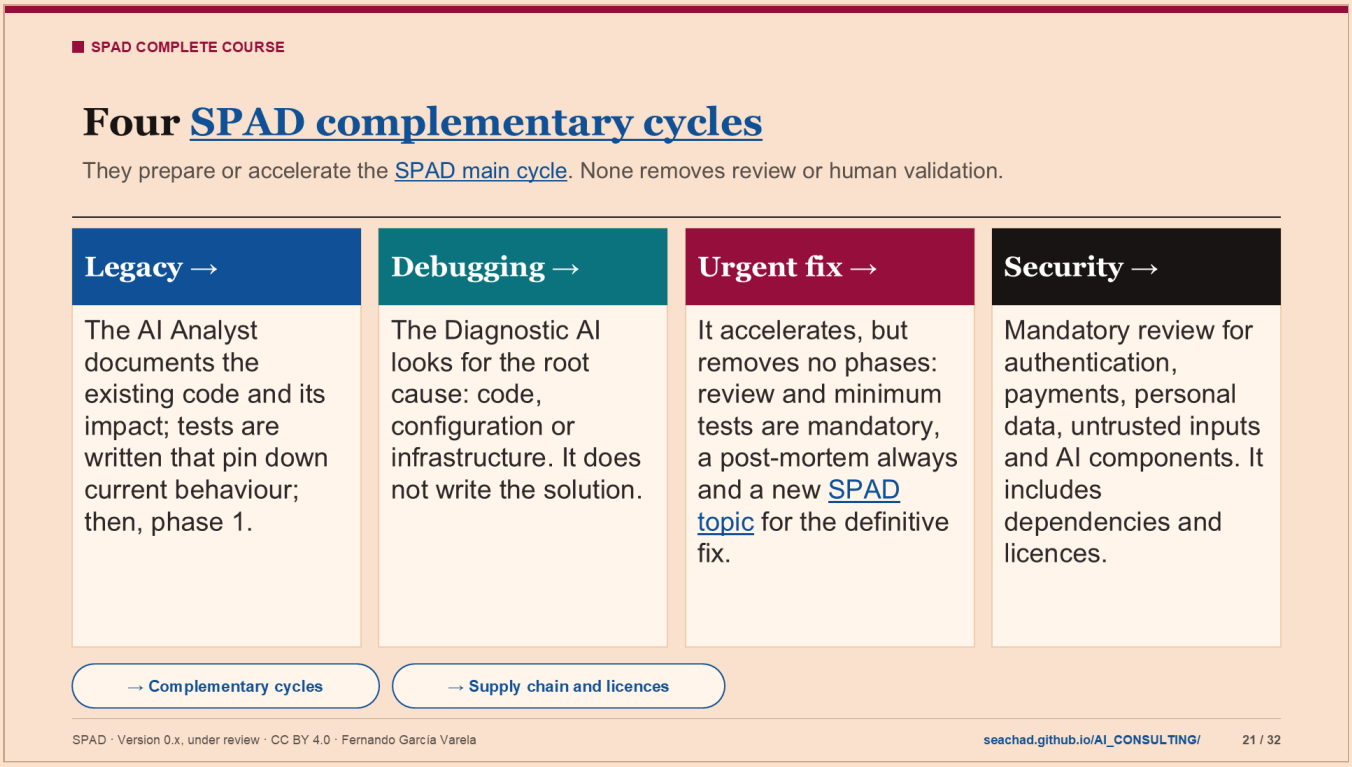
Question for the classroom: when there is an emergency in production in your company, which controls are skipped? Are they reviewed afterwards?

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

SOURCES AND LINKS

- **SPAD 00 · Other work cycles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#other-work-cycles
- **Other work cycles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#other-work-cycles



Notes

The four complementary cycles in document 04:

- Legacy: before changing a system that nobody documented, the AI Analyst documents it and analyses the impact of the change. Then characterisation tests are written: tests that pin down how the system behaves today, even if it is not perfect. That way, any change that alters that behaviour is detected. Only then does phase 1 start.
- Debugging: the Diagnostic AI investigates a failure and delivers a root cause report with a verdict: the problem lies in the code, the configuration or the infrastructure. It does not write the solution; the solution follows the main cycle.
- Urgent fix: for incidents in production. It accelerates, but removes no phases. Review and minimum tests are mandatory, the post-mortem is mandatory even if the fix worked, and a provisional fix opens a new topic to plan the definitive solution. The deadlines are set by the global context.
- Security: the S1 review is mandatory for authentication, authorisation, payments, personal data, untrusted inputs and AI components in production. It includes the supply chain: that dependencies exist, their known vulnerabilities, their licences and a software bill of materials.

Example: a duplicate charge in payment_gateway. First, debugging to find the cause. If it is urgent, an urgent fix with review and post-mortem. Then, a new topic with the definitive fix through the main cycle.

Why it matters: an emergency is when most controls get skipped; SPAD speeds it up without leaving it uncontrolled.

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 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

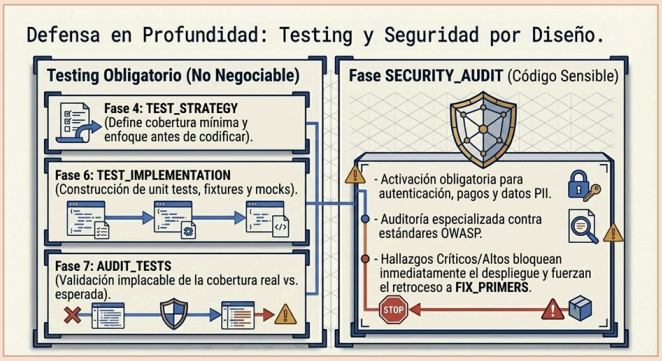
SOURCES AND LINKS

- **SPAD 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 · Security cycle** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#security-cycle
- **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 COMPLETE COURSE

Testing and security: what blocks (and module 6 exercise)

Mandatory tests in phases 4, 6 and 7 of the SPAD main cycle, and a security review of sensitive code.



- **Critical and high** They block deployment. Nobody accepts a critical one; a high one, only exceptionally, in writing and with a date.
- **Medium and low** They do not block. They are fixed or a person with authority accepts them as residual risk.
- **Exercise** On Friday a high vulnerability appears and the launch is on Monday. Who decides and what is put in writing? (See notes.)
- **Illustration** Illustration in Spanish. FIX_PRIMERS is phase 9, fixes.

No AI accepts a residual risk: the AI Security Reviewer classifies and recommends; a person accepts.

→ Severities and risk acceptance

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 22 / 32

Notes

The illustration has two parts. On the left, the mandatory tests: the strategy is defined in phase 4, before writing code; the tests are built in phase 6; and in phase 7 real coverage is checked against expected coverage. On the right, the security review of sensitive code, with OWASP as the reference list (for AI components, also the OWASP list for applications with language models).

The severities (document 04, section 5.3):

- Critical: blocks deployment, goes back to fixes and a new security review. Nobody may accept it as residual risk.
- High: blocks deployment. Only exceptionally does the risk authority defined in the global context accept it, in writing and with a fix date.
- Medium: does not block. It is fixed or accepted by the person with authority under the global context.
- Low: does not block. It is fixed or accepted by the Orchestrator.

A residual risk is never accepted by an AI: always by a person with authority, in writing.

Answer to the exercise: the high vulnerability blocks the launch. The AI does not decide. Only the risk authority in the global context may accept it, exceptionally, in writing and with a fix date. If it were critical, there would be no launch on Monday.

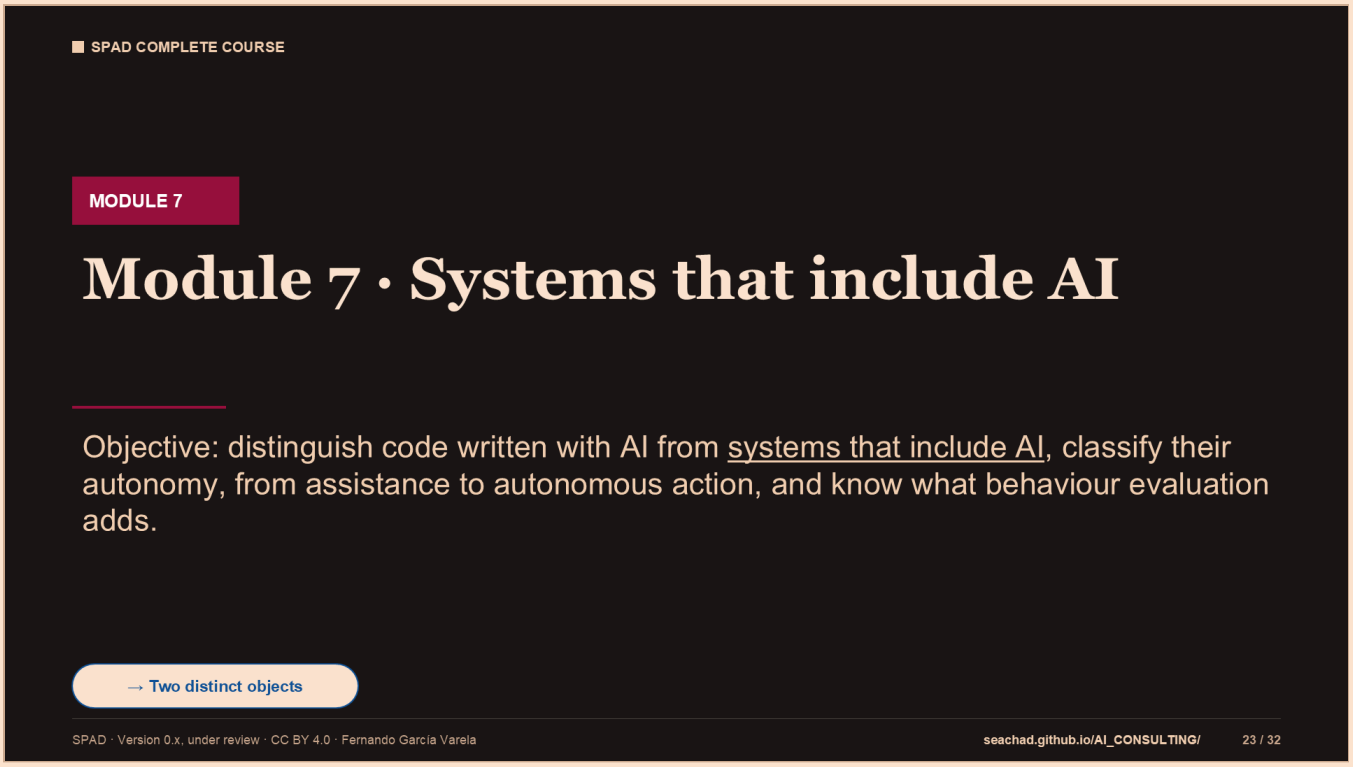
Why it matters: deadline pressure is real; the written rule protects whoever has to say no.

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

SOURCES AND LINKS

- **SPAD 04 · Severities 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 (official source)** https://top10.owasp.org/2025/
- **OWASP Top 10 for LLM applications** https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/
- **Severities and risk acceptance** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/04_SPAD_Ciclos_complementarios.html#5-3-severities-blocking-and-risk-acceptance



Notes

Objective of module 7: understand the highest-risk case.

So far we have talked about code written with the help of AI: the AI produces code, tests and documentation, which are reviewed before deployment. Once in production, the program always does the same thing.

A system that includes AI is something else: its behaviour in production depends on a model, on some instructions, on a knowledge base or on an agent that executes actions. It responds to inputs nobody has seen before. And it can change behaviour without anyone touching the code: it is enough for the provider to update the model.

Example: a program that calculates the VAT on an invoice is code. An assistant that answers customers' questions about their invoice is a system that includes AI. The first is tested once; the second must be evaluated beforehand and monitored during use.

Risks specific to the second: incorrect answers given with confidence, prompt injection, data leakage, unauthorised actions, bias and drift.

Question for the classroom: how many of your company's AI projects are of one type and how many of the other?

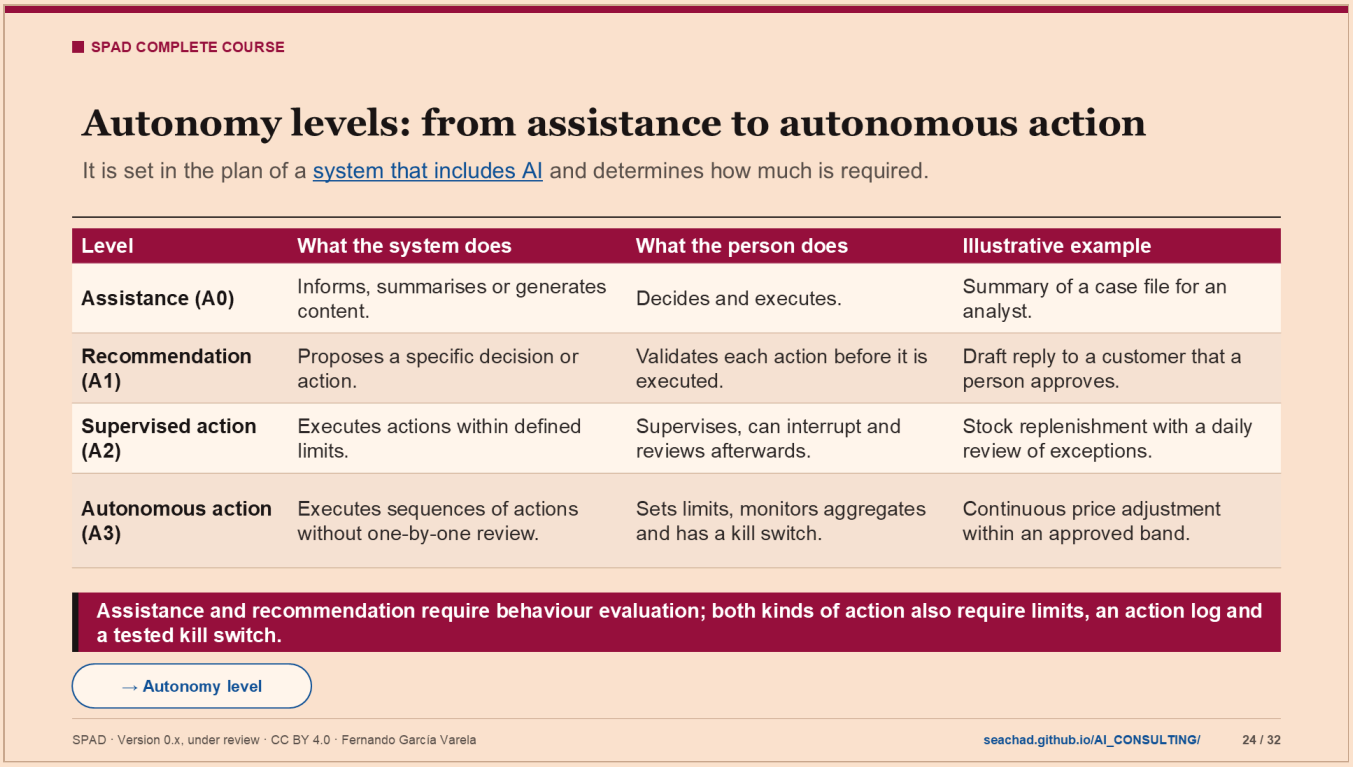
TERMS ON THIS SLIDE

- **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

- **Two distinct objects** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#two-distinct-objects



SPAD COMPLETE COURSE

Autonomy levels: from assistance to autonomous action

It is set in the plan of a system that includes AI and determines how much is required.

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.	Draft 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 a daily review of exceptions.
Autonomous action (A3)	Executes sequences of actions without one-by-one review.	Sets limits, monitors aggregates and has a kill switch.	Continuous price adjustment within an approved band.

Assistance and recommendation require behaviour evaluation; both kinds of action also require limits, an action log and a tested kill switch.

→ Autonomy level

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 24 / 32

Notes

Every AI component that decides or acts is classified from the plan onwards with a four-level autonomy scale, from A0 to A3. The table summarises document 05; the examples are illustrative.

The key for a business audience is in the person column. In A0 the person decides everything. In A1 they validate each action. In A2 they no longer validate one by one: they supervise and review afterwards. In A3 they only set the limits and monitor the whole, with a kill switch.

The more the system acts, the more is required:

- A0 and A1: behaviour evaluation.
- A2 and A3: in addition, action limits, a log of each action, a tested kill switch and an agent plan with its tools and permissions.

Example for debate: an assistant that drafts replies to complaints is A1 if a person approves them before they are sent. If it is decided to send them automatically to save time, it becomes A2. That is not a technical adjustment: it is a business decision that changes the requirements.

Question for the classroom: think of an AI use case in your sector. What level is it at today? What level do you want to take it to? What would you have to demonstrate first?

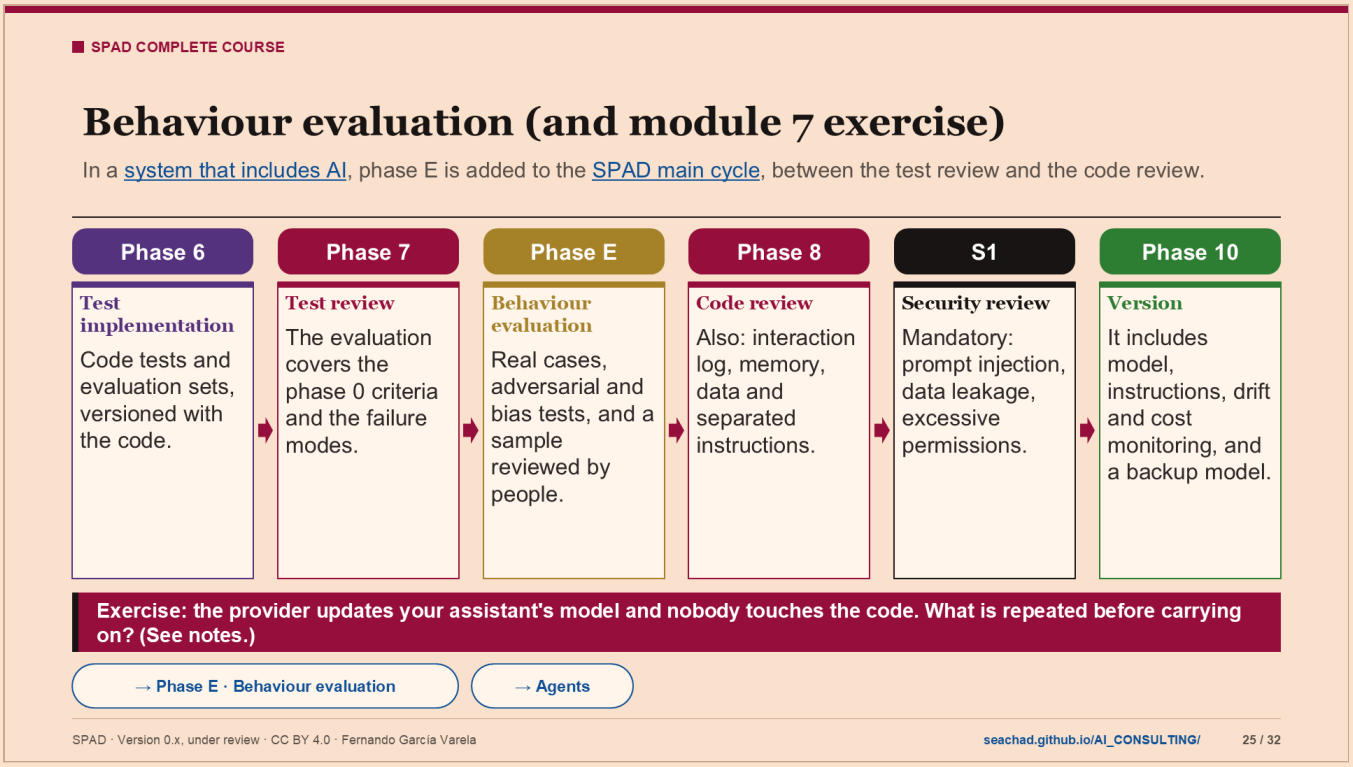
TERMS ON THIS SLIDE

- **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 · Autonomy level** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#autonomy-level
- **SPAD 05 · Agents** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#agents

- **Autonomy level** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#autonomy-level



Notes

Code tests verify that the program does what the code says. Behaviour evaluation verifies that the system does what the objective says.

Phase E runs the system on an evaluation set: cases representative of real use, with the expected answer or the criterion for judging it, including cases the system must refuse. It includes:

- Acceptance criteria with thresholds set in advance (accuracy, correct refusal, latency, cost).
- Adversarial tests: prompt injection attempts, direct or hidden in a retrieved document.
- Bias tests with counterfactual pairs: the same input changing only a protected attribute. They are mandatory if the system decides, recommends or communicates with people.
- A sample reviewed by people.
- The backup model, if there is one, with the same set.

A NO-GO sends work back to phase 1 if the failure is a design one and to phase 5 if it lies in the instructions.

Agents: tool contracts with minimum permissions, limits, controlled memory, an action log, their own identity and a tested kill switch. A key rule: retrieved content is data, not instructions.

Answer to the exercise: the behaviour evaluation is repeated with the same versioned set before continuing to use the system. In operation, things change without the code changing: the model, the instructions, the knowledge base, usage and cost.

Why it matters: without an evaluation set, the only proof that the assistant works is a demonstration with the questions that occurred to someone.

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 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

SOURCES AND LINKS

- **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 · What it adds to each phase** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#what-it-adds-to-each-phase
- **SPAD 05 · What changes without the code changing** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#operation-what-changes-without-the-code-changing
- **OWASP Top 10 for LLM applications** https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/
- **OWASP Top 10 for agentic applications** https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/
- **Phase E · Behaviour evaluation** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#phase-e-behaviour-evaluation
- **Agents** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/05_SPAD_Sistemas_que_incluyen_IA.html#agents



Notes

Objective of module 8: measure without promising.

A methodology that is not measured becomes a ritual. SPAD provides two instruments:

- The self-assessment of conformity: an internal review, done by a person independent of those who built, that checks whether a piece of work was done in line with SPAD.
- The process metrics: indicators calculated from the records in module 5.

An important message for the classroom: SPAD does not claim improvement figures. It does not say ‘it reduces incidents by a certain percentage’. Each organisation measures against its own starting point: rework after delivery, incidents per delivered piece of work, hours to prepare an audit and time to production, before and after using SPAD.

The module closes with the reduced version, because the metrics also tell you when the full method is excessive for a small piece of work.

Everyday example: a training plan does not promise how many kilos you will lose; it tells you what to measure each week to know whether it works.

Question for the classroom: if a provider promises you an exact improvement figure with their method, what would you ask for before believing it?

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

- **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

- **Self-assessment and metrics** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html

SPAD COMPLETE COURSE

Process metrics: your organisation supplies the figures

Six of the eight SPAD process metrics, calculated from the artefact records of each topic.

Metric	How it is calculated	What it shows
Invalidation rate per phase	Invalid responses ÷ responses in the phase × 100	Phases with weak instructions or unsuitable models.
NO-GO rate	Reviews with NO-GO ÷ reviews × 100	Quality before review. Zero for months: the review is a formality.
Iterations to GO	Mean and maximum per reviewed phase	Cost of rework within the cycle.
Time per phase	Median from entry to validation	Bottlenecks; basis for using the reduced version.
Escaped defects	Defects in production ÷ topics delivered	Effectiveness of the reviews. The figure that really matters.
Fixes outside a finding	Fixes without a finding ÷ fixes × 100	Covert refactoring. It should tend to zero.

The conformity self-assessment is not a certification: it is always recorded as 'conforming with SPAD (self-assessment)'.

→ Process metrics → Conformity conditions

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 27 / 32

Notes

The table shows six of the eight process metrics. The other two are the invalidation rate per model (which models follow the rules best) and real versus required coverage. For systems with AI, the behaviour evaluation metrics are added and, in operation, drift and cost per interaction.

How to read them, with examples from document 08:

- A high invalidation rate in a single phase shows that its instruction needs strengthening; it is not a problem with the method.
- A NO-GO rate close to zero for months is not good news: it usually shows that the review has become a formality.
- If the time per phase greatly exceeds building time on small pieces of work, the reduced version is advisable.
- Escaped defects are the figure that really matters: they measure whether the reviews work.

The self-assessment checks six conditions (seven if the system includes AI): decisions were taken before the code, each phase produced its artefact, reviews were independent, no phase was skipped, fixes were minimal and no AI approved anything. If a mandatory artefact is missing, the result is negative. The result is conforming, conforming with conditions or non-conforming, and its validity is set by the global context (starting value: twelve months).

It is neither a certification nor a seal: presenting it as such breaches the permitted use of the SPAD name.

Question for the classroom: which metric would you ask your team for in the first quarter?

TERMS ON THIS SLIDE

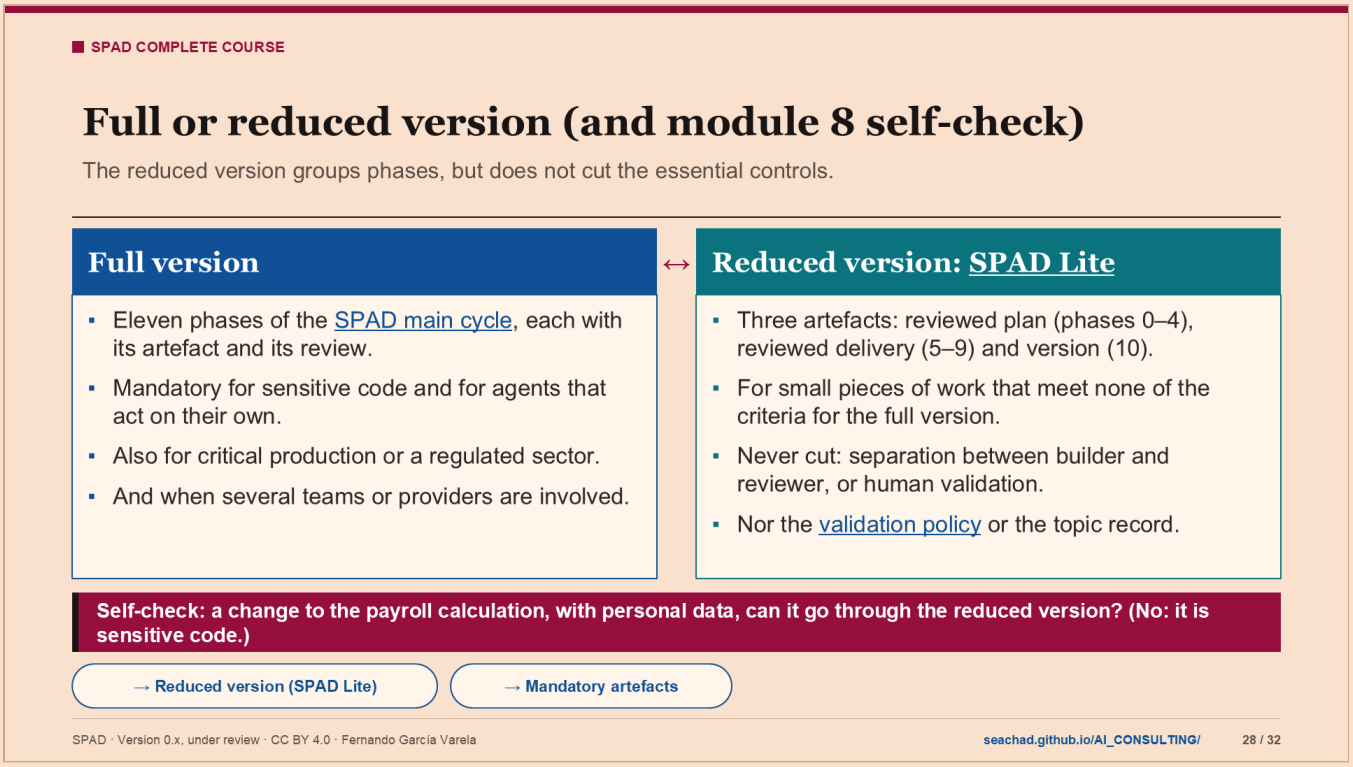
- **Process metrics**: Indicators measured by a team using SPAD, such as how many reviews end in NO-GO, how many defects reach production or how long a topic takes. SPAD promises no figures: it proposes measuring them

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

- **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
- **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 08 · Self-assessment procedure** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#procedure
- **SPAD 08 · When SPAD is worth using** 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

Full SPAD does not make sense for every small change. That is why the operating guide defines a reduced version, SPAD Lite, with three artefacts:

- Reviewed plan: groups phases 0 to 4.
- Reviewed delivery: groups phases 5 to 9.
- Version: phase 10.

Phases are grouped, but they do not disappear (rule 1). And there are four things that are never cut: the separation between builder and reviewer, human validation, the validation policy and the topic record.

The full version is used if any of these criteria is met: sensitive code (authentication, payments, personal data...), A2 or A3 agents, critical production, several teams or providers, or a regulated sector.

Document 00 adds when to use SPAD: it is highly advisable for systems that go into production, that touch sensitive data, money or decisions about people, or in regulated sectors. It is not necessary for disposable prototypes without real data. And at first it makes work slower, while the team learns.

Module 8 self-check (answers):

- Payroll change with personal data: full version, because it is sensitive code.
- Text change on an internal screen, with no personal data or money: it can go through the reduced version.
- Prototype for an internal demonstration, with no real data: SPAD is not necessary.

Why it matters: proportionality prevents the method from being abandoned as too heavy.

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 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
- **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
- **Mandatory artefacts** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#mandatory-artefacts



Notes

Objective of module 9: bring SPAD into the management conversation. It is the most important module for an executive audience.

SPAD is about how software is built with AI. For a company it has two uses.

Use 1 · Build its own software with AI, with method. If its teams already use code assistants or agents, SPAD gives them a cycle with blocking phases, independent reviews, human validation and artefacts that can be audited. The result is software that can be maintained and explained, not only code that works in a demonstration.

Use 2 · Require as a client. Its suppliers and integrators also build with AI and are more productive. It is the author's position that this gain must reach the client: as lower cost or as more scope for the same price. To negotiate it with data, the company can ask for what SPAD already defines: the artefacts of each phase, the model record, the human validation record, the verdicts of each review and the process metrics.

Both uses rest on the same thing: AI work that leaves a trail. Without a trail, the company does not know what the AI did, who reviewed it or how much effort it really cost.

Question for the classroom: does your company know whether its suppliers build with AI, and whether that gain appears in their invoices?

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 · 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

SPAD COMPLETE COURSE

What to ask from a supplier that builds with AI

Everything is already defined in SPAD, from artefact records to SPAD verdicts. No need to invent a new contract requirement.

What to ask for	What it tells the company	Where it is defined
Artefacts of each phase	That there was a plan, tests and reviews, not only delivered code.	Operating guide (01)
Model record	Which model, version and instruction were used for each artefact.	Global context (02)
Human validation record	Which person reviewed and accepted what the AI did, and when.	Global context (02)
Verdicts of each review	How many GO, GO WITH CHANGES and NO-GO there were, and what was fixed.	Operating guide (01)
Process metrics	Effort, correction cycles and escaped defects: the basis for negotiation.	Self-assessment and metrics (08)

The supplier's productivity gain must show in the invoice: lower cost or more scope for the same price.

→ Records per artefact → Process metrics

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 30 / 32

Notes

This table is the practical list for use 2: what the company can ask, as a client, from a supplier or an integrator that builds with AI.

Nothing in the table is a new requirement. They are artefacts and records that SPAD already defines:

- The artefacts of each phase (plan, code primer, test strategy, review reports, version) show that there was method.
- The model record says which provider, model, version and instruction produced each artefact.
- The human validation record says which person reviewed and accepted each result. A verdict from an AI reviewer never replaces that person.
- The verdicts say how many reviews passed the first time and how many went back.
- The process metrics give the real effort, the correction cycles and the defects that reached production.

With these data the company can negotiate. It is the author's position that, if the supplier takes less time thanks to AI, that gain must be passed on to the client: as lower cost or as more scope for the same price. SPAD sets no figure: the figure comes from each supplier's metrics.

A warning: asking for the records is not enough. The company has to read them. A NO-GO rate close to zero for months usually shows that the review has become a formality.

Why it matters: it turns a commercial promise (‘we use AI’) into data that can be checked in a contract.

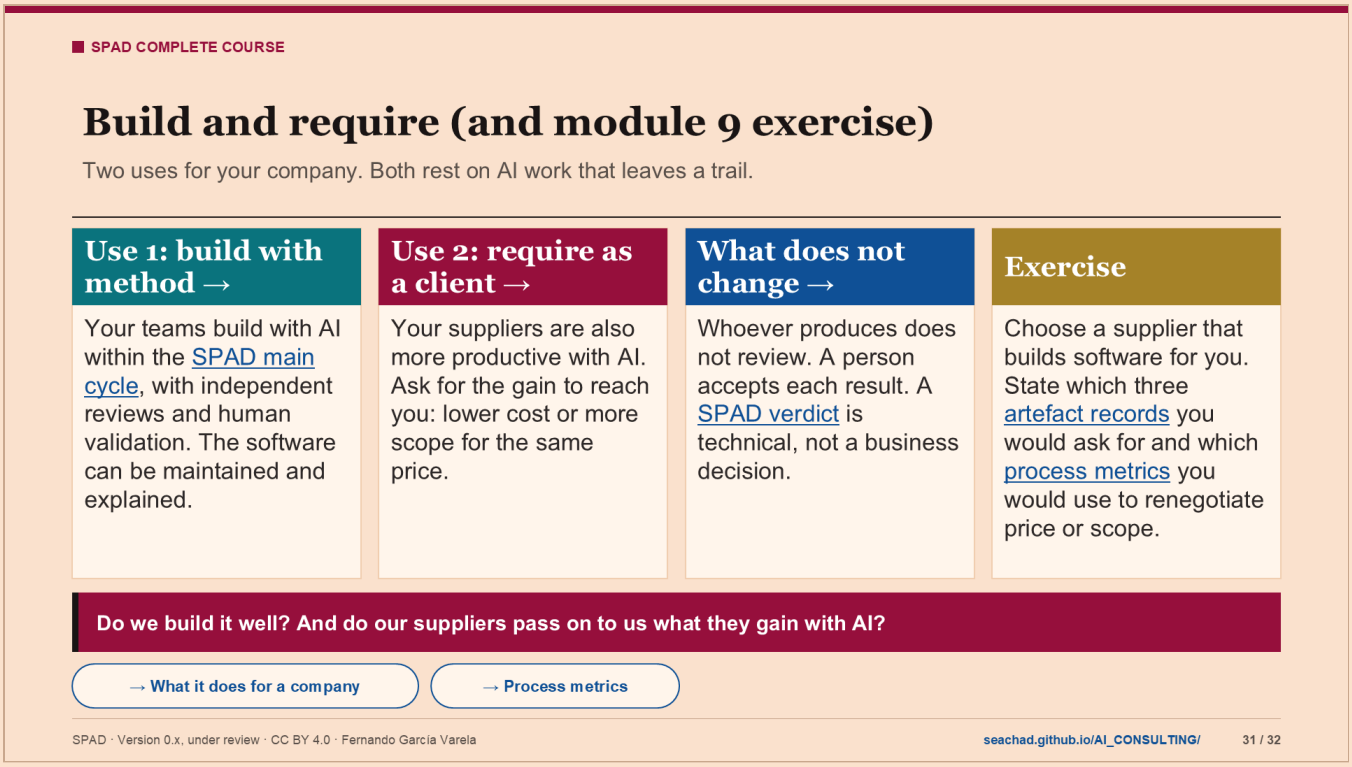
TERMS ON THIS SLIDE

- **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

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
- **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
- **Records per artefact** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#records-per-artefact



Notes

This slide summarises module 9 in two uses and one rule that does not change.

Use 1: the company builds its own software with AI, with method. It gains traceability, quality and the ability to explain what the AI did and who accepted it.

Use 2: the company requires as a client. It is the author's position that its suppliers' productivity gain must be passed on: as lower cost or as more scope for the same price. SPAD's artefacts, records, verdicts and metrics give the data for that conversation.

What does not change in either use: whoever produces does not review their own output; a person accepts each result; and a technical GO only says that the work is well built, not that the company should put it into production.

Module 9 exercise (ten minutes, in groups). Fictitious example answer: an integrator maintains the company's e-commerce website. I would ask for the model record, the human validation record and the process metrics for the last six months. I would use the effort per change and the correction cycles: if effort goes down and the rate does not, I would propose a lower price per change or more changes for the same fee. If the integrator cannot provide those records, that is also an answer.

Why it matters: AI changes the cost of building software, inside and outside the company. Without data, someone else keeps that change.

TERMS ON THIS SLIDE

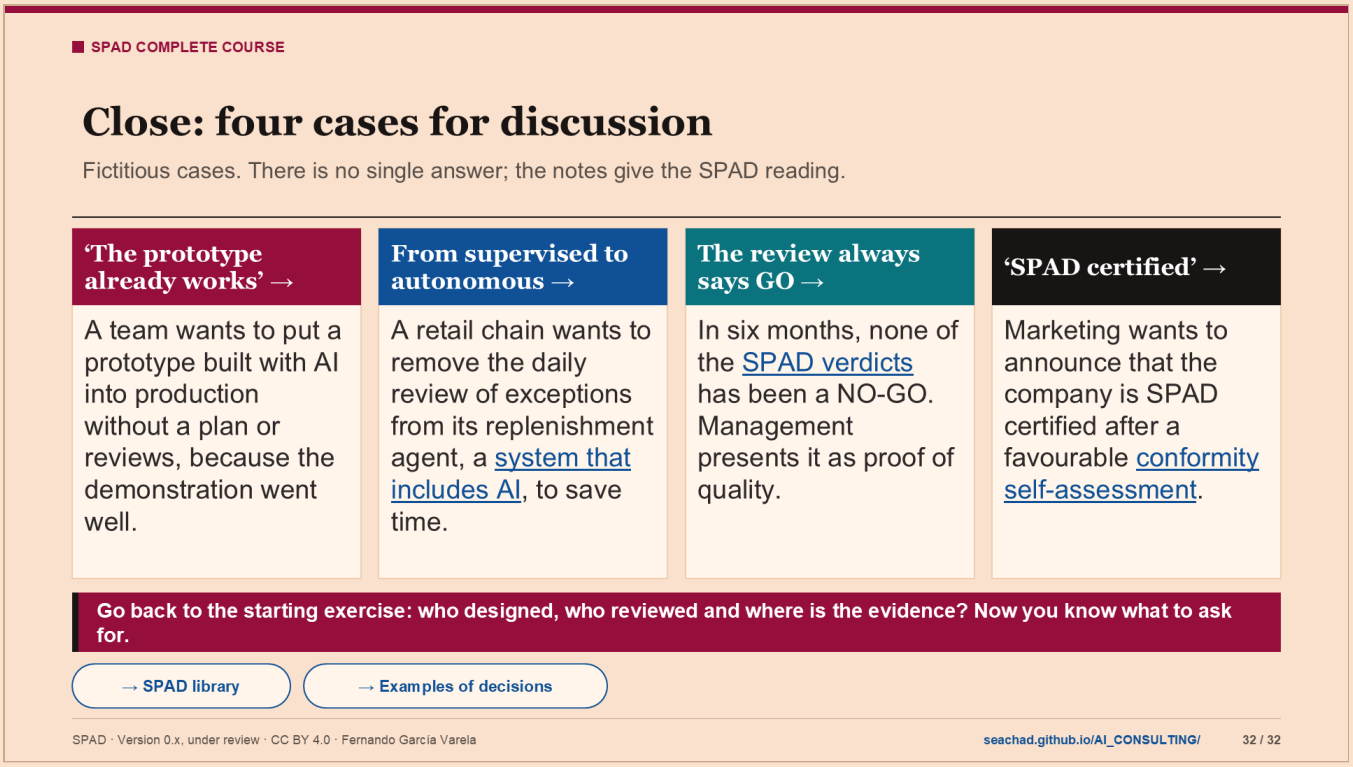
- **SPAD main cycle** (fases 0–10): SPAD's eleven phases: context, plan, plan review, code guide, test strategy, implementation, testing, test review, code review, fixes and release management. Each one leaves an artefact that is reviewed. The link opens the cycle in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa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

- **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 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 03 · Self-approval** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#3-5-self-approval
- **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



Notes

Four fictitious cases to debate in groups (fifteen to twenty minutes). SPAD reading for the teacher:

- 'The prototype already works': a disposable prototype without real data does not need SPAD. But as soon as it goes into production, with real data or customers, it is no longer a prototype. The reasonable course is to open a topic and go through the cycle: at least the reduced version; if it involves personal data, money or decisions about people, the full one, with human review, tests and security.
- From A2 to A3: removing the daily review turns a supervised agent into an autonomous one. It is not a technical adjustment: it is a decision with new requirements (strict limits, monitoring of aggregates, a tested kill switch) and the full version of SPAD. A person with authority must decide it, not the team to save time.
- The review always says GO: according to document 08, a NO-GO rate close to zero for months usually shows that the review has become a formality. Its independence must be checked and escaped defects examined.
- 'SPAD certified': SPAD does not certify. The self-assessment is always recorded as 'conforming with SPAD (self-assessment)'. Announcing a certification breaches the permitted use of the name and exposes the company.

Close: SPAD is published under a CC BY 4.0 licence (content) and MIT (code), citing the authorship: SPAD · Fernando García Varela. Documents 06 and 07 contain the instructions and contracts for each phase for anyone who wants to apply it.

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.

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
- **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 · 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 07 · Input and output contracts** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/07_SPAD_Contratos_de_entrada_y_salida.html#purpose
- **SEVEN-G 53 · Building solutions with AI (only bridge)** https://seachad.github.io/AI_CONSULTING/SEVEN-G/html/en/53_SEVEN-G_Construccion_de_soluciones_con_IA.html
- **SPAD 00 · Status and licence** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/00_SPAD_Que_es_y_para_que_sirve.html#status-licence-and-next-steps
- **SPAD library** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/index.html
- **Examples of decisions** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html#examples-of-decisions

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/