SPAD IN 1 SLIDE · STRUCTURED PROMPT-DRIVEN ENGINEERING

Building software with AI, under control

Think, validate, execute: a method so that the code AI writes can be reviewed, explained and audited. Illustration in Spanish.

The problem →

AI writes code very fast. Without a method, nobody decides the design, tests are missing and no evidence remains. Speed turns into debt and risk.

Eleven phases that block →

The phases run from context to plan, tests, code and version. Without validation, nothing advances. Small work: SPAD Lite.

Each AI, a single role →

One AI plans, another builds and a different one reviews: separate roles. No AI approves its own work.

GO · GO with changes · NO-GO →

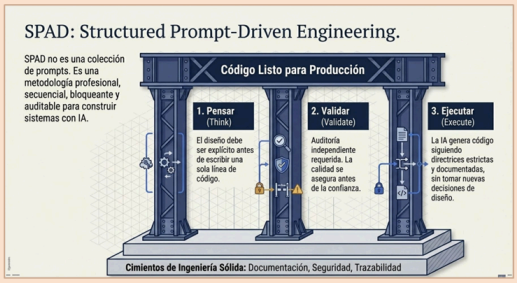
Each review gives a verdict. A person accepts it and also does the human code review.

Everything is written down →

Contexts set the rules; one topic per piece of work; records per artefact: which model and who validated.

For your company: two uses →

Build your software with AI with method. And ask suppliers for phases, records and process metrics: so productivity reaches you as cost or scope.



SPAD promises no figures and certifies nothing: it leaves evidence so that your company can measure and audit what AI builds.

→ What SPAD is | → Operating guide: phases and roles | → What it does for a company | → Process metrics

SPAD · Version 0.x, under review · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 1 / 1

Notes

SPAD (Structured Prompt-Driven Engineering) is Fernando García Varela's methodology for building software with the help of artificial intelligence in a governed way. It is not a collection of instructions for requesting code: it is a way of working that is sequential, blocking and auditable. The illustration sums it up in three pillars: think (the design is explicit before a single line of code is written), validate (an independent review) and execute (the AI writes the code without taking design decisions).

The problem. AI generates plausible code at great speed. Without structure, that speed produces decisions that nobody has taken consciously, scarce tests, vulnerabilities that nobody sees and technical debt. Example: a team requests code until the demonstration works; in production it fails and nobody knows why the system is built the way it is. Why it matters: the problem is not AI, it is the lack of method, and AI amplifies it.

The phases. The main cycle has eleven phases, from 0 to 10: context and objective; plan, with no code; plan review; code primer; test strategy; implementation; test implementation; test review; code review; minimal fixes; and version management, which is the handover to operations. No phase starts until the previous one has left its artefact and a person has validated it. Why it matters: an architecture error detected in the plan review costs a conversation; detected in production, it costs an incident.

The roles. A person orchestrates: sets the objective, validates each output and has the final say. The AI planner designs and does not write code. The AI builder writes the code and does not decide the design. The AI reviewer, different from the one that builds, evaluates and does not fix. The AI fixer applies minimal fixes. It is the segregation of duties of any serious process: whoever does the work does not approve it. The AI reviewer is an AI: it does not replace a person's review.

The verdicts. Each review ends with GO (work advances), GO with changes (work is adjusted and reviewed again) or NO-GO (work goes back). A GO with serious open problems is a self-approval and is discarded. In addition, a competent person reviews the code before it is merged; for sensitive code (payments, personal data, action limits of agents), a second person does. The validation policy adds a rule: a response that skips

its phase is discarded in its entirety, even if the code works. A valid NO-GO is worth more than an invalid GO.

Contexts and records. Before working, two contexts are loaded: the global one, with the company's rules (for example, no real personal data is sent to an external AI provider), and the project one. Each piece of work has a topic, for example pdf_export, that groups all its artefacts. Each artefact records which model produced it and which person validated it. Why it matters: months later, anyone can reconstruct why something was built that way, and an audit finds the evidence in a single place.

Reduced version. For limited changes, with no personal data or money, the reduced version (SPAD Lite) groups the phases into three artefacts: reviewed plan, reviewed delivery and version. The separation between the AI that builds and the one that reviews is not cut, nor is human validation. If the software contains AI, SPAD adds an evaluation of the system's behaviour.

What it does for your company. In the author's position, SPAD has two uses. First, building your own software with AI with method: phases, reviewed artefacts, independent review and human validation. Second, as a client: a company that knows how software is built with AI can require its suppliers and integrators to pass the productivity gain on to it, as lower cost or more scope for the same price. It can ask them for what SPAD already defines: the artefact of each phase, the record of the model and of human validation, the verdict of each review and the process metrics. Why it matters: AI changes what it costs to build software; if the company does not know how software is built, it cannot know whether that change reaches its invoice or stays with the supplier. SPAD sets no percentage: it lets you negotiate with data.

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.

What SPAD does not do: it promises no improvement figures (each organisation measures its own with the process metrics of document 08) and it certifies nothing; the self-assessment of conformity is carried out by the organisation itself.

Question for the classroom: if your team uses AI to write code, who decides the design today and who reviews what the AI writes?

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

SPAD IN 1 SLIDE

- **Human code review**: In SPAD, code generated by AI is always reviewed by a person before it is merged, with a second review for sensitive code, autonomous agents and critical production. AI helps to build, but it does not approve itself. The link opens the rule in the operating guide. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html#3-1-human-code-review
- **SPAD contexts**: Rule documents the AI must respect in each phase: the global context (organisation rules, data, approved models) and the project context. Exceptions are assessed by the reviewer AI and approved by a person. They ensure the AI does not work on its own assumptions. The link opens the contexts. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#contexts
- **SPAD topics**: A topic is SPAD's unit of work: a feature or a change that goes through the full cycle with its own artefact folder. It lets you follow from start to finish what was asked, planned and built, and who validated it. The link opens its definition. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#topics
- **Artefact records**: Each SPAD artefact carries three records: the model record (supplier, model, version, instruction and contexts used), the human validation record (who reviewed it and when) and the violation log (rules not complied with). They make the process auditable. The link opens the records. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html#4-1-model-record
- **Process metrics**: Indicators measured by a team using SPAD, such as how many reviews end in NO-GO, how many defects reach production or how long a topic takes. SPAD promises no figures: it proposes measuring them and setting your own targets. The link opens the metrics. https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html#process-metrics

SOURCES AND LINKS

- **SPAD 01 · Operating guide (phases, roles, verdicts, reduced version)** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.html
- **SPAD 02 · Contexts, topics and artefact records** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/02_SPAD_Contextos_temas_y_registro.html
- **SPAD 03 · Validation policy** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/03_SPAD_Politica_de_validacion.html
- **SEVEN-G 53 · Building solutions with AI (the only bridge)** https://seachad.github.io/AI_CONSULTING/SEVEN-G/html/en/53_SEVEN-G_Construccion_de_soluciones_con_IA.html
- **Operating guide: phases and roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/01_SPAD_Guia_operativa.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
- **Process metrics** https://seachad.github.io/AI_CONSULTING/SPAD/html/en/08_SPAD_Autoevaluacion_y_metricas.html

LEGAL NOTICE

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

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