SPAD EN 1 SLIDE · STRUCTURED PROMPT-DRIVEN ENGINEERING

Construir software con IA, con control

Pensar, validar, ejecutar: un método para que el código que escribe la IA se pueda revisar, explicar y auditar.

El problema →
La IA escribe código muy rápido. Sin método, nadie decide el diseño, faltan pruebas y no quedan evidencias. La velocidad se vuelve deuda y riesgo.

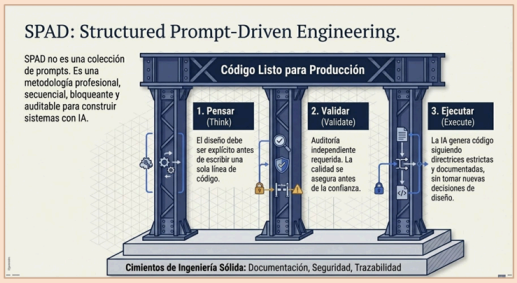
Once fases que bloquean →
Las fases van del contexto al plan, las pruebas, el código y la versión. Sin validación, no se avanza. Trabajo pequeño: SPAD Lite.

Cada IA, un solo papel →
Una IA planifica, otra construye y otra distinta revisa: son roles separados. Ninguna IA se aprueba a sí misma.

GO · GO con cambios · NO-GO →
Cada revisión da un veredicto. Lo acepta una persona, que además hace la revisión humana del código.

Todo queda escrito →
Los contextos fijan las reglas; un tema por trabajo; registros por artefacto: qué modelo y quién validó.

Para su compañía: dos usos →
Construir su software con IA con método. Y exigir a sus proveedores fases, registros y métricas del proceso: productividad en coste o alcance.



SPAD no promete cifras ni certifica: deja evidencias para que su compañía mida y audite lo que la IA construye.

→ Qué es SPAD | → Guía operativa: fases y roles | → Para qué le sirve a una compañía | → Métricas del proceso

SPAD · Versión 0.x, en revisión · CC BY 4.0 · Fernando García Varela seachad.github.io/AI_CONSULTING/ 1 / 1

Notas

SPAD (Structured Prompt-Driven Engineering, ingeniería estructurada guiada por instrucciones) es la metodología de Fernando García Varela para construir software con ayuda de la inteligencia artificial de forma gobernada. No es una colección de instrucciones para pedir código: es una forma de trabajar secuencial, bloqueante y auditable. La ilustración lo resume en tres pilares: pensar (el diseño es explícito antes de escribir una línea de código), validar (una revisión independiente) y ejecutar (la IA escribe el código sin tomar decisiones de diseño).

El problema. La IA genera código plausible a gran velocidad. Sin estructura, esa velocidad produce decisiones que nadie ha tomado de forma consciente, pruebas escasas, vulnerabilidades que nadie ve y deuda técnica. Ejemplo: un equipo pide código hasta que la demostración funciona; en producción falla y nadie sabe por qué el sistema está construido así. Por qué importa: el problema no es la IA, es la falta de método, y la IA lo amplifica.

Las fases. El ciclo principal tiene once fases, de la 0 a la 10: contexto y objetivo; plan, sin código; revisión del plan; guía de código; estrategia de pruebas; implementación; implementación de pruebas; revisión de pruebas; revisión del código; correcciones mínimas; y gestión de versiones, que es la entrega a operación. Ninguna fase empieza hasta que la anterior ha dejado su artefacto y una persona lo ha validado. Por qué importa: un error de arquitectura detectado en la revisión del plan cuesta una conversación; detectado en producción cuesta un incidente.

Los roles. Una persona orquesta: fija el objetivo, valida cada salida y tiene la última palabra. La IA planificadora diseña y no escribe código. La IA constructora escribe el código y no decide el diseño. La IA revisora, distinta de la que construye, evalúa y no corrige. La IA correctora aplica correcciones mínimas. Es la separación de funciones de cualquier proceso serio: quien hace no se aprueba a sí mismo. La IA revisora es una IA: no sustituye a la revisión de una persona.

Los veredictos. Cada revisión termina con GO (se avanza), GO con cambios (se ajusta y se revisa de nuevo) o NO-GO (se vuelve atrás). Un GO con problemas graves abiertos es una autoaprobación y se descarta.

Además, una persona competente revisa el código antes de incorporarlo; en código sensible (pagos, datos personales, límites de actuación de agentes), una segunda persona. La política de validación añade una regla: una respuesta que se salta su fase se descarta entera, aunque el código funcione. Un NO-GO válido vale más que un GO inválido.

Contextos y registros. Antes de trabajar se cargan dos contextos: el global, con las reglas de la compañía (por ejemplo, ningún dato personal real se envía a un proveedor externo de IA), y el de proyecto. Cada trabajo tiene un tema, por ejemplo pdf_export, que agrupa todos sus artefactos. Cada artefacto registra qué modelo lo produjo y qué persona lo validó. Por qué importa: meses después, cualquiera puede reconstruir por qué algo se construyó así, y una auditoría encuentra las evidencias en un solo lugar.

Versión reducida. Para cambios acotados, sin datos personales ni dinero, la versión reducida (SPAD Lite) agrupa las fases en tres artefactos: plan revisado, entrega revisada y versión. No se recorta la separación entre la IA que construye y la que revisa, ni la validación humana. Si el software lleva IA dentro, SPAD añade una evaluación del comportamiento del sistema.

Para qué le sirve a su compañía. Según la posición del autor, SPAD tiene dos usos. Primero, construir su propio software con IA con método: fases, artefactos revisados, revisión independiente y validación humana. Segundo, como cliente: una compañía que sabe cómo se construye software con IA puede exigir a sus proveedores e integradores que la ganancia de productividad llegue a ella, en menos coste o en más alcance por el mismo precio. Puede pedirles lo que SPAD ya define: el artefacto de cada fase, el registro del modelo y de la validación humana, los veredictos de cada revisión y las métricas del proceso. Por qué importa: la IA cambia lo que cuesta construir software; si la compañía no sabe cómo se construye, no puede saber si ese cambio llega a su factura o se queda en el proveedor. SPAD no fija un porcentaje: permite negociar con datos.

Un marco aparte. SPAD trata solo de cómo se construye el software; no es gobierno de la IA de la compañía. Quien quiera gobernar qué se construye, con qué riesgo y cómo se mide su valor tiene un marco aparte, SEVEN-G. El único puente es su documento 53, y los veredictos de SPAD no sustituyen a sus puertas de decisión.

Lo que SPAD no hace: no promete cifras de mejora (cada organización mide las suyas con las métricas del proceso del documento 08) y no certifica; la autoevaluación de conformidad la hace la propia organización.

Pregunta para el aula: si su equipo usa IA para programar, ¿quién decide hoy el diseño y quién revisa lo que la IA escribe?

TÉRMINOS DE ESTA SLIDE

- **Ciclo principal de SPAD** (fases 0–10): Las once fases de SPAD: contexto, plan, revisión del plan, guía de código, estrategia de pruebas, implementación, pruebas, revisión de pruebas, revisión del código, correcciones y gestión de versiones. Cada una deja un artefacto que se revisa. El enlace abre el ciclo en la guía operativa. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html#ciclo-principal-fases-0-a-10
- **SPAD Lite**: Versión reducida de SPAD para cambios pequeños y de bajo riesgo, con solo tres artefactos. Mantiene la revisión y la validación humana, pero con menos pasos. Sirve para no cargar un cambio menor con todo el ciclo. El enlace abre la versión reducida en la guía operativa. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html#version-reducida-spad-lite
- **Roles de SPAD**: Los papeles que intervienen en SPAD: el orquestador (una persona que dirige el ciclo y valida) y varias IA con funciones separadas: planificadora, revisora, constructora, correctora, analista y de diagnóstico. Quien construye no revisa lo que construye. El enlace abre el documento de presentación de SPAD, que describe cada rol. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/00_SPAD_Que_es_y_para_que_sirve.html
- **Veredictos de SPAD** (GO / NO-GO): Toda revisión de SPAD termina con uno de tres veredictos: GO (se continúa), GO con cambios (se continúa tras corregir lo indicado) o NO-GO (vuelve a la fase que corresponda). Un

solo vocabulario en todas las revisiones evita interpretaciones. El enlace abre los veredictos en la guía operativa. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html#veredictos

- **Revisión humana del código**: En SPAD, el código que genera la IA siempre lo revisa una persona antes de integrarlo, y hay una segunda revisión en código sensible, agentes con autonomía y producción crítica. La IA ayuda a construir, pero no se aprueba a sí misma. El enlace abre la regla en la guía operativa. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html#3-1-revision-humana-del-codigo
- **Contextos de SPAD**: Documentos de reglas que la IA debe respetar en cada fase: el contexto global (normas de la organización, datos, modelos aprobados) y el contexto de proyecto. Las excepciones las evalúa la IA revisora y las aprueba una persona. Sirven para que la IA no trabaje con supuestos propios. El enlace abre los contextos. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/02_SPAD_Contextos_temas_y_registro.html#contextos
- **Temas de SPAD**: Un tema es la unidad de trabajo de SPAD: una funcionalidad o un cambio que recorre el ciclo completo con su propia carpeta de artefactos. Permite seguir de principio a fin qué se pidió, qué se planificó, qué se construyó y quién lo validó. El enlace abre su definición. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/02_SPAD_Contextos_temas_y_registro.html#temas
- **Registros por artefacto**: Cada artefacto de SPAD lleva tres registros: del modelo (proveedor, modelo, versión, instrucción y contextos usados), de validación humana (quién lo revisó y cuándo) y de infracciones (reglas que no se cumplieron). Hacen el proceso auditable. El enlace abre los registros. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/02_SPAD_Contextos_temas_y_registro.html#4-1-registro-del-modelo
- **Métricas del proceso**: Indicadores que mide un equipo que usa SPAD, como cuántas revisiones terminan en NO-GO, cuántos defectos llegan a producción o cuánto tarda un tema. SPAD no promete cifras: propone medirlas y fijar objetivos propios. El enlace abre las métricas. https://seachad.github.io/AI_CONSULTING/SPAD/html/es/08_SPAD_Autoevaluacion_y_metricas.html#metricas-del-proceso

FUENTES Y ENLACES

- **SPAD 01 · Guía operativa (fases, roles, veredictos, versión reducida)** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html
- **SPAD 02 · Contextos, temas y registro de artefactos** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/02_SPAD_Contextos_temas_y_registro.html
- **SPAD 03 · Política de validación** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/03_SPAD_Politica_de_validacion.html
- **SEVEN-G 53 · Construcción de soluciones con IA (único puente)** https://seachad.github.io/AI_CONSULTING/SEVEN-G/html/es/53_SEVEN-G_Construccion_de_soluciones_con_IA.html
- **Guía operativa: fases y roles** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/01_SPAD_Guia_operativa.html
- **Para qué le sirve a una compañía** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/00_SPAD_Que_es_y_para_que_sirve.html#para-que-le-sirve-a-una-compania
- **Métricas del proceso** https://seachad.github.io/AI_CONSULTING/SPAD/html/es/08_SPAD_Autoevaluacion_y_metricas.html

AVISO LEGAL

Aviso legal: ayuda metodológica genérica y gratuita, «tal cual» y con fines informativos; no es asesoramiento jurídico ni está revisada jurídicamente para ningún caso. Cada organización la adapta y es responsable de su revisión legal y de su cumplimiento regulatorio; puede no recoger cambios normativos recientes. En la máxima medida permitida por la ley, el autor no asume responsabilidad alguna por los efectos de su aplicación. No otorga certificación de ningún tipo. Detalle en el documento 93.

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