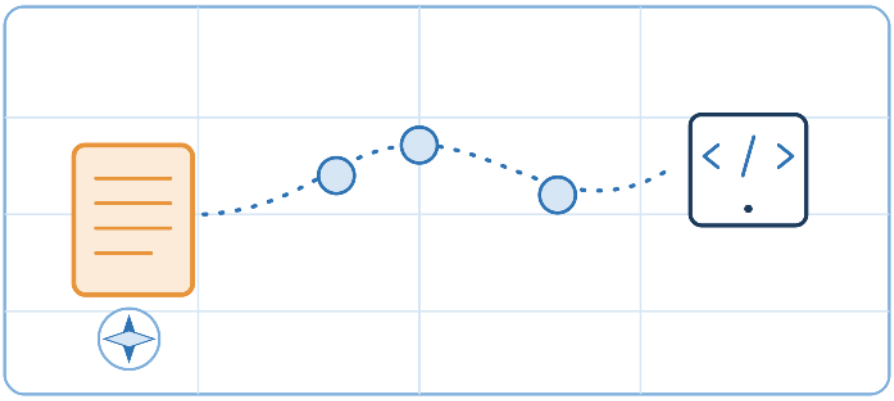
Scrum Manager
Uncovering better ways

State of the art

Spec Driven Development

en equipos ágiles

Mapa de referencia del conocimiento profesional vigente



Actualizado a junio de 2026

Scrum Manager® · Skill Arena

Sobre este documento

Este documento es un mapa curado de los conocimientos, prácticas y modelos empleados profesionalmente a la fecha de su publicación para dirigir el desarrollo de software con IA mediante especificaciones (Spec Driven Development) en equipos ágiles. Para cada uno, este mapa lo sitúa, indica su grado de adopción en la práctica profesional actual y orienta sobre dónde encontrar información de referencia.

Cómo usar este documento

Úsalo como observatorio y punto de referencia para contrastar si el conocimiento profesional que empleas está alineado con la práctica actual y en vanguardia.

Este mapa se complementa con tres apoyos:

- **Referencias para localizar la información.** Cada entrada incluye enlaces a fuentes donde ampliar el conocimiento. Son enlaces disponibles y verificados a la fecha de este documento.
- **Un documento de desarrollo.** Un texto paralelo desarrolla en profundidad los temas que aquí solo se enuncian. Está disponible en Spec Driven Development en equipos ágiles.
- **Una plataforma de entrenamiento y evaluación en Skill Arena.** En Spec Driven Development en equipos ágiles puedes contrastar tu nivel de conocimiento y, si lo superas, obtener un diploma que acredita curricularmente la solvencia y vanguardia profesional en esta área.

Estado del conocimiento

El conocimiento sobre Spec Driven Development evoluciona de forma extremadamente rápida: método, herramientas y prácticas se renuevan en cuestión de meses, y aunque algunos fundamentos ya están asentados, buena parte del ecosistema sigue en plena formación. En los distintos apartados del documento, las etiquetas (ESTABLECIDO, EN CONSOLIDACIÓN, EMERGENTE) ayudan a identificar la madurez de cada concepto:

- **ESTABLECIDO** consenso asentado; conocimiento que se da por necesario.
- **EN CONSOLIDACIÓN** gana adopción con rapidez; aún no universal pero ya relevante.
- **EMERGENTE** frontera reciente; alta relevancia y alta volatilidad.

Bloque A — El paradigma: por qué SDD

Qué es SDD, qué problema resuelve y por qué no es un retroceso a waterfall. La base conceptual común a cualquier rol del equipo.

Del vibe coding al desarrollo dirigido por especificación · ESTABLECIDO

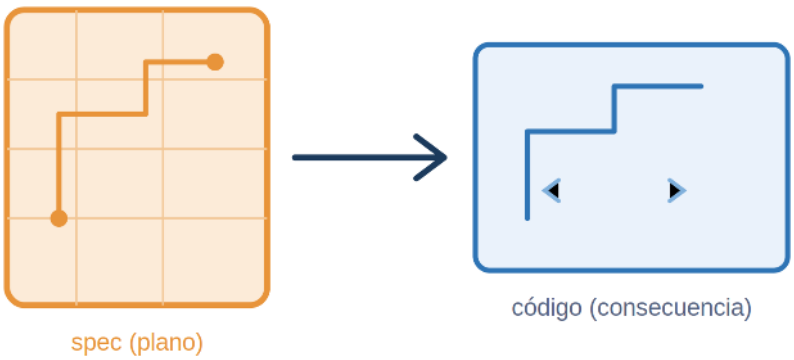
El vibe coding —pedir código a la IA de forma conversacional y aceptarlo si funciona— es productivo para prototipos pero tiene un techo: a partir de cierta escala genera requisitos implícitos, contaminación de contexto y deriva arquitectónica. SDD responde escribiendo una especificación que actúa como contrato antes de generar código.

Por qué está aquí ahora. El vibe coding se ha consolidado como práctica y, con él, sus límites: mediciones del sector muestran que la sensación de velocidad no corresponde a la productividad real y que una proporción notable del código generado introduce vulnerabilidades. SDD es la respuesta metodológica que ha ganado tracción frente a esos límites.

Dónde mirar. Martin Fowler — Understanding Spec-Driven Development · Addy Osmani — How to write a good spec for AI agents

La spec como artefacto primario del desarrollo · ESTABLECIDO

El centro del trabajo se desplaza de escribir código a especificar con claridad qué se construye, por qué, bajo qué restricciones y con qué criterios de éxito. La spec es el plano; el código es la consecuencia. Para un agente literal, cada decisión no especificada es una decisión que toma por su cuenta.



La spec es el plano; el código es la consecuencia que se deriva de ella.

Por qué está aquí ahora. Es el cambio de identidad profesional que vertebra SDD: «la IA no es el cuello de botella; tu spec lo es». Afecta a todos los roles, no solo a quien programa, porque la especificación precisa es competencia compartida del equipo.

Dónde mirar. Addy Osmani — How to write a good spec for AI agents · Martin Fowler — Exploring Gen AI

SDD y agilidad: no es el regreso de waterfall · EN CONSOLIDACIÓN

SDD usa fases secuenciales y puertas de aprobación, lo que recuerda a waterfall, pero difiere en tres ejes: el alcance (cada incremento, no el proyecto entero), el feedback (en cada puerta, no al final) y la reversibilidad (modificar un documento es trivial frente a reescribir código). Es la aplicación coherente de principios ágiles cuando parte del equipo son agentes.

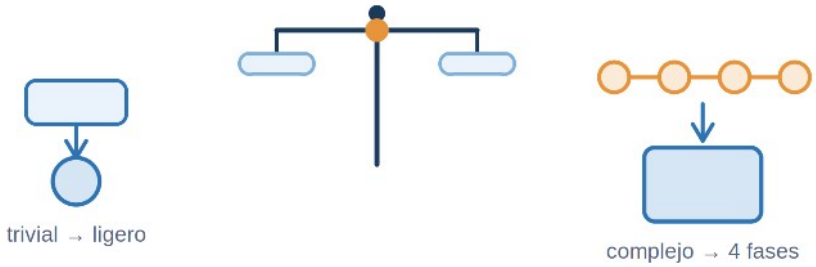
Por qué está aquí ahora. Es la objeción que cualquier profesional ágil plantea ante SDD, y resolverla es lo que permite adoptarlo sin traicionar la agilidad. Thoughtworks lo sitúa en «Assess»: aporta

valor con requisitos ambiguos, equipos distribuidos o necesidad de trazabilidad, y puede no compensar en problemas triviales.

Dónde mirar. Thoughtworks Technology Radar — SDD · Manifiesto Ágil

El principio de proporcionalidad · ESTABLECIDO

Ajustar la intensidad del proceso al tamaño del problema. Un cambio trivial no necesita una spec de cuatro fases; una funcionalidad compleja sí. SDD mal calibrado se convierte en una burocracia de especificación que ralentiza en lugar de acelerar.



Se ajusta la intensidad del proceso al tamaño del problema: ni un mazo para una nuez, ni al revés.

Por qué está aquí ahora. Es el criterio que separa el uso útil de SDD de su caricatura burocrática. La propia Böckeler documentó cómo una herramienta convirtió un bug menor en cuatro historias con dieciséis criterios: «usar un mazo para romper una nuez». Saber cuándo NO aplicar SDD es parte de dominarlo.

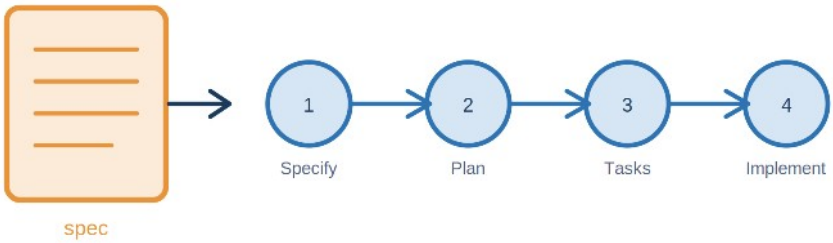
Dónde mirar. Martin Fowler — SDD tools · Scrum Manager — Scrum en la era de la IA

Bloque B — El método: fases y puertas

La mecánica de SDD: el flujo de cuatro fases y las puertas de aprobación que lo hacen viable. Es el núcleo operativo del tema.

El flujo de cuatro fases (Requisitos → Diseño → Tareas → Implementación) · ESTABLECIDO

Todas las implementaciones de SDD convergen en la misma secuencia: qué se construye (requisitos), cómo (diseño), en qué unidades atómicas (tareas) y la ejecución (implementación). Cada fase produce un documento que se revisa antes de avanzar. Kiro y GitHub Spec Kit usan nombres distintos para la misma estructura.



El flujo converge en cuatro fases: requisitos, diseño, tareas e implementación.

Por qué está aquí ahora. Es la columna vertebral del método, estable y compartida por todas las herramientas. Refleja la secuencia lógica mínima para pasar de una intención a código cuando el constructor es un agente.

Dónde mirar. GitHub Spec Kit · AWS Kiro

Las puertas de aprobación (approval gates) · ESTABLECIDO

Los puntos donde el humano ejerce criterio entre fases: ¿los requisitos son correctos?, ¿el diseño es viable?, ¿las tareas cubren el diseño? Concentran la revisión donde la información es más valiosa y el coste de corrección menor, en lugar de supervisar cada acción del agente. Revisar en las puertas, no durante la implementación.

El criterio humano se concentra en las puertas entre fases, donde corregir cuesta menos.

Por qué está aquí ahora. Son lo que distingue SDD de «waterfall con documentos cortos» y lo que combate la fatiga de aprobación. El coste de corregir un error crece de forma acelerada por fase, así que detectarlo en la puerta correcta es el mecanismo más rentable del proceso.

Dónde mirar. Martin Fowler — Exploring Gen AI · GitHub Spec Kit — documentación

Tareas atómicas, oleadas y commits atómicos · EN CONSOLIDACIÓN

El diseño se descompone en tareas pequeñas (uno a tres ficheros), agrupadas en oleadas según sus dependencias: las tareas de una misma oleada se ejecutan en paralelo por subagentes con contexto limpio; las oleadas, en secuencia. Cada tarea produce un commit independiente, lo que da reversibilidad granular y trazabilidad.

Por qué está aquí ahora. Es la práctica que convierte un cambio grande en una secuencia manejable y aprovecha el paralelismo de los agentes sin contaminación de contexto. La granularidad de commit por tarea conecta la intención de negocio con cada línea de código.

Dónde mirar. GitHub Spec Kit · Anthropic — Building effective agents

El Impact Report en codebase existente (brownfield) · EN CONSOLIDACIÓN

En proyectos con código previo —la situación habitual— la primera fase no empieza escribiendo requisitos, sino analizando el código actual: qué ficheros se verán afectados, qué patrones existen que respetar y qué efectos colaterales puede haber. Evita que la spec pida algo que duplica o contradice lo existente.

Por qué está aquí ahora. La mayor parte de la literatura asume proyectos nuevos (greenfield), pero el trabajo real es sobre código heredado, donde un agente que ignora el contexto causa más daño. Es donde SDD aporta más valor y donde el análisis previo es imprescindible.

Dónde mirar. Martin Fowler — Exploring Gen AI · GitHub Spec Kit

Bloque C — Escribir buenas specs

La competencia central de SDD: redactar especificaciones que un agente pueda ejecutar con fidelidad. Aquí está el mayor valor diferencial del profesional.

La spec inteligente, no extensa (principios de Osmani) · ESTABLECIDO

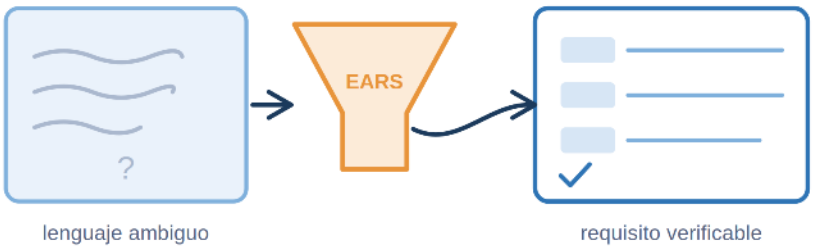
Una buena spec cubre lo justo para guiar al agente sin abrumarlo: parte de una visión de alto nivel que el agente elabora, se estructura como documento profesional, se divide en prompts modulares, incorpora autoverificación y conocimiento experto, y se trata como documento vivo. Minimal no significa corto.

Por qué está aquí ahora. Es la referencia más citada sobre cómo escribir specs para agentes. La calidad de la spec determina la calidad del resultado más que cualquier otra variable; los cinco principios son el marco práctico de calibración.

Dónde mirar. Addy Osmani — How to write a good spec for AI agents · AGENTS.md

La notación EARS para criterios de aceptación · EN CONSOLIDACIÓN

Easy Approach to Requirements Syntax: un conjunto de patrones (ubicuo, dirigido por evento, por estado, comportamiento no deseado, opcional) que restringen el lenguaje natural lo justo para eliminar ambigüedad sin sacrificar legibilidad. Los agentes responden bien a requisitos estructurados con patrones consistentes.



EARS estructura el lenguaje natural lo justo para eliminar la ambigüedad sin perder legibilidad.

Por qué está aquí ahora. Adoptada por varias herramientas SDD y por organizaciones de ingeniería exigente. Reduce la malinterpretación del agente al estructurar el «qué debe hacer el sistema» de forma verificable, que es el corazón de la fase de requisitos.

Dónde mirar. Alistair Mavin — EARS · GitHub Spec Kit

El sistema de boundaries: Always / Ask First / Never · EN CONSOLIDACIÓN

Un marco de delegación de tres niveles en lugar de una lista plana de reglas: qué puede hacer el agente sin preguntar, qué requiere aprobación por su impacto y qué está absolutamente prohibido. Opera a nivel de proyecto (constitución, tipo CLAUDE.md o AGENTS.md) y a nivel de tarea. Es autonomía gradual y revisable.

La delegación se gradúa en tres niveles: Always, Ask First y Never.

Por qué está aquí ahora. El análisis de miles de configuraciones de agentes muestra que las specs más efectivas usan este sistema, no listas planas. Funciona como la delegación a un profesional competente y es la base de una autonomía que el equipo puede ampliar a medida que gana confianza.

Dónde mirar. Addy Osmani — How to write a good spec for AI agents · AGENTS.md

La maldición de las instrucciones (instruction overload) · EN CONSOLIDACIÓN

A medida que se acumulan instrucciones en un prompt, la probabilidad de que el modelo cumpla cada una cae de forma predecible. La respuesta no es escribir menos, sino dividir mejor: prompts modulares, instrucciones prioritarias primero, y separación de niveles. Es el fundamento técnico de la descomposición en tareas.

Por qué está aquí ahora. Es un hallazgo de investigación que explica por qué las specs sobrecargadas fallan y por qué la fase de tareas funciona. Conocerlo cambia la intuición natural de «añadir más detalle para cubrir más casos», que tiene rendimientos decrecientes y luego negativos.

Dónde mirar. Curse of Instructions (paper) · Anthropic — Effective context engineering

Specs vivas, deriva y la Clarity Gate · EN CONSOLIDACIÓN

Mantener la spec alineada con el código que evoluciona. La taxonomía de Böckeler distingue spec-first (efímera), spec-anchored (documentación viva) y spec-as-source (la spec es la fuente, el código se regenera). La Clarity Gate es la prueba de calidad: ¿puede otro agente generar código equivalente solo con la spec?

Por qué está aquí ahora. La deriva entre spec y código es el riesgo principal de SDD a medio plazo: una spec desactualizada es peor que no tenerla. Saber en qué nivel de vida opera el equipo y aplicar la Clarity Gate es lo que sostiene el valor de la spec en el tiempo.

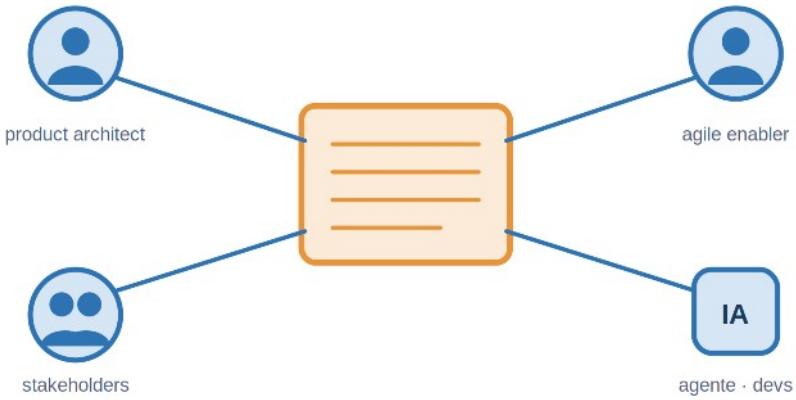
Dónde mirar. Martin Fowler — SDD tools · Simon Willison — weblog

Bloque D — SDD en el equipo y su ecosistema

Cómo encaja SDD en el equipo ágil —incluidos los roles no técnicos— y el estado de las herramientas que lo soportan. La zona de mayor movimiento del tema.

Encaje en roles y ceremonias ágiles · EN CONSOLIDACIÓN

SDD redistribuye el trabajo del equipo: el product owner / product architect gana exigencia de precisión en los requisitos y aprueba la primera puerta; el agile enabler facilita las puertas sin que se vuelvan cuellos de botella; los stakeholders ganan visibilidad porque la spec es legible por no técnicos. El Sprint pasa a contener specs por crear y specs aprobadas listas para implementar.



La spec, legible por no técnicos, redistribuye el trabajo y abre la colaboración del equipo.

Por qué está aquí ahora. Es la dimensión que conecta SDD con la organización del equipo y la que justifica que no sea un asunto solo de desarrolladores. La spec abre a los roles no técnicos una caja que el código mantenía cerrada, y eso cambia la colaboración.

Dónde mirar. Scrum Manager — Scrum en la era de la IA · Scrum.org — El Scrum Master y la IA

Antipatrones: teatro de especificación y documentación zombi · EN CONSOLIDACIÓN

Los modos de fallo característicos de SDD mal aplicado: sobreespecificación (más tiempo en la spec que en el software), puertas como cuellos de botella, teatro de especificación (specs que nadie revisa de verdad) y documentación zombi (specs que nadie consulta ni actualiza). El agile enabler vigila estos riesgos.

Por qué está aquí ahora. Reconocer los antipatrones es tan parte del dominio como conocer el método. Son los puntos donde SDD deja de aportar valor y empieza a estorbar, y aparecen con facilidad si no se calibra con el principio de proporcionalidad.

Dónde mirar. Marmelab — SDD: The Waterfall Strikes Back · Martin Fowler — SDD tools

El ecosistema de herramientas (Spec Kit, Kiro, y más) · EMERGENTE · *nuevo desde 2026*

El instrumental crece deprisa. GitHub Spec Kit (CLI open source, soporta 30+ agentes de codificación) es la opción más adoptada por la comunidad; AWS Kiro ofrece un IDE dedicado a SDD; Tessl explora el nivel spec-as-source; y han aparecido alternativas como BMAD o GSD. Conviene no confundir el método con el instrumento: SDD es la metodología, no la herramienta.

Por qué está aquí ahora. Es la zona más volátil del tema: las versiones y la lista de herramientas cambian de mes en mes (Spec Kit ha superado las 90.000 estrellas y publica versiones casi semanales). Conocer el panorama orienta la elección, pero la herramienta concreta caducará antes que el método.

Dónde mirar. GitHub Spec Kit — releases · AWS Kiro

Qué vigilar en la próxima revisión

Señales de cambio que el equipo editorial anticipa para la siguiente edición de este mapa:

- La evolución del nivel spec-as-source: si pasa de experimental a adopción real, con herramientas que lo soporten de forma fiable.
- La consolidación o reordenación del ecosistema de herramientas (Spec Kit, Kiro, Tessl, BMAD, GSD y nuevas entradas).
- La maduración de SDD como norma de industria y su posible salto de «Assess» a «Trial» en los radares tecnológicos.
- La integración de SDD con los estándares del ecosistema agéntico (AGENTS.md, MCP) y con las prácticas de supervisión humana exigidas por regulación.

Nota sobre las referencias

Los enlaces incluidos en este documento estaban disponibles y verificados en la fecha de su actualización. Dado el ritmo de cambio del área, algunos pueden modificarse o dejar de estar accesibles con el tiempo; cada revisión del mapa actualiza también sus referencias.