COMPETITIVE ANALYSIS

SDD Competitive Analysis

Ed Barlow www.webcloudstudio.com August 5, 2026

Specification Driven Desing Competitive Analysis

I asked Claude to honestly rate competitive Specification Driven Development Methedologies in categories and groups it created on a scale from 0-5.

There are two comparisons - the first group is a comparison against the common methedologies and the second section contains a comparison of all delivery methods it could find.

Key	Product	What it is	License	Runs on
DD	Drydock	Specification builder based on Agile and Test Driven Development	MIT	Your LLM subscription
SK	GitHub Spec Kit	Prompt templates turning an idea into spec, plan, and tasks	MIT	Your LLM subscription
KI	Kiro (AWS)	IDE with a spec-first mode and agent hooks	Proprietary	Metered inference
BM	BMAD-METHOD	Multi-agent role framework for elicitation and story delivery	MIT	Your LLM subscription
CM	LLM Native	One rules file and an agent	Free	Your LLM subscription

Summary

Category	Drydock	Spec Kit	Kiro	BMAD	LLM	Winner
Specification	2.8	1.8	2.4	3.0	1.0	BMAD
Planning	3.8	1.3	1.5	3.3	1.0	Drydock
Execution	3.5	1.2	1.7	3.2	1.2	Drydock
Verification	3.8	0.5	0.8	0.8	0.5	Drydock
Change Management	3.3	0.7	0.8	0.7	0.0	Drydock
Context and Cost	4.0	0.8	0.0	2.0	1.5	Drydock
Governance and Review	4.3	1.1	1.9	1.7	1.4	Drydock
Adoption	2.3	4.2	2.2	3.5	4.7	LLM Native
Overall	3.5	1.4	1.4	2.2	1.4	Drydock

1 Specification

#	Feature	DD	SK	KI	BM	CM	Winner
1	Authors a specification from scratch	0	4	4	5	2	BMAD
2	Ingests existing source material	5	1	2	3	2	Drydock
3	Typed specification files	5	3	3	3	1	Drydock
4	Specification quality audit	4	0	1	2	0	Drydock
5	Executable behavioral specs	0	1	2	2	0	Kiro / BMAD

BMAD wins because its analyst and product-manager agents produce a specification from nothing. Drydock does not author specifications.

2 Planning

#	Feature	DD	SK	KI	BM	CM	Winner
6	Dependency graph database	5	1	1	2	0	Drydock
7	Stable Build IDs enable rebuild	5	3	4	4	0	Drydock
8	Effort estimated as token cost	5	0	0	2	0	Drydock
9	Blocker gate halts planning	5	1	2	3	0	Drydock
10	Spike phase before planning	2	2	1	4	1	BMAD
11	Workflow right-sized to problem	1	1	1	5	5	BMAD / LLM Native

Drydock wins because its plan is a queryable dependency graph rather than an ordered task list. BMAD is second on the strength of adaptive, right-sized workflows.

3 Execution

#	Feature	DD	SK	KI	BM	CM	Winner
12	Executes the plan to working software	5	3	4	4	3	Drydock
13	Per-step context stacking	5	2	2	4	1	Drydock
14	Automatic repair loop	5	0	1	2	1	Drydock
15	Model escalation on retry	5	0	0	1	0	Drydock
16	Parallel execution	0	0	1	3	1	BMAD
17	Multi-agent role specialization	1	2	2	5	1	BMAD

Drydock wins on repair and resumption; it scores zero on parallel execution, which BMAD leads among these five.

4 Verification

#	Feature	DD	SK	KI	BM	CM	Winner
18	TDD RED to GREEN enforced	4	1	2	3	2	Drydock
19	Acceptance executed without an LLM	5	0	1	1	0	Drydock

#	Feature	DD	SK	KI	BM	CM	Winner
20	Tests hash-locked before build	● 0	● 0	● 0	● 0	● 0	Tie
21	Test-quality analysis	★ 5	● 0	● 0	● 0	● 0	Drydock
22	EARS criteria with stable IDs	★ 5	● 0	● 2	● 0	● 0	Drydock
23	Guardrails behind a hard gate	★ 5	● 2	● 0	● 0	● 2	Drydock
24	Release verdict with blockers	★ 5	● 0	● 1	● 2	● 0	Drydock
25	CI release gate with exit code	● 1	● 1	● 0	● 0	● 0	Drydock / Spec Kit

Drydock wins because acceptance criteria are executed rather than judged by the model. It does not lock test artifacts before implementation, which caps row 18 at 4.

5 Change Management

#	Feature	DD	SK	KI	BM	CM	Winner
26	Drift detection by content hash	★ 5	● 0	● 1	● 0	● 0	Drydock
27	Sealed specs need a change ticket	★ 5	● 1	● 0	● 0	● 0	Drydock
28	Replan scoped to impacted work	★ 5	● 1	● 2	● 2	● 0	Drydock
29	Delta change format with archive	● 2	● 1	● 1	● 1	● 0	Drydock
30	Semantic diff and impact warning	● 1	● 0	● 0	● 0	● 0	Drydock
31	Fix spec first, regenerate code	● 2	● 1	● 1	● 1	● 0	Drydock

Drydock wins because it is the only product here that detects specification drift and replans against it. Kiro is second and manual.

6 Context and Cost

#	Feature	DD	SK	KI	BM	CM	Winner
32	Specification compaction	★ 5	● 0	● 0	● 3	● 1	Drydock
33	Compact selection by build graph	★ 5	● 0	● 0	● 0	● 0	Drydock
34	No API-key path	★ 5	● 3	● 0	● 3	★ 5	Drydock / LLM Native
35	Model routing and cost tiering	● 1	● 0	● 0	● 2	● 0	BMAD

Drydock wins because context is compacted and priced per story. Kiro is last: it is the only product billed per token.

7 Governance and Review

#	Feature	DD	SK	KI	BM	CM	Winner
36	Injected rules and branding	★ 5	● 2	● 3	● 2	● 4	Drydock
37	Persistent intent injection	★ 5	● 4	● 3	● 3	● 4	Drydock
38	Build evidence per step	★ 5	● 1	● 2	● 2	● 0	Drydock
39	Dependency legitimacy guard	● 4	● 0	● 0	● 0	● 1	Drydock
40	Review console	★ 5	● 0	● 4	● 1	● 0	Drydock

#	Feature	DD	SK	KI	BM	CM	Winner
41	Documentation generation	★ 5	● 1	● 1	● 2	● 1	Drydock
42	Retrospectives and lessons capture	● 1	● 0	● 0	● 2	● 0	BMAD

Drydock wins on injected governance and per-step evidence. A rules file is second on rules injection and needs no tooling to get there.

8 Adoption

#	Feature	DD	SK	KI	BM	CM	Winner
43	Time to first working code	● 1	● 3	● 4	● 2	★ 5	LLM Native
44	Zero install	● 1	● 3	● 1	● 2	★ 5	LLM Native
45	Open source	★ 5	★ 5	● 1	★ 5	★ 5	Tie
46	Stack coverage out of the box	● 2	● 4	● 4	● 4	★ 5	LLM Native
47	Community and ecosystem	● 1	★ 5	● 3	● 4	★ 5	Spec Kit / LLM Native
48	Provider and IDE neutrality	● 4	★ 5	● 0	● 4	● 3	Spec Kit

A plain rules file wins on every adoption measure except licensing. Drydock is last: one stack shipped, no ecosystem, and four commands before code runs.

Which To Use

Situation	Use
Small specification, one-file fix, exploratory work	LLM Native
No specification written yet	BMAD-METHOD
Free, agent-neutral team convention	GitHub Spec Kit
IDE-centric team wanting spec-first editing	Kiro
Large specification	Drydock
Enterprise adoption under governance and audit	Drydock
Long-lived product whose specification keeps changing	Drydock
Parallel agent throughput	Spec Kitty or GSD

Full Analysis (All Tools)

The same forty-eight features across eleven systems. Maturity follows Böckeler's taxonomy: **spec-first** (the spec guides the first build, then drifts), **spec-anchored** (spec and code co-evolve under tests), **spec-as-source** (humans edit only specs).

Key	Product	License	Maturity
DD	Drydock	MIT	Spec-anchored
SK	GitHub Spec Kit	MIT	Spec-first
KI	Kiro (AWS)	Proprietary	Spec-first
TS	Tessl + intent-integrity-kit	Proprietary	Spec-as-source

Key	Product	License	Maturity
OS	OpenSpec	MIT	Spec-anchored
BM	BMAD-METHOD	MIT	Spec-first
KT	Spec Kitty	Open source	Spec-anchored
WA	Walden	Open source	Spec-anchored
SP	Superpowers	Open source	Spec-first
GS	GSD (Get Shit Done)	Open source	Spec-anchored
TR	Traycer	Proprietary	Spec-first

#	Feature	DD	SK	KI	TS	OS	BM	KT	WA	SP	GS	TR
1	Authors a specification from scratch	0	4	4	3	3	5	3	2	3	4	4
2	Ingests existing source material	5	1	2	2	4	3	2	1	1	2	3
3	Typed specification files	5	3	3	4	3	3	3	3	1	3	2
4	Specification quality audit	4	0	1	3	2	2	1	3	1	2	1
5	Executable behavioral specs	0	1	2	4	3	2	2	2	1	1	1
6	Dependency graph database	5	1	1	1	1	2	2	2	2	4	2
7	Stories with stable IDs enabling rebuild	5	3	4	2	3	4	4	3	2	4	3
8	Effort estimated as token cost	5	0	0	0	0	2	0	0	0	1	0
9	Blocker gate halts planning	5	1	2	1	2	3	2	3	3	3	2
10	Spike phase before planning	2	2	1	3	5	4	2	2	5	5	4
11	Workflow right-sized to problem	1	1	1	2	4	5	2	1	3	4	4
12	Executes the plan to working software	5	3	4	3	3	4	4	3	4	4	3
13	Per-step context stacking	5	2	2	2	2	4	2	1	3	4	3
14	Automatic repair loop	5	0	1	2	1	2	2	1	3	3	1
15	Model escalation on retry	5	0	0	1	0	1	1	0	2	5	3
16	Parallel execution	0	0	1	0	1	3	5	0	4	5	3
17	Multi-agent role specialization	1	2	2	2	1	5	4	1	4	4	3
18	TDD RED to GREEN enforced	4	1	2	3	1	3	2	3	5	3	1
19	Acceptance executed without an LLM	5	0	1	3	1	1	1	4	2	2	0

#	Feature	DD	SK	KI	TS	OS	BM	KT	WA	SP	GS	TR
20	Tests hash-locked before build	0	0	0	5	0	0	0	2	1	0	0
21	Test-quality analysis	5	0	0	5	0	0	0	1	2	2	0
22	EARS criteria with stable IDs	5	0	2	0	0	0	0	5	0	0	0
23	Guardrails behind a hard gate	5	2	0	2	0	0	1	3	0	2	0
24	Release verdict with blockers	5	0	1	2	1	2	2	4	1	2	1
25	CI release gate with exit code	1	1	0	2	2	0	2	5	0	2	0
26	Drift detection by content hash	5	0	1	4	2	0	1	5	0	2	1
27	Sealed specs need a change ticket	5	1	0	2	3	0	1	3	0	1	0
28	Replan scoped to impacted work	5	1	2	3	3	2	2	3	1	2	2
29	Delta change format with archive	2	1	1	2	5	1	2	2	0	2	1
30	Semantic diff and impact warning	1	0	0	5	3	0	1	4	0	2	1
31	Fix spec first, regenerate code	2	1	1	5	3	1	2	3	1	2	1
32	Specification compaction	5	0	0	2	1	3	0	0	1	2	1
33	Compact selection by build graph	5	0	0	0	0	0	0	0	0	1	0
34	No API-key path	5	3	0	0	3	3	3	4	4	3	0
35	Model routing and cost tiering	1	0	0	1	0	2	1	0	2	5	3
36	Injected rules and branding	5	2	3	1	1	2	1	2	1	1	1
37	Persistent intent injection	5	4	3	3	3	3	3	4	2	3	2
38	Build evidence per step	5	1	2	3	1	2	2	4	1	2	1
39	Dependency legitimacy guard	4	0	0	0	0	0	0	0	0	5	0
40	Review console	5	0	4	2	3	1	4	2	1	1	5
41	Documentation generation	5	1	1	2	1	2	1	1	0	1	1
42	Retrospectives and lessons capture	1	0	0	1	1	2	5	4	2	2	1
43	Time to first working code	1	3	4	2	3	2	2	3	3	2	4

#	Feature	DD	SK	KI	TS	OS	BM	KT	WA	SP	GS	TR
44	Zero install	1	3	1	1	3	2	2	3	4	2	1
45	Open source	5	5	1	1	5	5	5	5	5	5	1
46	Stack coverage out of the box	2	4	4	3	4	4	4	4	4	4	4
47	Community and ecosystem	1	5	3	2	3	4	2	2	3	2	2
48	Provider and IDE neutrality	4	5	0	2	4	4	5	5	2	2	3
	Overall	**3.5**	**1.4**	**1.4**	**2.2**	**2.0**	**2.2**	**2.0**	**2.5**	**1.9**	**2.6**	**1.7**

Ranked overall: Drydock 3.5, GSD 2.6, Walden 2.5, BMAD 2.2, Tessl 2.2, OpenSpec 2.0, Spec Kitty 2.0, Superpowers 1.9, Traycer 1.7, Spec Kit 1.4, Kiro 1.4.

Walden is the closest system to Drydock, reaching EARS criteria, drift detection, and proof-based completion independently, and taking the CI release gate outright. Tessl leads test integrity by locking tests before implementation. Spec Kitty, Superpowers, and GSD lead parallel execution. BMAD leads right-sizing and role specialization. GSD leads model routing and package legitimacy.

Legend

Mark	Score	Meaning
★	5	Best in field; deterministic and enforced
●	4	Strong first-class capability
●	3	Present; advisory rather than enforced
●	2	Partial or manual
●	1	Nominal or documentation-only
●	0	Absent

Method

Every product is scored on the same forty-eight features. The feature set is drawn from all eleven systems, so each product is measured on ground it wins as well as ground it loses. Scores are documentation-derived judgment on a 0 to 5 scale. No product was executed and no benchmark was run.

Sources

To reproduce this analysis: *provide a competitive analysis of the Specification Driven Development field using honest benchmarks of various features from the following documentation.*

- Web Cloud Studio — https://www.webcloudstudio.com

- Drydock Specification — https://webcloudstudio.github.io/Drydock/Drydock_Specification.pdf
- GitHub Spec Kit — https://github.com/github/spec-kit
- Kiro — https://kiro.dev/docs/
- BMAD-METHOD — https://docs.bmad-method.org/
- OpenSpec — https://github.com/Fission-AI/OpenSpec
- Spec Kitty — https://github.com/Priivacy-ai/spec-kitty
- Walden — https://andrearaponi.github.io/walden/
- Superpowers — https://github.com/obra/superpowers
- GSD — https://github.com/gsd-build/get-shit-done
- Traycer — https://docs.traycer.ai/quickstart
- Tessl intent-integrity-kit — https://tessl.io/registry/tessl-labs/intent-integrity-kit/2.7.5
- Böckeler, *Understanding Spec-Driven Development: Kiro, spec-kit, and Tessl* — https://martinfowler.com/articles/exploring-gen-ai/sdd-3-tools.html
- *Spec-Driven Development: From Code to Contract* — https://arxiv.org/html/2602.00180v1
- cameronsjo/spec-compare — https://github.com/cameronsjo/spec-compare
- *Skepticism Toward Specification-Driven Development* — https://zenn.dev/cbmrham/articles/202601-spec-driven-development-skepticism