Universidade Federal do Pará - UFPA

Desenvolvimento de Software Apoiado por IA — 2026.4

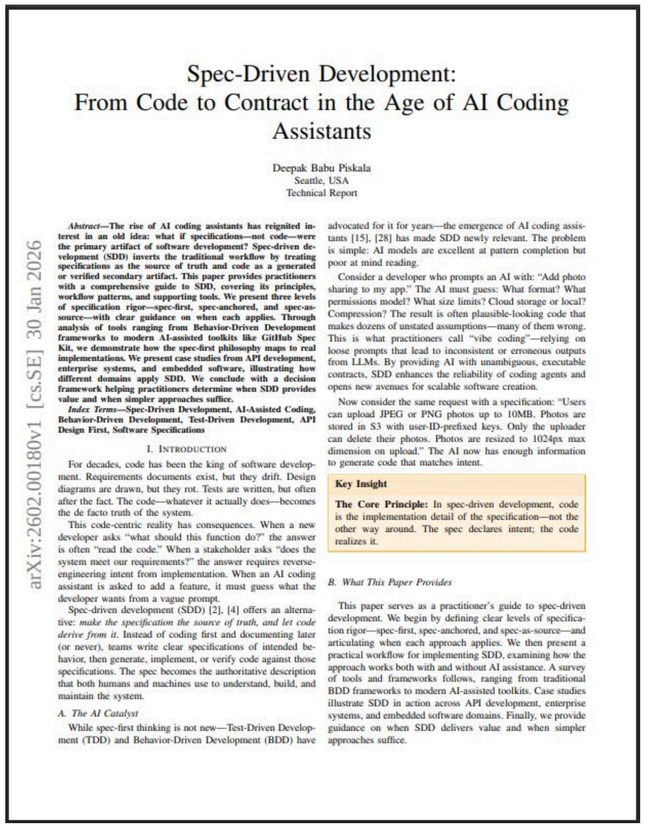
Specification-Driven Development, e a evidência que contraria a promessa

Aula 5

Dr. Gustavo pinto
Orientando: Leandro Veloso

Specification-Driven Development (SDD)

- Desenvolvimento guiado por especificações
- Especificação como fonte de verdade
- Intenção separada dos detalhes de implementação
- Redução de ambiguidades
- Rastreabilidade entre especificação e implementação
- Comunicação desenvolvedor–agente de IA



Spec-Driven Development:
From Code to Contract in the Age of AI Coding Assistants

Deepak Babu Piskala
Seattle, USA
Technical Report

***Abstract*—The rise of AI coding assistants has reignited interest in an old idea: what if specifications—not code—were the primary artifact of software development? Spec-driven development (SDD) inverts the traditional workflow by treating specifications as the source of truth and code as a generated or verified secondary artifact. This paper provides practitioners with a comprehensive guide to SDD, covering its principles, workflow patterns, and supporting tools. We present three levels of specification rigor—spec-first, spec-anchored, and spec-as-source—with clear guidance on when each applies. Through analysis of tools ranging from Behavior-Driven Development frameworks to modern AI-assisted toolkits like GitHub Spec Kit, we demonstrate how the spec-first philosophy maps to real implementations. We present case studies from API development, enterprise systems, and embedded software, illustrating how different domains apply SDD. We conclude with a decision framework helping practitioners determine when SDD provides value and when simpler approaches suffice.**

***Index Terms*—Spec-Driven Development, AI-Assisted Coding, Behavior-Driven Development, Test-Driven Development, API Design First, Software Specifications**

I. Introduction

For decades, code has been the king of software development. Requirements documents exist, but they drift. Design diagrams are drawn, but they rot. Tests are written, but often after the fact. The code—whatever it actually does—becomes the de facto truth of the system.

This code-centric reality has consequences. When a new developer asks "what should this function do?" the answer is often "read the code." When a stakeholder asks "does the system meet our requirements?" the answer requires reverse-engineering intent from implementation. When an AI coding assistant is asked to add a feature, it must guess what the developer wants from a vague prompt.

Spec-driven development (SDD) [2], [4] offers an alternative: *make the specification the source of truth, and let code derive from it.* Instead of coding first and documenting later (or never), teams write clear specifications of intended behavior, then generate, implement, or verify code against those specifications. The spec becomes the authoritative description that both humans and machines use to understand, build, and maintain the system.

A. The AI Catalyst

While spec-first thinking is not new—Test-Driven Development (TDD) and Behavior-Driven Development (BDD) have advocated for it for years—the emergence of AI coding assistants [15], [28] has made SDD newly relevant. The problem is simple: AI models are excellent at pattern completion but poor at mind reading.

Consider a developer who prompts an AI with: "Add photo sharing to my app." The AI must guess: What format? What permissions model? What size limits? Cloud storage or local? Compression? The result is often plausible-looking code that makes dozens of unstated assumptions—many of them wrong. This is what practitioners call "vibe coding"—relying on loose prompts that lead to inconsistent or erroneous outputs from LLMs. By providing AI with unambiguous, executable contracts, SDD enhances the reliability of coding agents and opens new avenues for scalable software creation.

Now consider the same request with a specification: "Users can upload JPEG or PNG photos up to 10MB. Photos are stored in S3 with user-ID-prefixed keys. Only the uploader can delete their photos. Photos are resized to 1024px max dimension on upload." The AI now has enough information to generate code that matches intent.

Key Insight

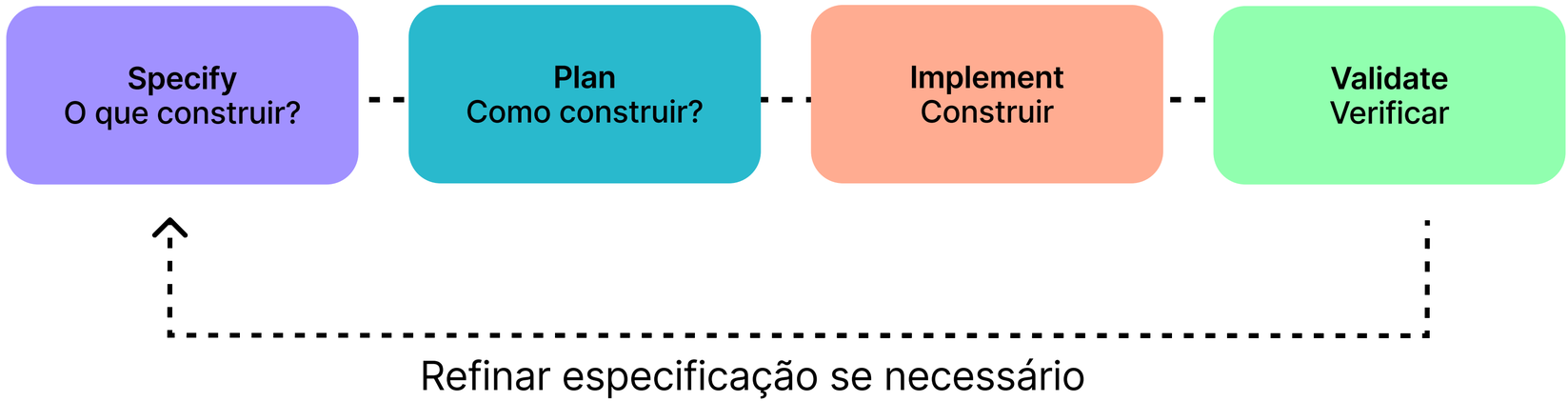
The Core Principle: In spec-driven development, code is the implementation detail of the specification—not the other way around. The spec declares intent; the code realizes it.

B. What This Paper Provides

This paper serves as a practitioner's guide to spec-driven development. We begin by defining clear levels of specification rigor—spec-first, spec-anchored, and spec-as-source—and articulating when each approach applies. We then present a practical workflow for implementing SDD, examining how the approach works both with and without AI assistance. A survey of tools and frameworks follows, ranging from traditional BDD frameworks to modern AI-assisted toolkits. Case studies illustrate SDD in action across API development, enterprise systems, and embedded software domains. Finally, we provide guidance on when SDD delivers value and when simpler approaches suffice.

Fluxo de trabalho

- Especificar antes de implementar



Fluxo de trabalho

- Especificar antes de implementar
  - Histórias de usuário
  - Critérios de aceitação
  - Requisitos

Specify
O que construir?

Fluxo de trabalho

- Especificar antes de implementar

- Stack
- Tarfeas
- Ordem de implementação

Refinar especificação se necessário

Fluxo de trabalho

- Especificar antes de implementar

- Tarefas
- Produção do código
- Revisão

Implement
Construir

Validate
Verificar

Refinar especificação se necessário

Fluxo de trabalho

- Especificar antes de implementar

- Testes
- Correções
- Finalização

Validate
Verificar

Refinar especificação se necessário

Qual o objetivo do SDD?

- Eliminar ambiguidades
- Servir de contrato entre I.A e humanos
- Desenvolvimento paralelo
- Alinhamento Contínuo
- Reduzir o retrabalho

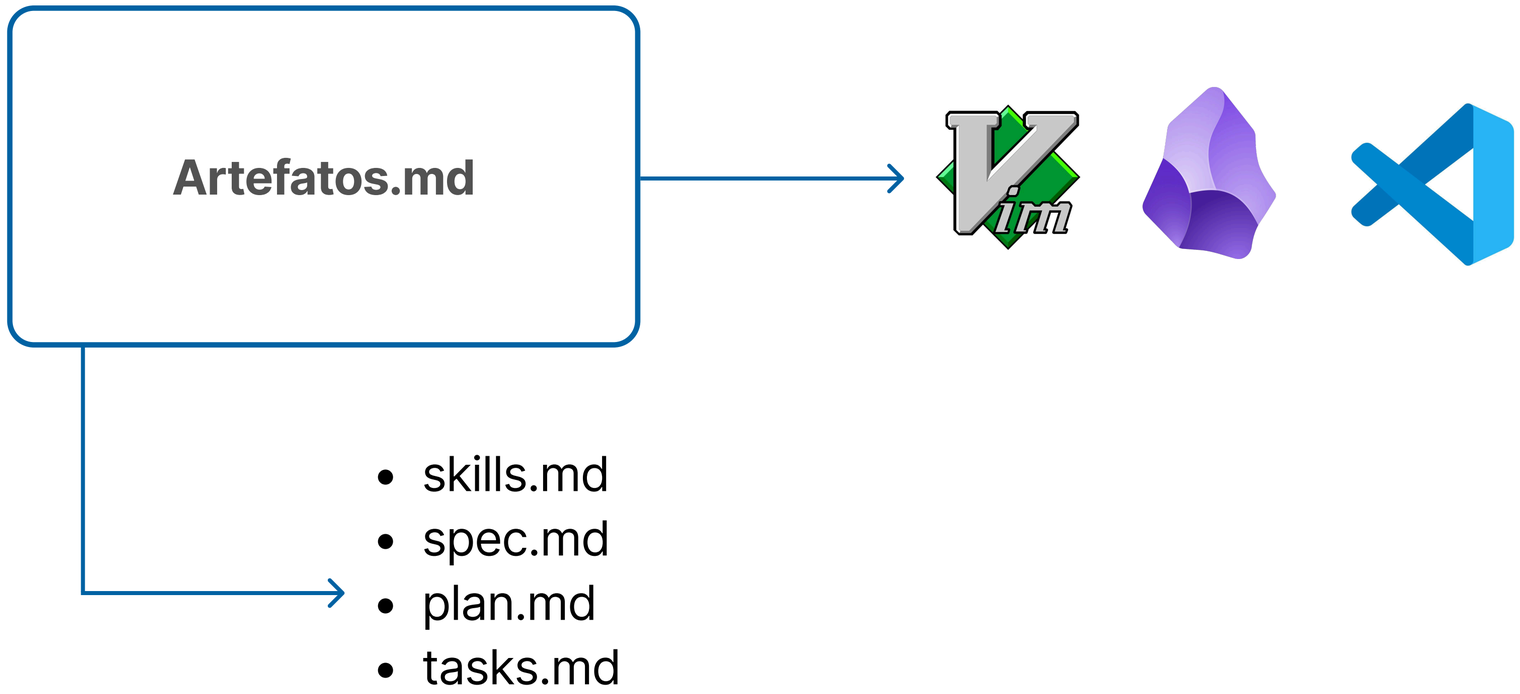
Comunicação com a IA em SDD

Artefatos.md

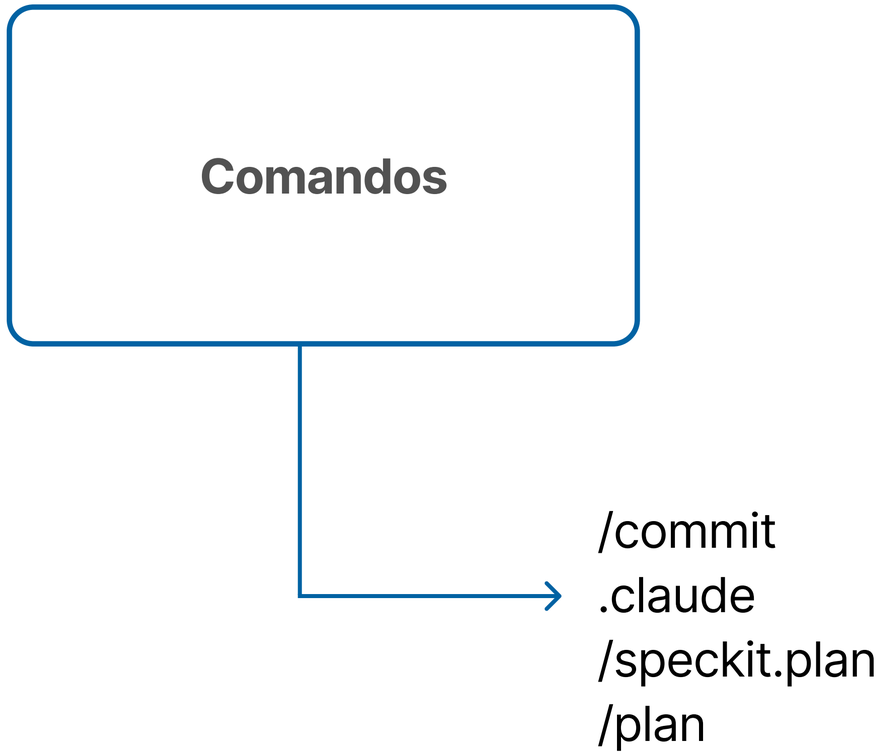
Comandos

Prompt

Comunicação com a IA em SDD



Comunicação com a IA em SDD



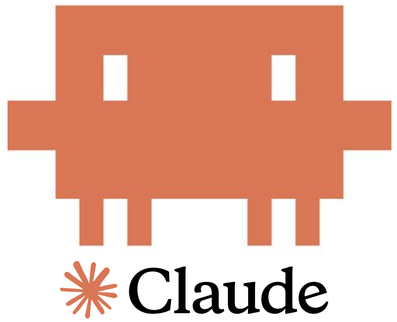
Comunicação com a IA em SDD

”Crie uma função em Python chamada cadastrar_usuario que receba três parâmetros: nome (string), email (string) e senha (string).”

- Descrição
- Perguntas
- Aprovação

Como usar o Spec-Driven Development?

Agente

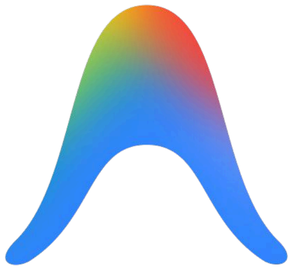


Como usar o Spec-Driven Development?

Editior

Visual Studio Code

VSCodium

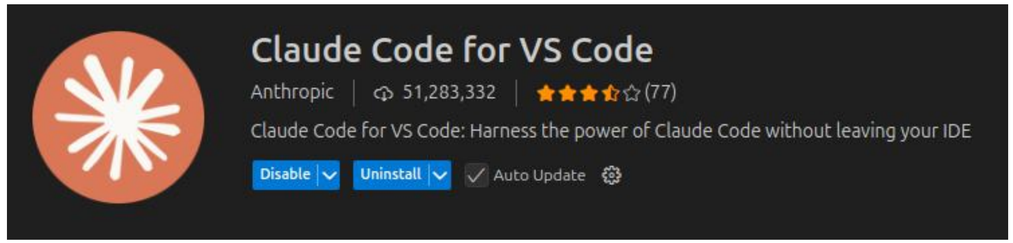


Google Antigravity

Kiro

Como usar o Spec-Driven Development?

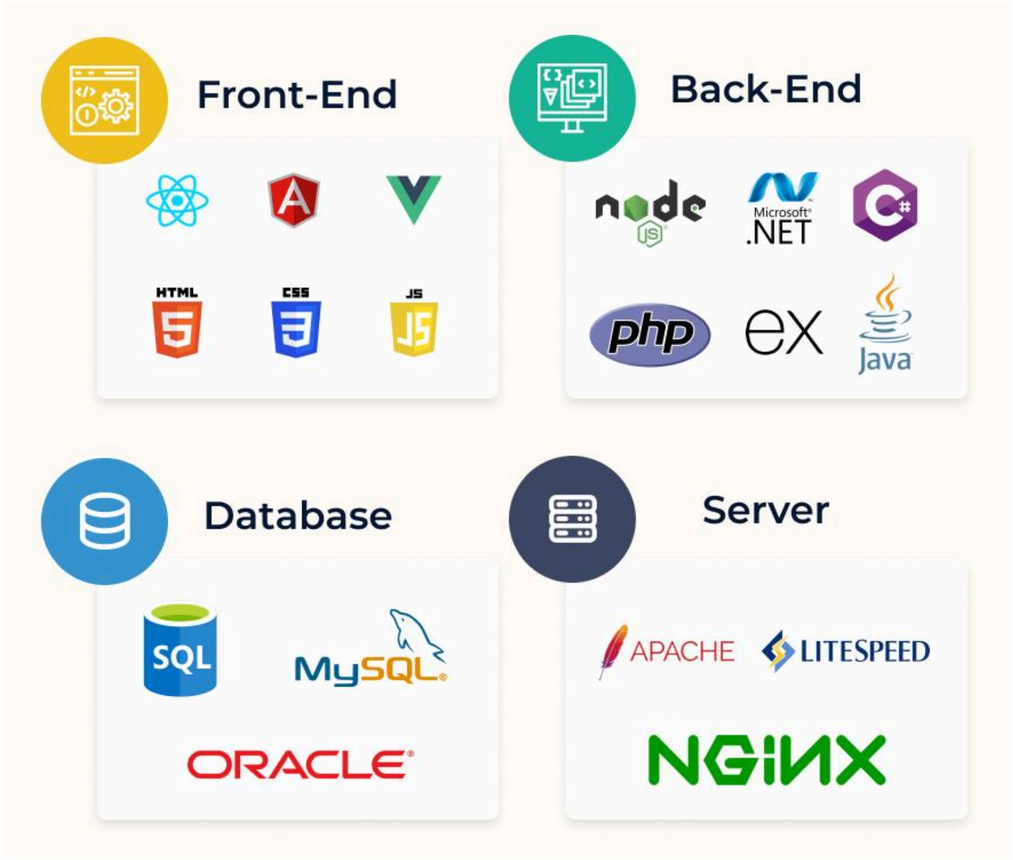
Extensões



- Opcional
- Facilita o uso

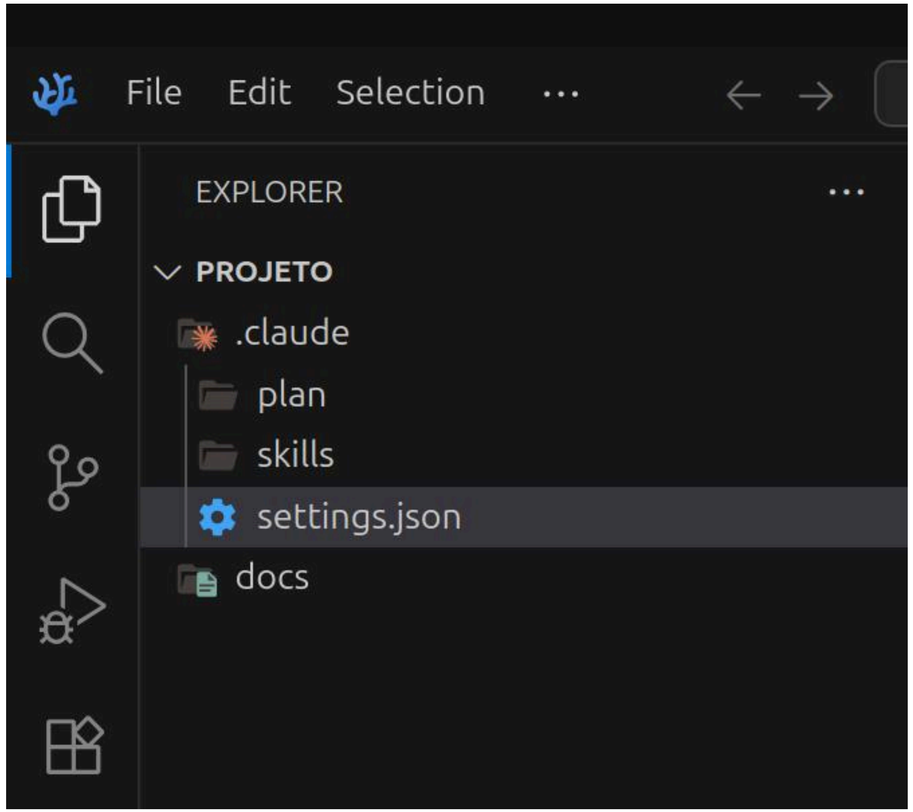
Como usar o Spec-Driven Development?

Stack



Como usar o Spec-Driven Development?

Diretório



- .claude/ — Configuração do agente
- plan/ — Planejamento
- skills/ — Habilidades do agente
- settings.json — Permissões
- docs/ — Documentação / Especificação

Como usar o Spec-Driven Development?

Iniciando o agente

Da promessa à evidência

Does Spec-Driven Development Reduce Defects? An Empirical Test of Industry Claims Across 119 Open-Source Repositories

20 Pages • Posted: 28 Apr 2026

Brenn Hill
Delivery Hero S.E.

Date Written: April 03, 2026

Abstract

Spec-driven development (SDD) tools — GitHub's Spec Kit, Amazon's Kiro, and others — claim that writing specifications before implementation reduces defects, prevents rework, and improves code quality. These claims lack empirical evidence. We provide the first large-scale test: 100,247 pull requests across 119 open-source repositories, using SZZ defect tracing and within-author fixed-effects estimation. Five hypotheses derived from vendor claims are tested. None are supported. Within-author, specifications are associated with higher defect rates (+1.4pp, $p = 0.056$) and higher rework (+5.0pp, $p < 0.001$). Specification quality has zero effect on rework ($p = 0.997$). The AI scope-constraint claim is null ($p = 0.617$). Four robustness checks — alternative outcomes, JIT defect features, individual quality dimensions, and repo-level aggregation — all confirm. Specification artifacts proxy for task complexity, not quality improvement.

Keywords: Spec-driven development, AI-assisted software engineering, defect prediction, SZZ algorithm, within-author fixed effects, code quality, empirical software engineering, confounding by indication

Declaration of Interest

None

Ethics Approval

NA

Funder Statement

NA

Suggested Citation:

Hill, Brenn, Does Spec-Driven Development Reduce Defects? An Empirical Test of Industry Claims Across 119 Open-Source Repositories (April 03, 2026). Available at SSRN: https://ssrn.com/abstract=6515898 or http://dx.doi.org/10.2139/ssrn.6515898

Show Contact Information >

- SDD reduz defeitos, retrabalho e escopo do código?
- Analisa repositórios com e sem especificações
- Não encontrou evidências

Exercício: Construa um projeto com SDD

- Escolham um sistema simples. Ex: lista de tarefas (To-Do)
- Escreva a especificação
- Utilize um agente de codificação com IA para implementar o sistema
- Altere ou remova propositalmente uma parte da especificação, deixando um requisito ambíguo ou incompleto, e peça novamente ao agente para implementar a funcionalidade

Referências

KIRO. Kiro Documentation. 2026.

ANTHROPIC. Claude: The AI for problem solvers. 2026.

HILL, B. Does Spec-Driven Development Reduce Defects? 2026.

TAGHAVI, P.; BHAVANI, S. Spec Kit Agents: Context-Grounded Agentic Workflows. 2026.

PISKALA, D. B. Spec-Driven Development: From Code to Contract in the Age of AI Coding Assistants. 2026.

Obrigado
Dúvidas?

Leandro Veloso
leandro.velos@icen.ufpa.br