

Desenvolvimento Orientado à Especificação com IA

De Peter Naur à Era da Inteligência Artificial: O Caso Prático do OpenSpec

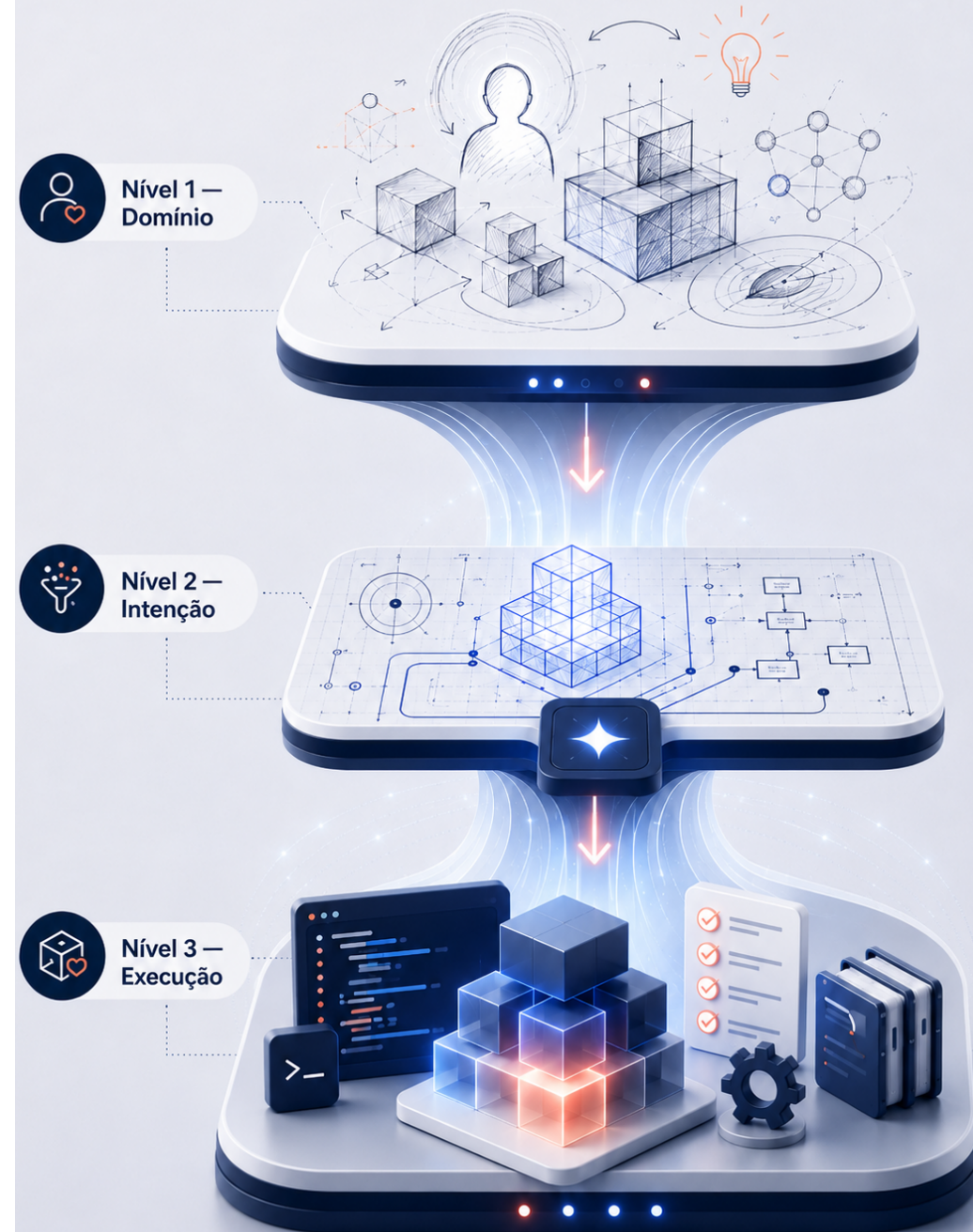
16° EnGiTEC

João Lima - 21/Maio/2026



Aplicando a IA

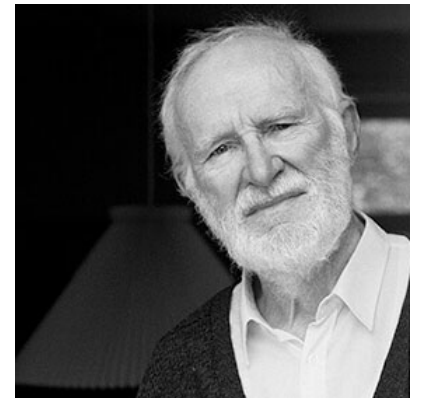
- Em 3 níveis de abstração
 - PRD – Product Requirements Description
 - Spec – a partir do PRD
 - Código – a partir das Specs



PARTE I:

A Base Teórica

Programação como Construção de Teoria (A Visão de Peter Naur)



O Paradigma Tradicional vs. Construção da Teoria

A Visão Comum

- ✗ Foco quase exclusivo na criação do texto (código-fonte).
- ✗ A documentação é vista como o produto secundário que, após entrar em produção, fica desatualizada.
- ✗ Programadores são tratados como operários de uma linha de montagem.
- ✗ Acredita-se erroneamente que regras mecânicas garantem a qualidade.

A Visão de Construção de Teoria (Naur)

- ✓ Foco total na **compreensão** do domínio do problema.
- ✓ Documentação é entendida como um auxílio para a memória.
- ✓ Programadores constroem e mantêm um **modelo mental vivo**.
- ✓ A qualidade real depende do alinhamento dessa teoria com o mundo real.

Vida, Morte e Renascimento de um Sistema



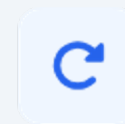
1. Vida Ativa

A fase ideal. A **equipe fundadora** possui a Teoria internalizada. O código evolui fluentemente de acordo com as necessidades do negócio, mantendo sua integridade arquitetural.



2. Morte do Sistema

A equipe original se dissolve e a Teoria é perdida. O código continua rodando em produção (Online), mas **não pode mais ser modificado com inteligência**, apenas por meio de remendos.



3. Renascimento

O sistema se torna insustentável. Uma nova equipe é chamada para reconstruir a Teoria e o sistema a partir do zero. É um processo notoriamente doloroso, demorado e extremamente caro.

O Custo Oculto das Modificações SEM Teoria

O que acontece quando alteramos código baseados apenas em suposições superficiais?

Destruição Arquitetural

- 🧨 Soluções rápidas e ignorantes sobre a estrutura original começam a minar os pilares do sistema.
- 🧨 O design inicial é corrompido, tornando cada nova alteração exponencialmente mais difícil de implementar.

O Código Frankenstein

- 🧟 O sistema se enche de "patches" isolados, códigos sem coesão colados uns aos outros.
- 🧟 **Falsa Flexibilidade:** A flexibilidade embutida originalmente torna-se inútil porque ninguém sabe qual era a intenção inicial.

A Crise na Era da Inteligência Artificial

Velocidade vs. Compreensão

A impressionante velocidade de geração da IA colide diretamente com a necessidade humana de **compreensão** e retenção de contexto a longo prazo.

O Resultado Prático: A IA gera volumes gigantescos de texto e código ultrarrápido, mas possui absolutamente **zero Teoria de Domínio** intrínseca.

Requisitos importantes se perdem rapidamente em históricos de chats efêmeros.

Sem a teoria, a capacidade de modificação entra em um colapso acelerado.



O Resgate da Teoria: Spec-Driven Development (SDD)



**Humanos e IA concordam
rigorosamente sobre o QUE construir
por meio de artefatos verificáveis,
antes de gerar qualquer linha de código.**



A Especificação é a Fonte da Verdade.

PARTE II: A Abordagem OpenSpec

Engenharia de Contexto e Spec-Driven Development na Prática



SpecKit
Kiro
BMad

Introduzindo o OpenSpec



1. Organização

Pastas dedicadas com o propósito claro de abrigar specs, designs e tarefas de implementação de forma limpa e estruturada no repositório.



2. Integração Universal

Projetado para atuar em harmonia com +25 assistentes do mercado (Claude, GitHub Copilot, Cursor, Windsurf) através de comandos slash via prompt.



3. Previsibilidade

Traz a segurança e a consistência de métodos de engenharia robustos, combinados com a agilidade necessária para o fluxo de IA.

Codificação Tradicional IA vs. SDD (OpenSpec)

Codificação *ad hoc* (Sem SDD)

- Uso constante de prompts longos, vagos e frequentemente confusos.
- Requisitos ficam reféns do histórico do chat, perdendo-se ao recarregar a janela.
- Resultados altamente imprevisíveis gerando código frágil.
- Nenhuma teoria preservada de forma tangível para o futuro do projeto.

Fluxo OpenSpec (Com SDD)

- ✓ Alterações guiadas e auditadas por pequenos artefatos estruturados.
- ✓ Especificações vivem de forma permanente no repositório, totalmente versionadas.
- ✓ Processo essencialmente iterativo, previsível e fácil de ser verificado.
- ✓ A "Teoria" é ativamente mantida, atualizada e transferível entre times.

Os 4 Pilares da Filosofia OpenSpec



Fluido, não rígido

Não há fases engessadas ('waterfall'). Atualize artefatos a qualquer momento conforme descobre mais sobre o problema.



Iterativo, não cascata

O entendimento profundo se revela durante o processo de construção. A teoria evolui junto com as tentativas de codificação.



Fácil, não complexo

Sem configurações burocráticas enormes ou processos dolorosos; o objetivo é sair do seu caminho para que você apenas especifique e codifique.



Brownfield-first

Projetado desde o dia zero para modificar e estender sistemas complexos já existentes, que é a realidade da maior parte da engenharia.

Como atualizar o OpenSpec



1 Atualize o pacote



```
>_ npm install -g @fission-ai/openspec@latest
```



Instala a versão mais recente globalmente.



2 Atualize as instruções do agente

Execute dentro de cada projeto.



```
>_ openspec update
```



Regenera a orientação de IA e ativa os slash commands mais recentes.



upgrade do pacote



refresh das instruções

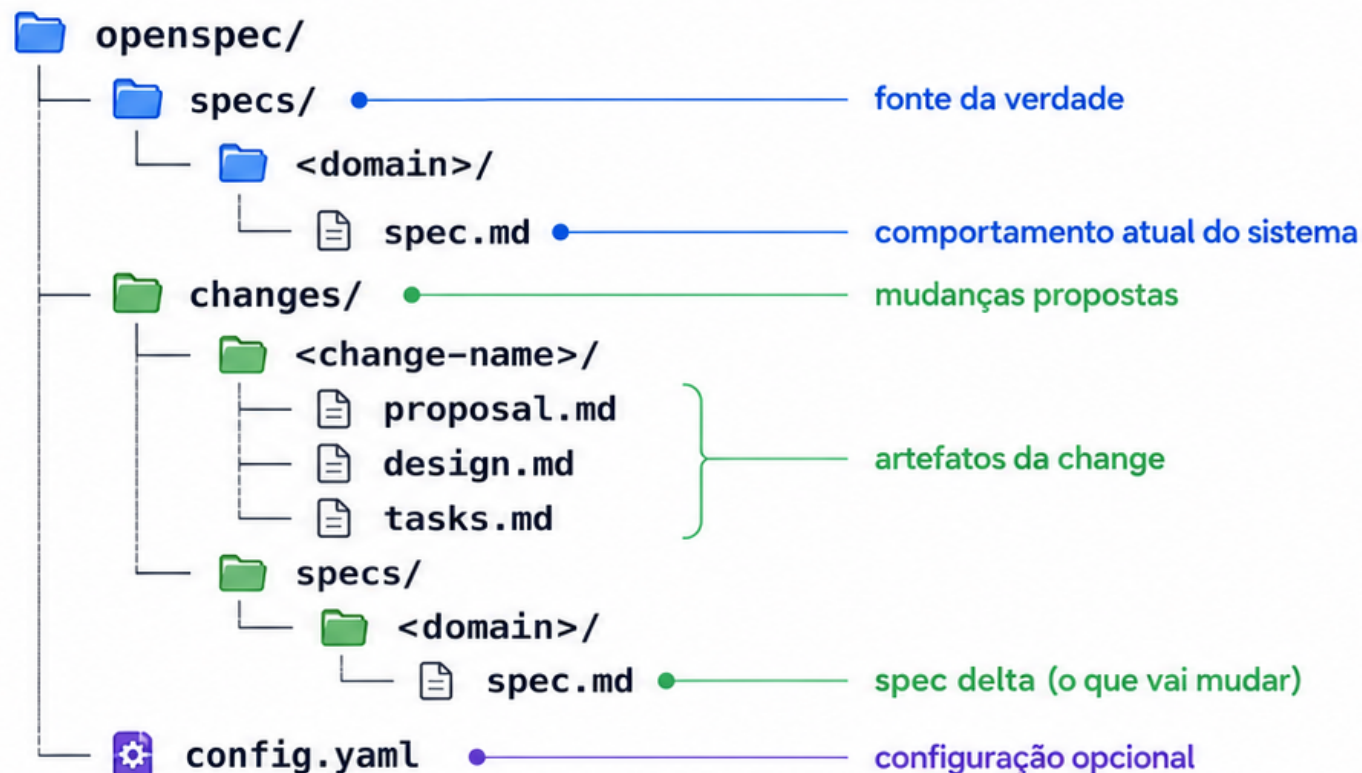


slash commands atualizados

O que o OpenSpec cria

Depois de executar ``openspec init``, seu projeto ganha esta estrutura:

```
$ openspec init
```



specs/

- fonte da verdade
- descreve o comportamento atual
- organizado por domínio

ex.: specs/auth/ • specs/payments/



changes/

- modificações propostas
- cada change tem sua própria pasta
- quando concluída, a spec é incorporada à base

1



specs/
= base atual



2



changes/
= evolução proposta



3



merge final
= atualização da base

A Arquitetura da Solução: IA em 3 Níveis

Como construímos a teoria em etapas progressivas de abstração.

Nível 1: O Domínio

(A Fonte da Verdade)

A criação do *Product Requirements Document* (PRD).
O foco absoluto aqui é detalhar o comportamento observável do sistema.
O humano fornece a intenção central e a IA atua redigindo, isolando a regra de negócio do código final.

Nível 2: A Intenção

(A Proposta)

A derivação de uma Proposta de mudança de forma isolada do resto do sistema.
Atua como a ponte entre a visão global (PRD) e o que de fato será executado, focando a atenção da IA sem gerar distrações [contexto].
PRD > Specs (MVP + Changes)

Nível 3: A Execução

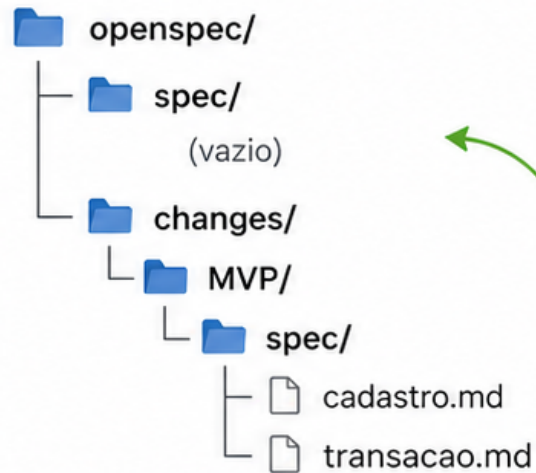
(O Código)

A tradução determinística da Spec aprovada em código real através de uma checklist de tarefas geradas e auditadas.
Garante rastreabilidade de onde surgiu cada linha escrita no terminal.

OpenSpec (SDD) — Greenfield vs Brownfield



Greenfield



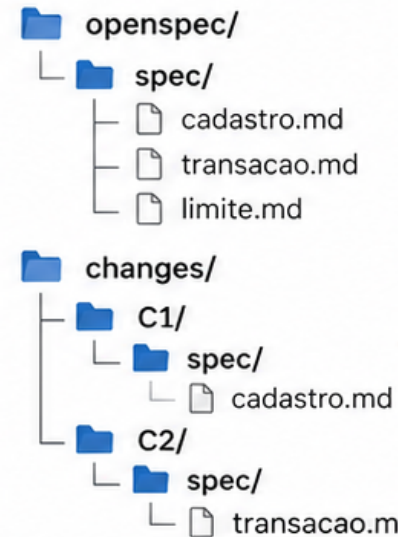
- produto novo
- sem legado
- spec base vazia



Greenfield = criar a base



Brownfield



spec base existente

- cadastro.md
- transacao.md
- limite.md

atualiza cadastro.md

atualiza transacao.md

- sistema existente
- com legado
- mudanças incrementais

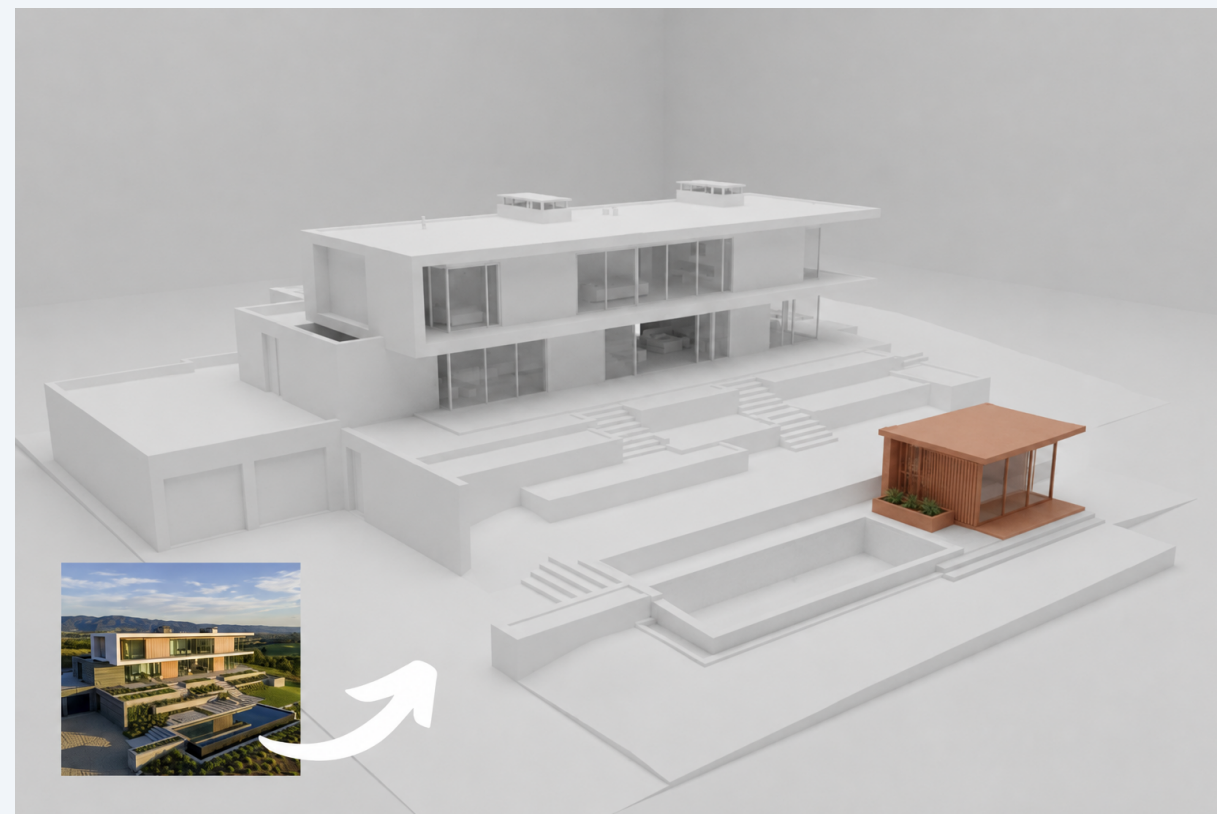


Brownfield = evoluir a base

O Segredo do Nível 2: Delta Specs

Descrevendo a diferença em um ambiente isolado (Sandbox), não recriando o mundo.

- **A Realidade Brownfield:** Aceita o fato de que 90% do trabalho do engenheiro é modificar código que já existe.
- **A Proposta:** Em vez de reescrever PRDs inteiros, o fluxo cria as "Delta Specs".
- **Estrutura Exata:** Detalha isoladamente o comportamento novo (**ADDED**), a alteração de regra (**MODIFIED**), e a funcionalidade obsoleta (**REMOVED**).
- **A Grande Vantagem:** Evita conflitos de instrução na IA, reduzindo a chance de o assistente alucinar e quebrar uma funcionalidade já validada no sistema.



O Fluxo de Comandos e Consolidação

Transformando a proposta em software e atualizando a Teoria.

`/openspec apply`

A Execução

Passo a passo, a IA lê a proposta e a lista de tarefas.

Executa no código-fonte, atualizando as caixas de seleção [x] de tasks.



`/openspec proposal`

O Início

Cria a intenção estruturada. A IA lê as convenções e o PRD atual.

Regra de Ouro: Nunca inicie código sem uma proposta gerada.

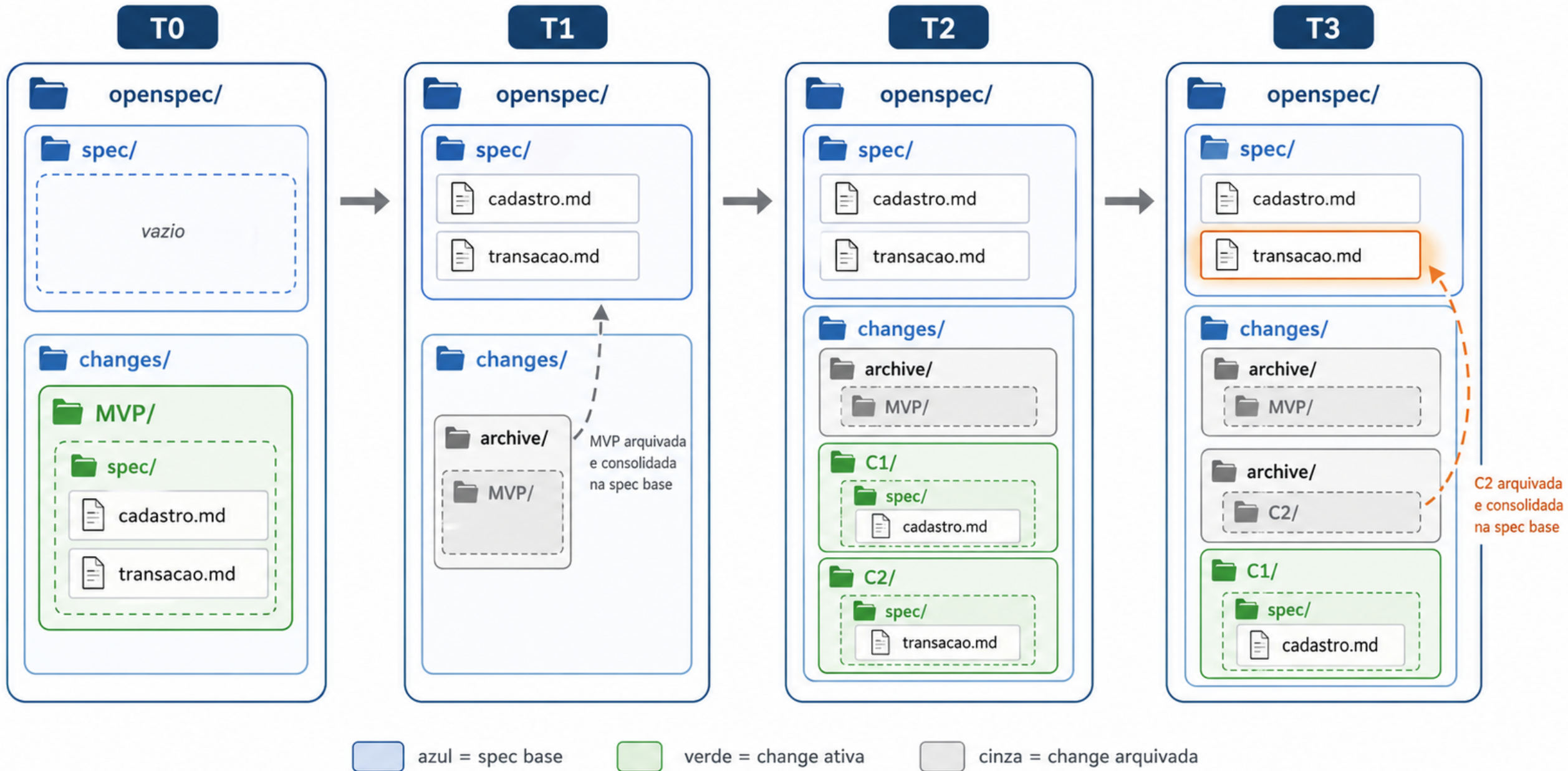


`/openspec archive`

A Consolidação

Fecha o ciclo. O PRD Master absorve os Deltas aprovados. A pasta da mudança é arquivada e o Conhecimento Tácito vira Explícito.

OpenSpec (SDD) — evolução da estrutura de diretórios



OpenSpec (SDD) — proposal → apply → archive

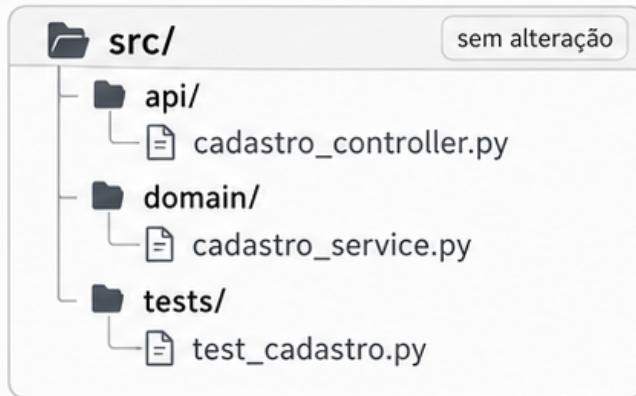
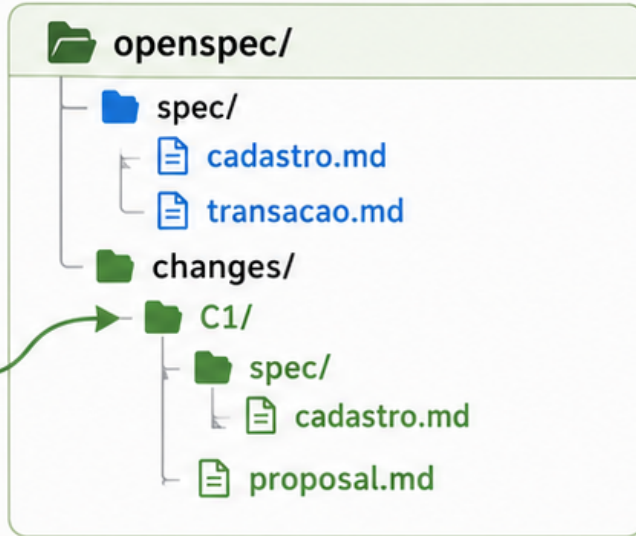


proposal

\$ openspec proposal C1

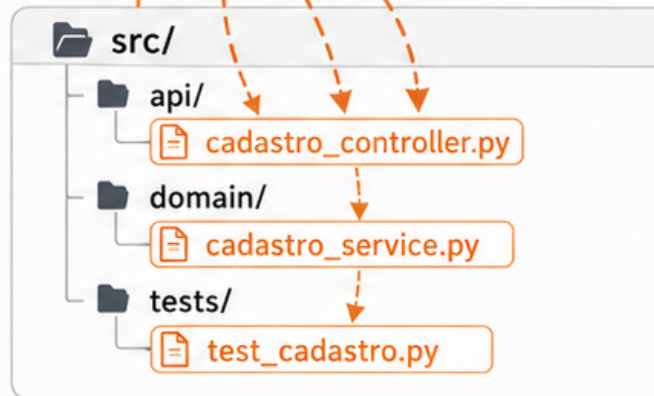
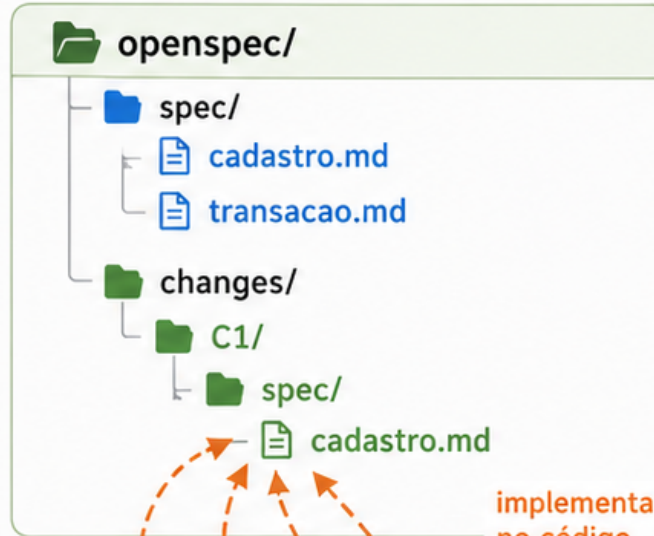


ideia



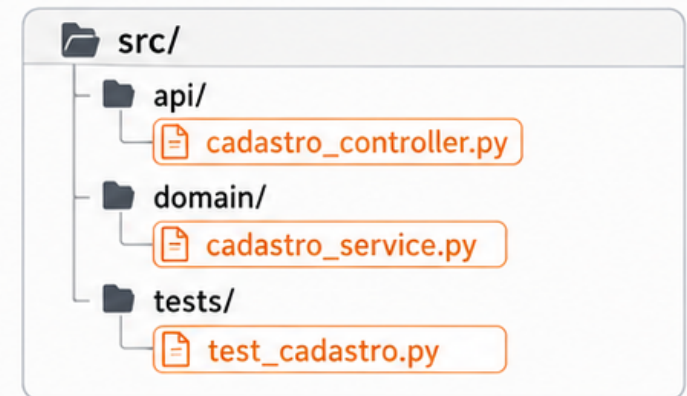
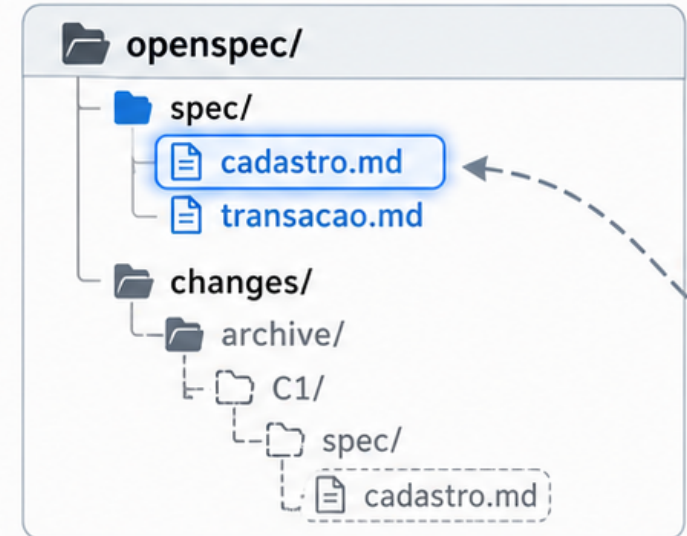
apply

\$ openspec apply C1



archive

\$ openspec archive C1



azul = spec base



verde = proposta ativa



laranja = código aplicado



cinza = change arquivada

OpenSpec — comandos de inspeção (1/2)

COMANDO

1 # List active changes

```
openspec list
```

O QUE FAZ



Lista todas as changes ativas (em aberto).

EXEMPLO DE SAÍDA

```
$ openspec list
```

CHANGE NAME	STATUS	CREATED
add-dark-mode	active	2024-05-20
payments-refactor	active	2024-05-18
add-mfa	active	2024-05-15

3 active change(s)

ONDE ATUA

```
openspec/  
├── specs/  
│   ├── auth/  
│   └── payments/  
├── changes/  
│   ├── add-dark-mode/  
│   ├── payments-refactor/  
│   └── add-mfa/  
└── config.yaml
```

Lê a pasta changes/

COMANDO

2 # View change details

```
openspec show add-dark-mode
```

O QUE FAZ



Mostra os detalhes de uma change específica: propósito, escopo, tarefas, specs afetadas, etc.

EXEMPLO DE SAÍDA

```
$ openspec show add-dark-mode
```

```
Change: add-dark-mode  
Status: active  
Created: 2024-05-20  
Description:  
Adiciona suporte ao modo escuro na aplicação.  
Affected specs:  
• specs/ui/theme/spec.md  
• specs/settings/preferences/spec.md  
Tasks: 4 total, 0 complete  
[...]
```

ONDE ATUA

```
openspec/  
├── specs/  
│   ├── ui/theme/  
│   └── - settings/preferences/  
├── changes/  
│   └── add-dark-mode/  
│       ├── proposal.md  
│       ├── design.md  
│       ├── tasks.md  
│       └── specs/  
│           ├── ui/theme/spec.md  
│           └── settings/preferences/spec.md  
└── config.yaml
```

Lê os arquivos da change específica

OpenSpec — comandos de inspeção (2/2)

COMANDO	O QUE FAZ	EXEMPLO DE SAÍDA	ONDE ATUA										
3 # Validate spec formatting openspec validate add-dark-mode	 Valida a formatação e a estrutura das especificações da change.	\$ openspec validate add-dark-mode <table><tr><td>✓ proposal.md</td><td>OK</td></tr><tr><td>✓ design.md</td><td>OK</td></tr><tr><td>✓ tasks.md</td><td>OK</td></tr><tr><td>✓ specs/ui/theme/spec.md</td><td>OK</td></tr><tr><td>✓ specs/settings/preferences/spec.md</td><td>OK</td></tr></table> Validation passed! ✓	✓ proposal.md	OK	✓ design.md	OK	✓ tasks.md	OK	✓ specs/ui/theme/spec.md	OK	✓ specs/settings/preferences/spec.md	OK	<pre>openspec/ ├── specs/ │ ├── ui/theme/ │ │ └── settings/preferences/ ├── changes/ │ └── add-dark-mode/ │ ├── proposal.md │ ├── design.md │ ├── tasks.md │ └── specs/ │ ├── ui/theme/spec.md │ └── settings/preferences/spec.md └── config.yaml</pre> Valida os arquivos da change
✓ proposal.md	OK												
✓ design.md	OK												
✓ tasks.md	OK												
✓ specs/ui/theme/spec.md	OK												
✓ specs/settings/preferences/spec.md	OK												

O Ecossistema de Artefatos OpenSpec

A anatomia padrão de uma pasta de mudança (ex: changes/add-auth/).



1. Proposal

O arquivo central de intenção. Documenta o porquê da mudança existir e suas justificativas de negócio diretas antes que qualquer código seja tocado.



2. Delta Specs & Design

Documentação modular: O que efetivamente muda no domínio de regras e quais as decisões cruciais de arquitetura (técnicas) tomadas para permitir isso.



3. Tasks

O checklist sequencial e rigoroso de implementação para a IA (ex: 1. Configurar schema, 2. Atualizar rotas, 3. Escrever testes).
Garante rastreabilidade total.

Matriz de Decisão: Quando Criar uma Proposta?

NÃO CRIAR

Alteração Direta (Apenas PR)

- Correção de bugs simples e isolados.
- Ajustes finos de Linting e formatação.
- Pequenos ajustes e refinamentos de UI (padding, cores).

Ação: Comande a IA de forma livre para alterar o código diretamente.

SIM, CRIAR (Fluxo SDD)

A mudança altera o comportamento do sistema, a arquitetura ou cria um novo domínio?

- ✓ Desenvolvimento de novas funcionalidades.
- ✓ Refatorações arquiteturais estruturais.
- ✓ Qualquer mudança em contratos de API ou regras de negócio centrais.

Ação: Use `/opsx:propose`.

Síntese: O Guardião da Teoria

Fechando o ciclo entre as lições de Naur (1985) e o Futuro do desenvolvimento com Inteligência Artificial.

A Teoria Original (1985)

Focava na compreensão humana profunda e na criação rigorosa de um modelo mental unificado.

Sua manutenção contínua e repasse de contexto através de comunicação clara são cruciais para evitar a morte prematura de um programa útil.

A Prática OpenSpec (Hoje)

Traz a abstração estruturada em propostas isoladas. Documenta os Deltas Exatos (Added, Modified, Removed).

A execução deixa de ser "dedução" e passa a ser baseada em tarefas verificáveis e mapeadas, **forçando a construção da Teoria antes de tocar no código.**

O triunfo do SDD: Transforma o "Problema da IA pura" (código farto sem contexto) em Sistemas Extremamente Resilientes.

O Futuro do Desenvolvimento de Software

De Codificadores Digitadores para Arquitetos de Domínio.

A Inteligência Artificial, em muito breve, escreverá a grande maioria das linhas técnicas de código.

O grande valor da engenharia humana passará a ser modelar o domínio de negócios complexo e garantir a rígida integridade da Teoria do sistema.

Adote o Spec-Driven Development.

Pare de apenas codar.

Comece a propor.