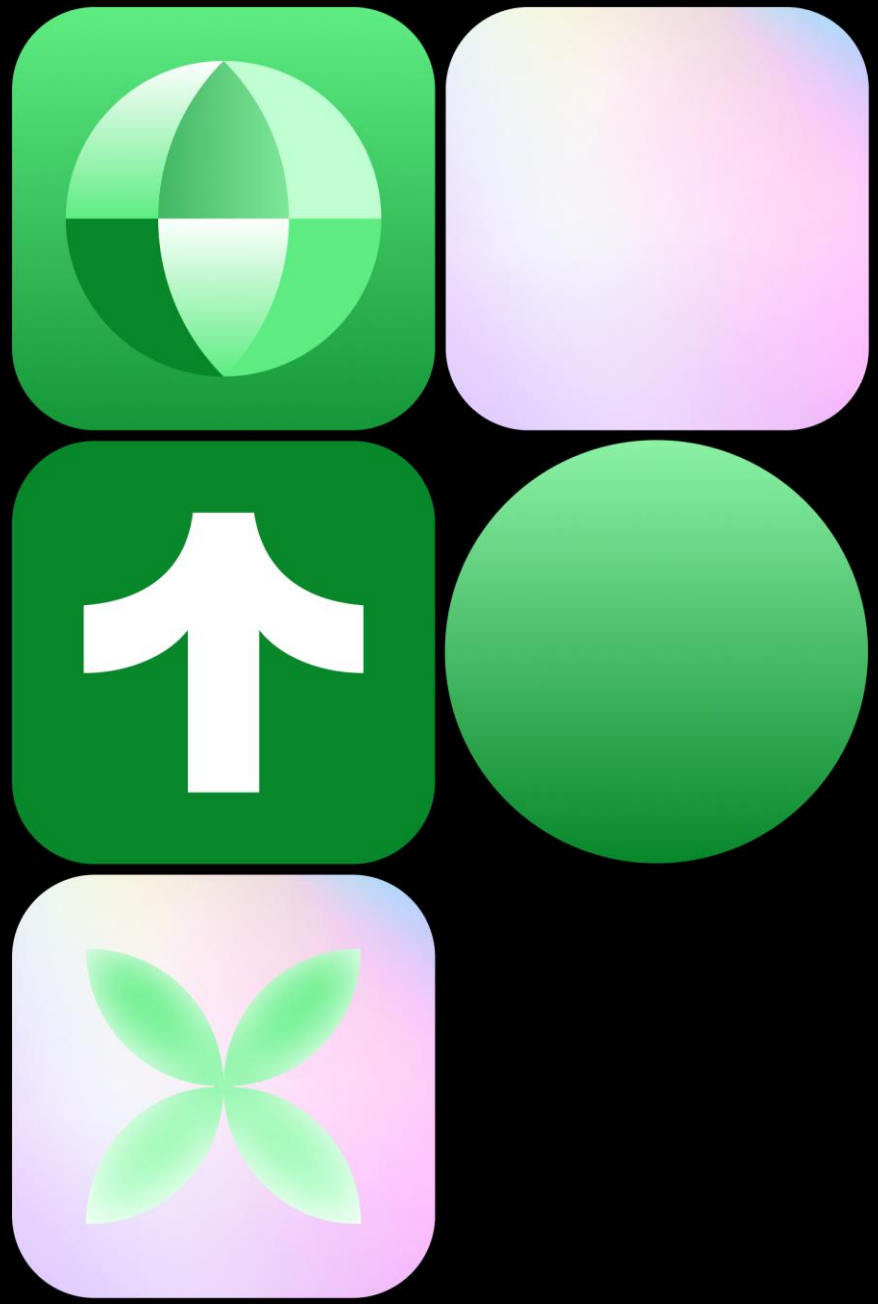


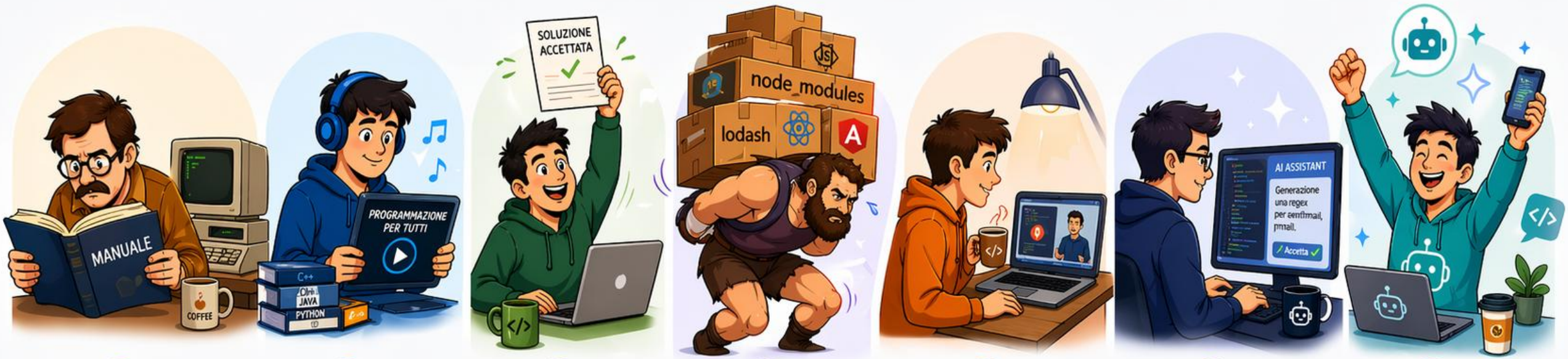
GitHub Spec-Kit:

Oltre il vibe coding.
Odiavo scrivere specifiche,
ora è amore.

Pietro Brambati

AI Apps Architect, Microsoft - in/pietro.brambati

… c’era una volta



1

RTFM Man

~1975

Tutto ciò che serve è nel manuale. Davvero.

2

DVD Learner

~2005–2010

Impara dai corsi, metti in pausa, riavvolgi, ripeti.

3

Stack Overflow Sapiens

~2008

Trova la risposta. Copiala. Capiscila (forse).

4

npm install Erectus

~2012

Più dipendenze, più potere. E più disco occupato.

5

Tutorial Hero

~2016

Segui il tutorial passo passo. Funziona!

6

Copilot Presapiens

~2021

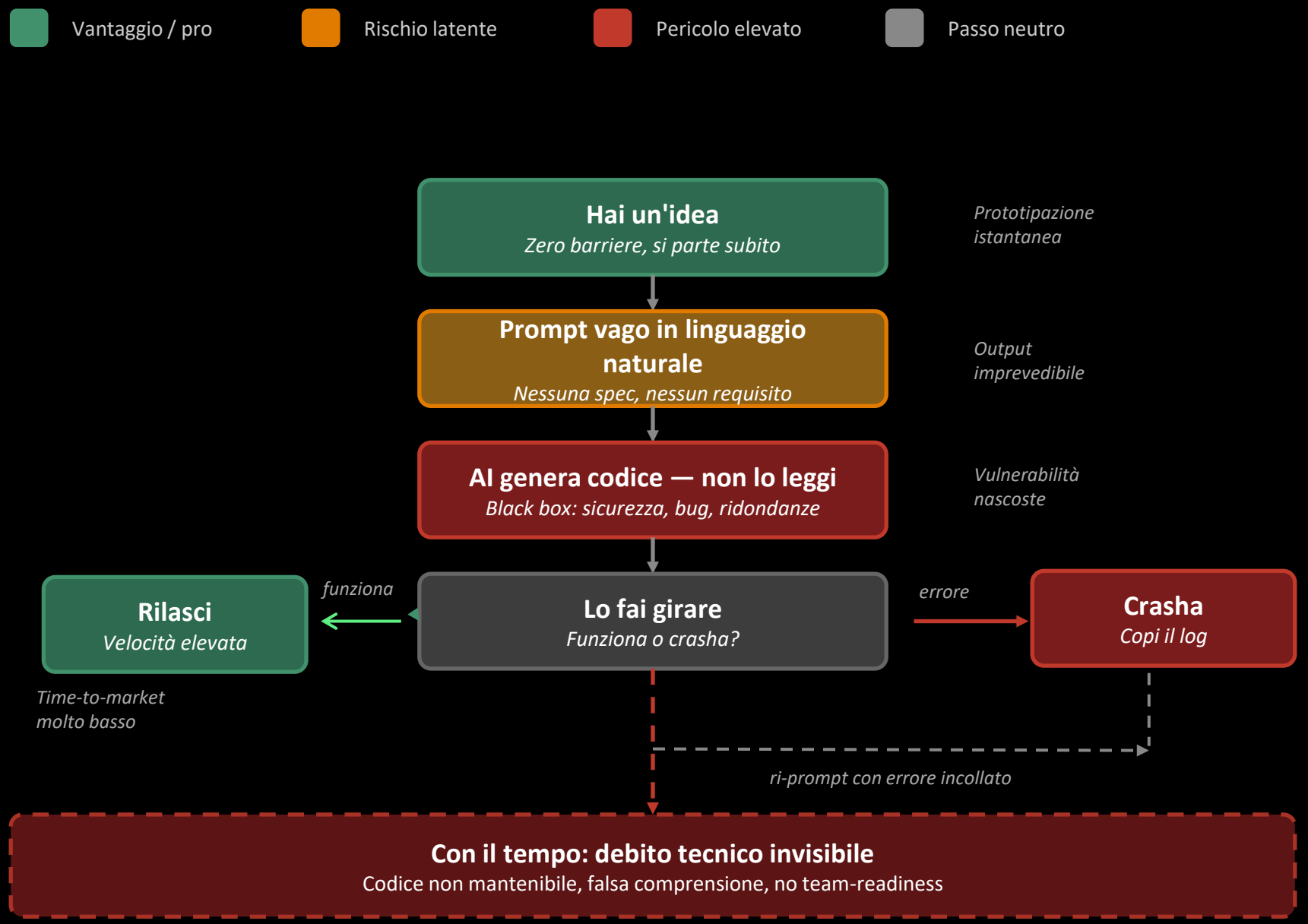
L'AI ti suggerisce, tu valuti, tu programmi.

7

Vibe Coder

~2024

Descrivi l'idea. L'AI costruisce. Tu crei il futuro.


Vantaggio / pro
Rischio latente
Pericolo elevato
Passo neutro
Hai un'idea
Zero barriere, si parte subito
Prototipazione istantanea
Prompt vago in linguaggio naturale
Nessuna spec, nessun requisito
Output imprevedibile
AI genera codice — non lo leggi
Black box: sicurezza, bug, ridondanze
Vulnerabilità nascoste
Rilasci
Velocità elevata
funziona
Lo fai girare
Funziona o crasha?
errore
Crasha
Copi il log
Time-to-market molto basso
ri-prompt con errore incollato
Con il tempo: debito tecnico invisibile
Codice non mantenibile, falsa comprensione, no team-readiness

The Register®

OSES

13

Linus Torvalds is OK with vibe coding as long as it's not used for anything that matters

Linux inventor also discusses Rust in the kernel, Nvidia's proprietary code, and the problem of AI crawlers

Esiste un modo per sfruttare l’AI, ma in modo “solido” …

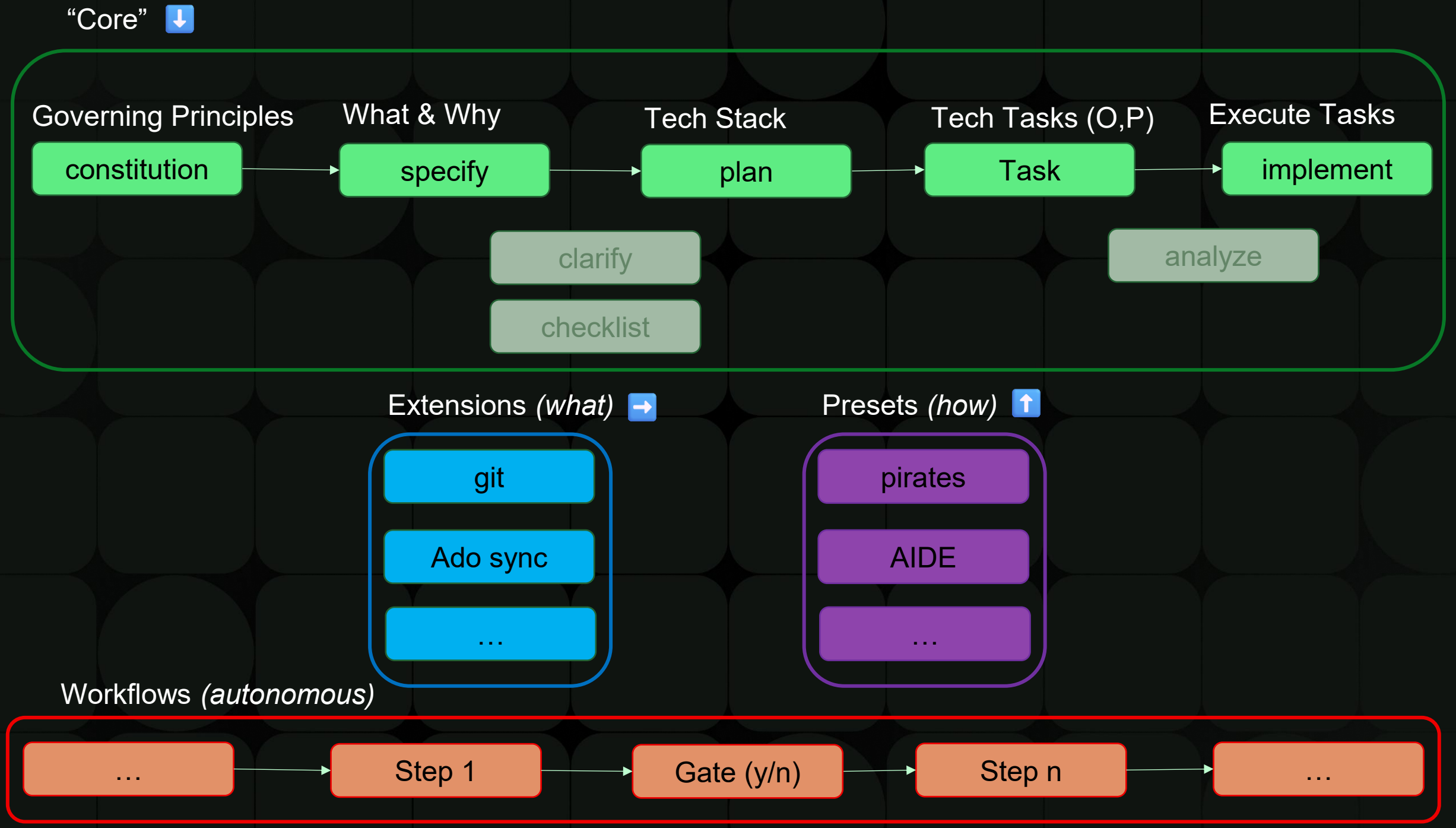
… proviamo con Spec Driven Development (SDD)

GitHub Spec-Kit

 github/spec-kit

Spec-Kit SPERIMENTALE

ATTENZIONE:
questo Spec-Kit è ancora un esperimento!

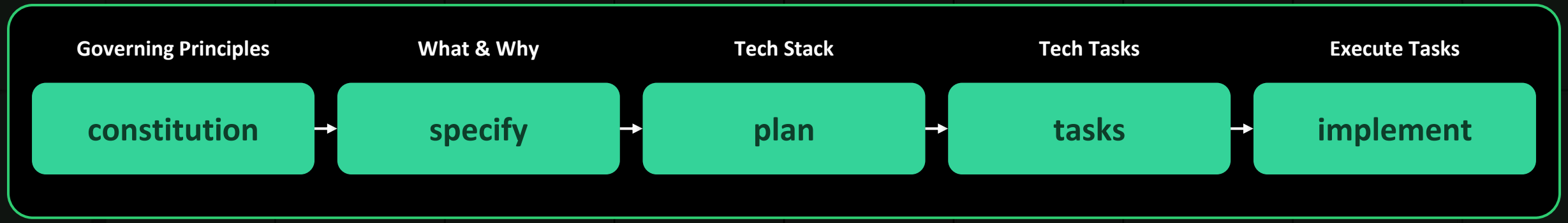

“Core” ⬇️
Governing Principles
What & Why
Tech Stack
Tech Tasks (O,P)
Execute Tasks
constitution
specify
plan
Task
implement
clarify
analyze
checklist
Extensions (what) ➡️
Presets (how) ⬆️
git
Ado sync
…
pirates
AIDE
…
Workflows (autonomous)
…
Step 1
Gate (y/n)
Step n
…

Let’s see it in action!

SPEC-KIT

Core Flow

I 5 passi principali · file generati · cosa contengono



constitution

GENERA

- constitution.md

CONTIENE

Stabilisce i principi fondanti del progetto: vincoli, valori, policy tecniche, regole di ingaggio. È il riferimento costante per tutte le fasi successive.

specify

GENERA

- spec.md
- requirements.md

CONTIENE

Descrive cosa deve fare la feature e perché: user stories, requisiti funzionali, criteri di accettazione. Nessun dettaglio implementativo.

plan

GENERA

- plan.md
- research.md
- data-model.md
- quickstart.md
- contracts/*.md

CONTIENE

Traduce la spec in architettura: stack tecnologico, modello dati, contratti API e guida di avvio rapido.

tasks

GENERA

- tasks.md

CONTIENE

Scompone il piano in task atomici ordinati, ognuno eseguibile e verificabile in modo indipendente. È la to-do list operativa per l'implementazione.

implement

GENERA

- codice sorgente
- tasks.md (✓ task completati)

CONTIENE

Esegue i task uno a uno: scrive il codice, aggiorna tasks.md marcando i completamenti, produce l'output finale del progetto.

spec-driven development · flusso sequenziale: ogni passo consuma gli artefatti del precedente

FASE 0 · FOUNDATION

constitution.md · spec.md · checklists/requirements.md

1 constitution.md

WRITE /speckit.constitution (step 0)
READ /plan (gate), /constitution

CONTIENE

Principi fondanti: Core Principles (5 slot), sezioni estensibili, Governance, versione semver, footer Ratified/Last Amended. Placeholder [PRINCIPLE_1_NAME]…

FUNZIONE QUANDO LETTO

/plan esegue il Constitution Check prima della Phase 0 e dopo la Phase 1: ogni violazione blocca il progresso salvo giustificazione in plan.md (Complexity Tracking).

2 spec.md

WRITE /speckit.specify
READ /plan, /tasks, /clarify, /analyze, /checklist

CONTIENE

User Scenarios (US1/US2/US3 con priorità P1-P3, Given/When/Then), Edge Cases, Functional Requirements (FR-001…), Key Entities, Success Criteria (SC-001…), Assumptions.

FUNZIONE QUANDO LETTO

/plan estrae entità + FR per il Technical Context; /tasks usa le user story con priorità per organizzare le fasi 3+ (una fase per story); /checklist le legge come "spec under test".

3 checklists/requirements.md

WRITE /speckit.specify (self-validation)
READ /implement (gate bloccante)

CONTIENE

Content Quality (no impl detail, stakeholder-readable), Requirement Completeness (no NEEDS CLARIFICATION, testabilità, SC misurabili), Feature Readiness (AC per FR, flow coperti).

FUNZIONE QUANDO LETTO

/implement scansiona tutti i file in checklists/: se resta un [ ] si ferma e chiede conferma esplicita. Garantisce qualità della spec prima di scrivere codice.

FASE 1 · PLAN

Un comando, 5 artefatti

4 plan.md

WRITE /plan
READ /tasks, /implement, /analyze

CONTIENE

Summary, Technical Context (linguaggio, dipendenze, storage, testing, target, performance), Constitution Check, Project Structure, Complexity Tracking.

FUNZIONE QUANDO LETTO

/tasks estrae stack e struttura per definire path e convenzioni; /implement lo usa come contesto architetturale.

5 research.md

WRITE /plan · Phase 0
READ /tasks, /implement (opz.)

CONTIENE

Un blocco per decisione: Decision / Rationale / Alternatives. Risolve ogni NEEDS CLARIFICATION emerso dalla spec.

FUNZIONE QUANDO LETTO

Separato da plan.md perché plan.md NON può contenere NEEDS CLARIFICATION dopo Phase 0. Spiega PERCHÉ una libreria o pattern è stato scelto.

6 data-model.md

WRITE /plan · Phase 1
READ /tasks, /implement (opz.)

CONTIENE

Per ogni entità: nome, campi, relazioni, validation rules (dalle FR), state transitions se applicabili.

FUNZIONE QUANDO LETTO

Ogni entità → task [P] "Create <Entity> model" assegnato alla prima user story che la usa (o Setup se condivisa).

7 contracts/*.md

WRITE /plan · Phase 1 (se API)
READ /tasks, /implement (opz.)

CONTIENE

Contratti per interfacce esterne: API, schemi CLI, endpoint, UI contracts. Omesso per tool puramente interni.

FUNZIONE QUANDO LETTO

Ogni contratto → task di contract test [P] prima dell'implementazione, nella fase della user story che serve. Parallelizzabili.

8 quickstart.md

WRITE /plan · Phase 1
READ /tasks, /implement (opz.)

CONTIENE

Walkthrough eseguibile di smoke-test derivato dalle user story, legato allo stack del Technical Context.

FUNZIONE QUANDO LETTO

Generato da /plan e non /specify: deve conoscere linguaggio e framework. Target di validazione end-to-end della Polish phase.

tasks.md · checklists/*.md — e le 4 distinzioni che definiscono il flusso

9 tasks.md

WRITE /tasks (→ /implement ri-scrive ✓)
READ /implement (req.), /analyze

CONTIENE

Phase 1 Setup, Phase 2 Foundational, Phase 3+ per user story (US1, US2, US3), Polish finale. Format: - [ ] T001 [P?] [USn?] descrizione + path.

FUNZIONE QUANDO LETTO

/implement parsea fasi, marker [P] e label [USn] per sequenziare l'esecuzione, parallelizzare i task sicuri, marcare ✓ e applicare i checkpoint di fine-fase.

10 checklists/*.md

WRITE /checklist (on-demand)
READ /implement (gate bloccante)

CONTIENE

Un file per dominio (ux.md, api.md, security.md…). Items CHK001+ con dimensione di qualità: Completeness, Clarity, Measurability, Coverage, Edge Cases, NFR.

FUNZIONE QUANDO LETTO

/implement costruisce tabella stato (Total/Completed/Incomplete). Qualsiasi item incompleto → STOP e prompt "proceed anyway?". "Unit test for English" sulla spec.

DISTINZIONI CHIAVE

research.md vs plan.md

plan.md cattura IL COSA (decisioni, stack, struttura). research.md cattura IL PERCHÉ (rationale + alternative). Così plan.md può imporre "no NEEDS CLARIFICATION dopo Phase 0".

quickstart.md è di /plan, non /specify

Richiede lo stack tecnico concreto dal Technical Context per essere eseguibile. /specify è volutamente tech-agnostico.

data-model.md → regola 1:1

Un task modello per entità, marcato [P], assegnato alla prima user story che la usa (o Setup se condivisa).

checklists/*.md = gate bloccante

/implement si ferma su qualsiasi item non completato, sbloccabile solo con un "yes" esplicito.

Il programmatore più prezioso, domani, sarà chi saprà comunicare meglio. La competenza rara del nostro tempo è scrivere specifiche che racchiudano per intero l'intento e i valori di chi le pensa, ciò richiederà esperienza.

Thank you

github/spec-kit

Pietro Brambati

AI Apps Architect, Microsoft EMEA

in/pietro.brambati