>>AI CONF 2026

Spec Driven Development: guidare gli agenti AI dalle specifiche al codice

Gian Maria Ricci

Senior Solutions Architect

improove

Kudos++

Executive

beSharp. we make IT run.

LEXROOM

REPLY STORM

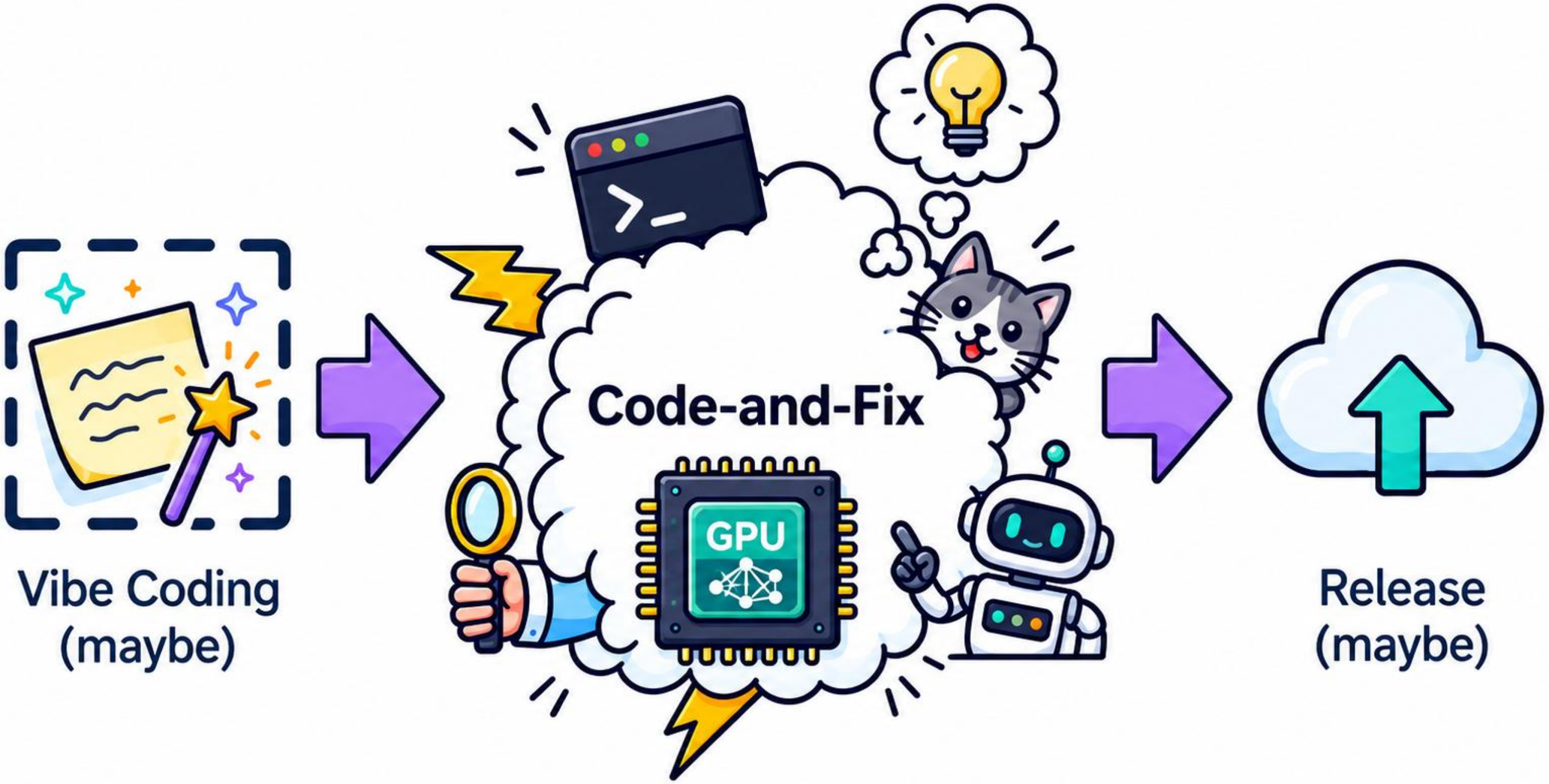
Gold

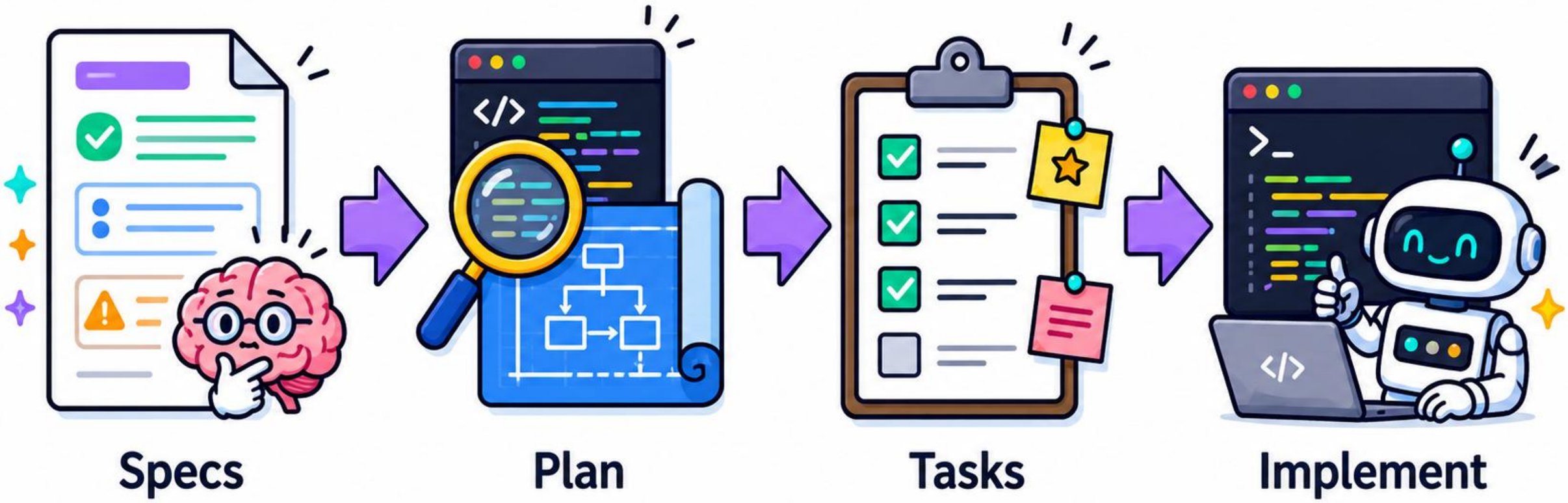
seeweb

FASTWEB + vodafone

Era il **1996** ….. (Rapid Development di Steve McConnell)

Assomiglia molto ad un certo Vibe Coding.

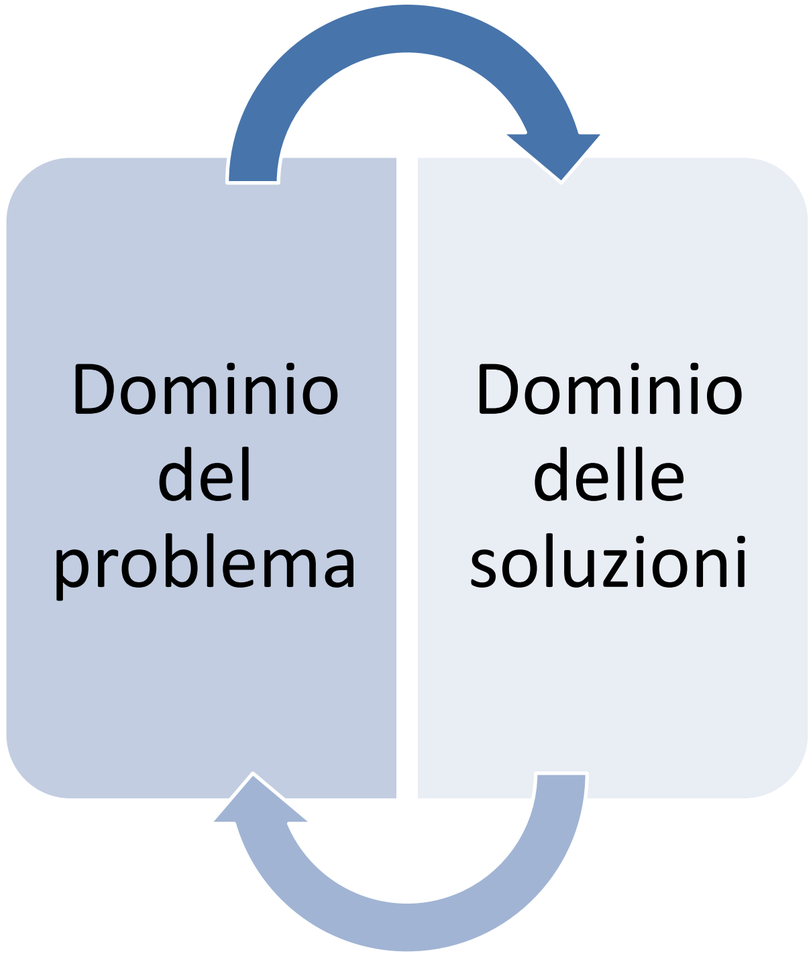




Spec driven development

Demo: Da una spec al codice

Cosa guadagniamo?



Più tempo speso nella comprensione del problema e delle possibili soluzioni.

Utilizziamo i modelli per scrivere storie coerenti e comprensibili

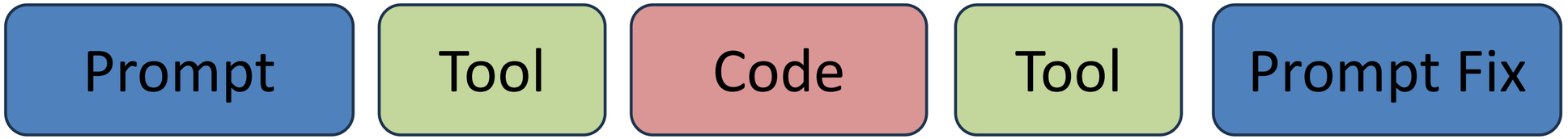
Controlli di coerenza riguardo la costituzione e precedenti storie implementate

Riduzione delle allucinazioni

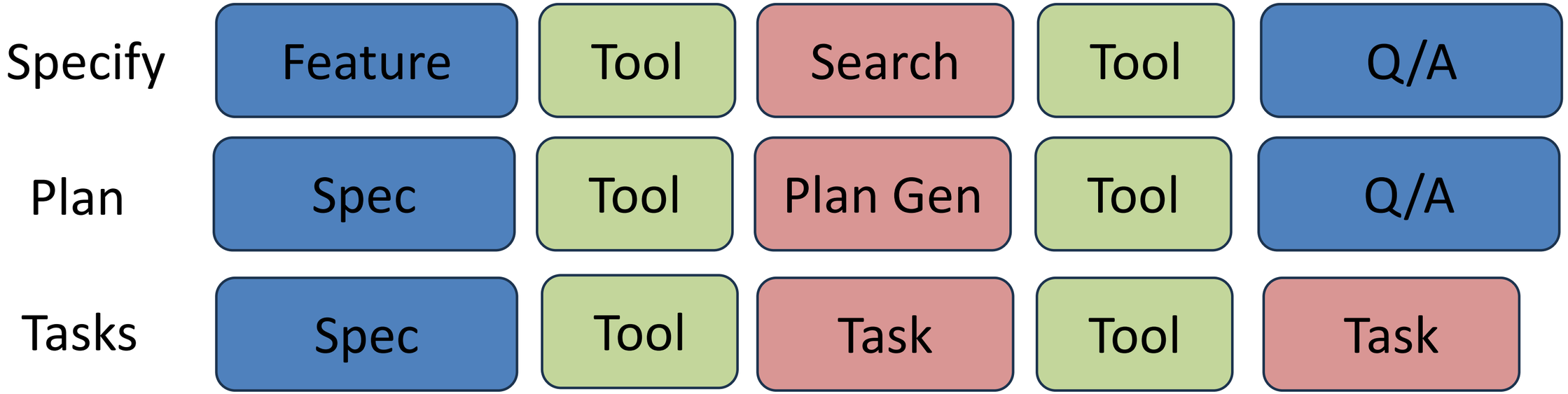
Durante la fase di specify si possono immediatamente controllare capabilities e fare grounding delle ideee

Ottimizzazione del contesto utilizzato

Vibe coding non strutturato



Spec Driven Development



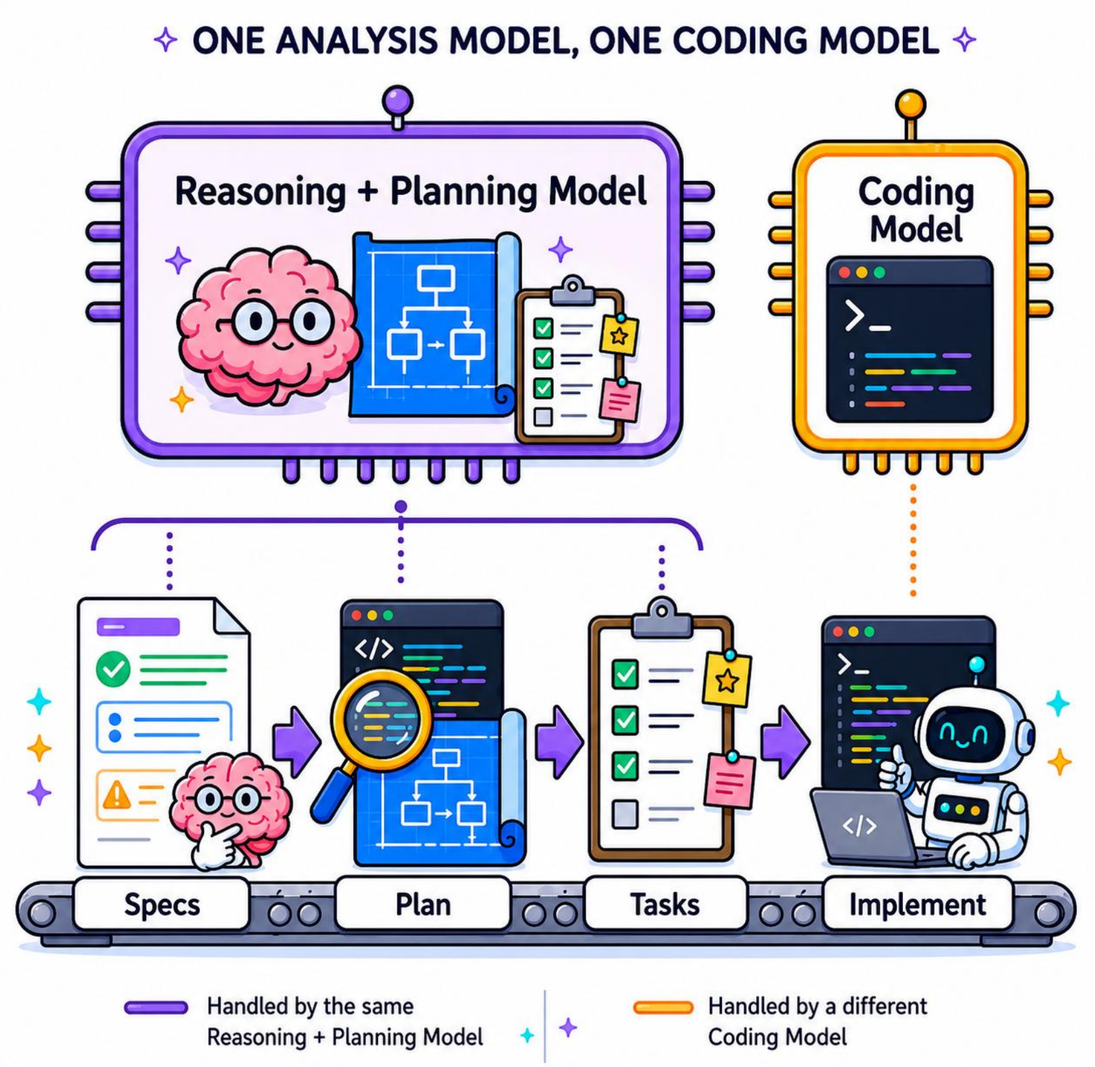
SDD ha comunque un costo

Meno codice prodotto dalla AI per unità di tempo (per me è un vantaggio)

Necessita di tempo per revisionare specifiche e piani di esecuzione

Bisogna raffinare o studiare un framework esistente

Modelli differenti durante le varie fasi

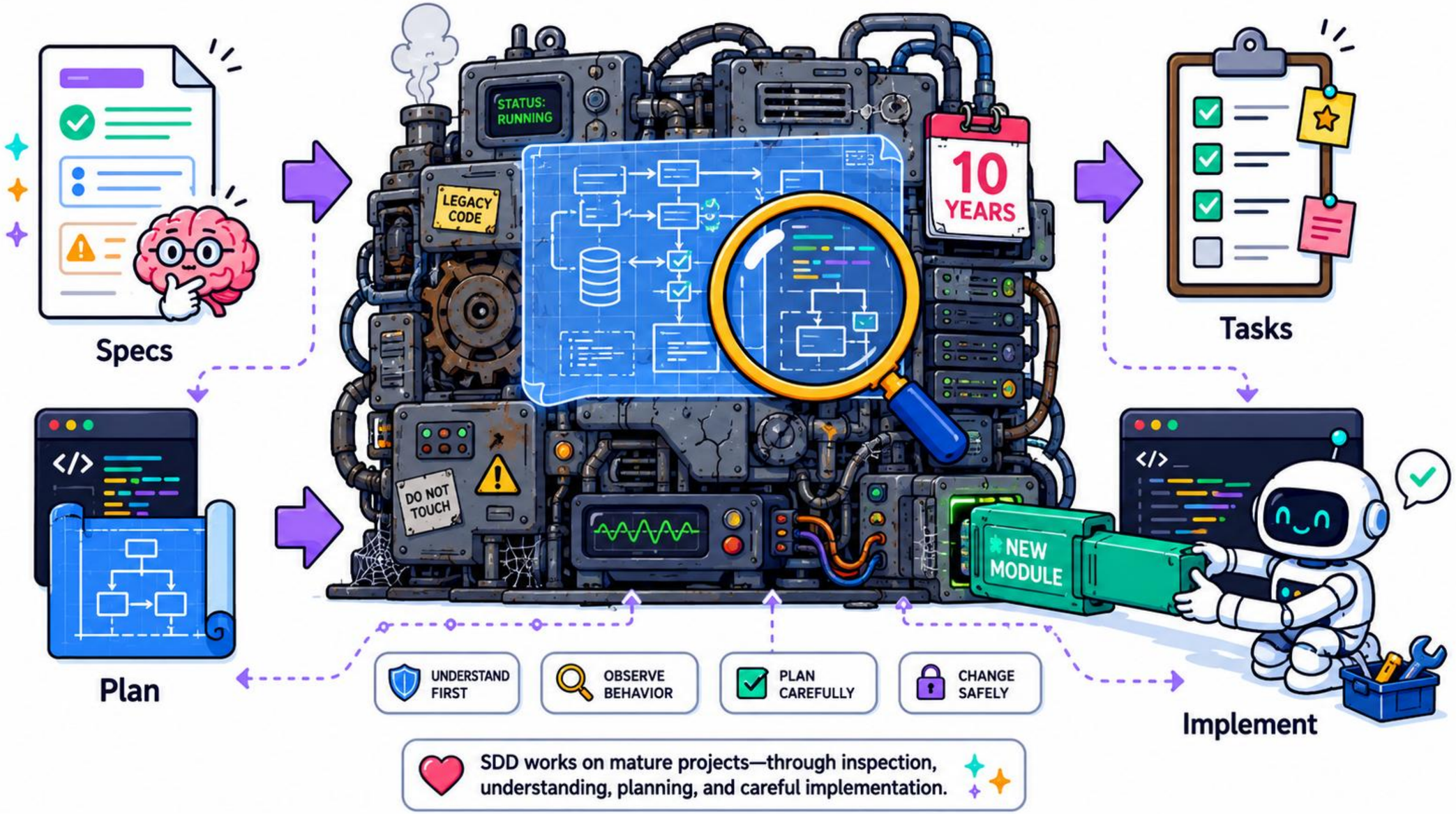


Utilizzo di modelli grandi per la fase di spec e plan, dove le capacità sono essenziali

In queste parti l’uso di token è solitamente minore

Utilizzo di modelli piccoli (haiku) per l’implementazione,quando il numero di token di output e token utilizzzati è maggiore.

Posso fare SDD su un progetto vecchio di 13 anni?



Partiamo con una classica problematica / idee

Business Description

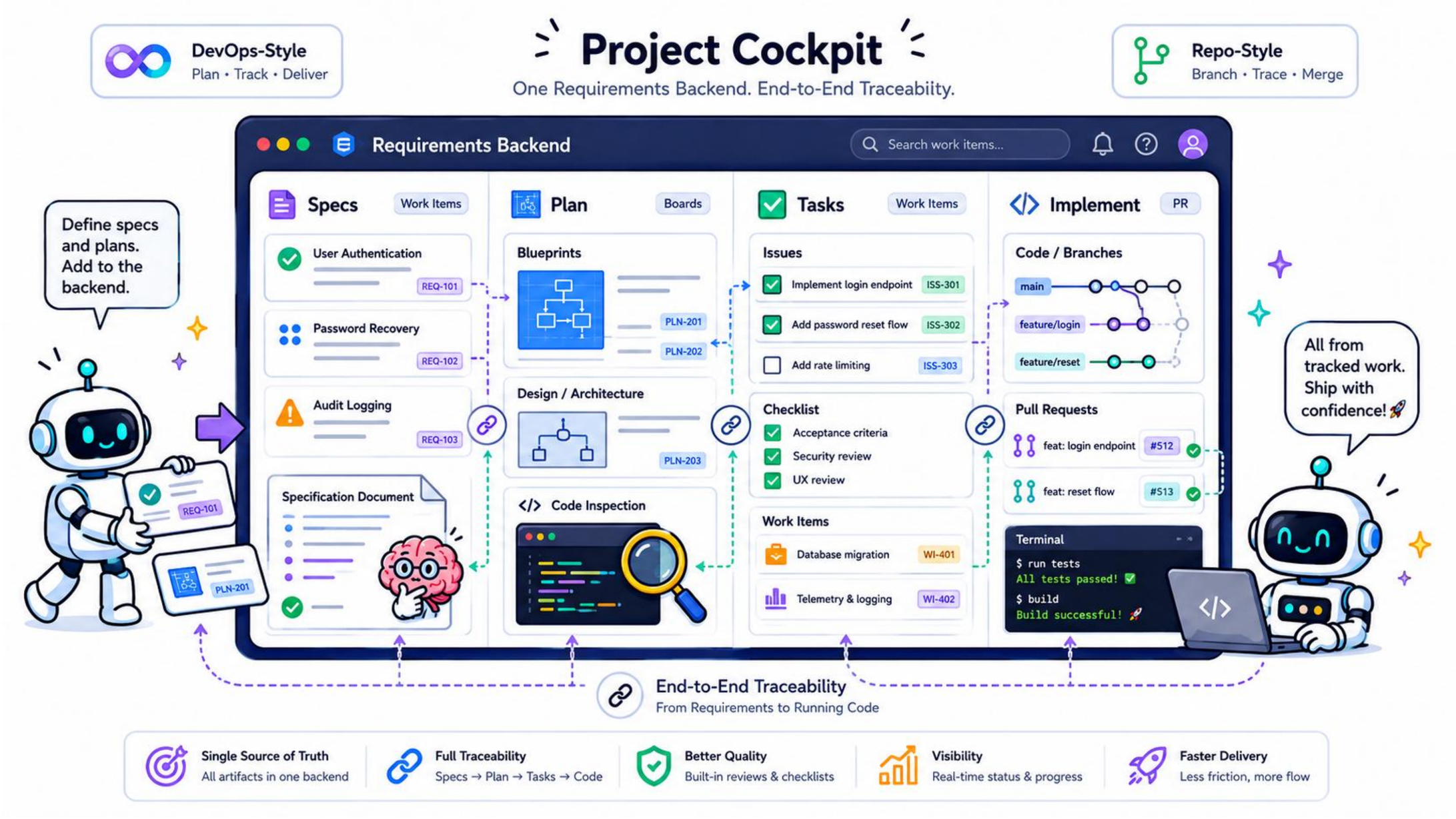
[redacted]

Sempre più spesso c'è la necessità di configurare la sezione delle ricerche in modo da:

1. poter gestire gruppi di filtri
2. Il gruppo di filtri può anche essere usato per sfruttare al meglio la gestione dei quick-filter; infatti oggi se si impostano dei filtri per escludere (ad es.) dei tipi o dei dati come filtro "base" su cui applicare il resto dei filtri, questo ha un impatto sui quickfilter se si vuole mettere come quickfilter proprio uno di quei campi utilizzato nei filtri per applicare delle condizioni di base. Invece di ipotizzare delle funzionalità nuove di "filtri base", si può usare un gruppo messo in AND con il resto dei campi base, avendo i quick filter applicabili solo sui campi base

Demo: SDD su un progetto reale, che ha anni di storia

Integrazioni con strumenti di gestione dei requisiti

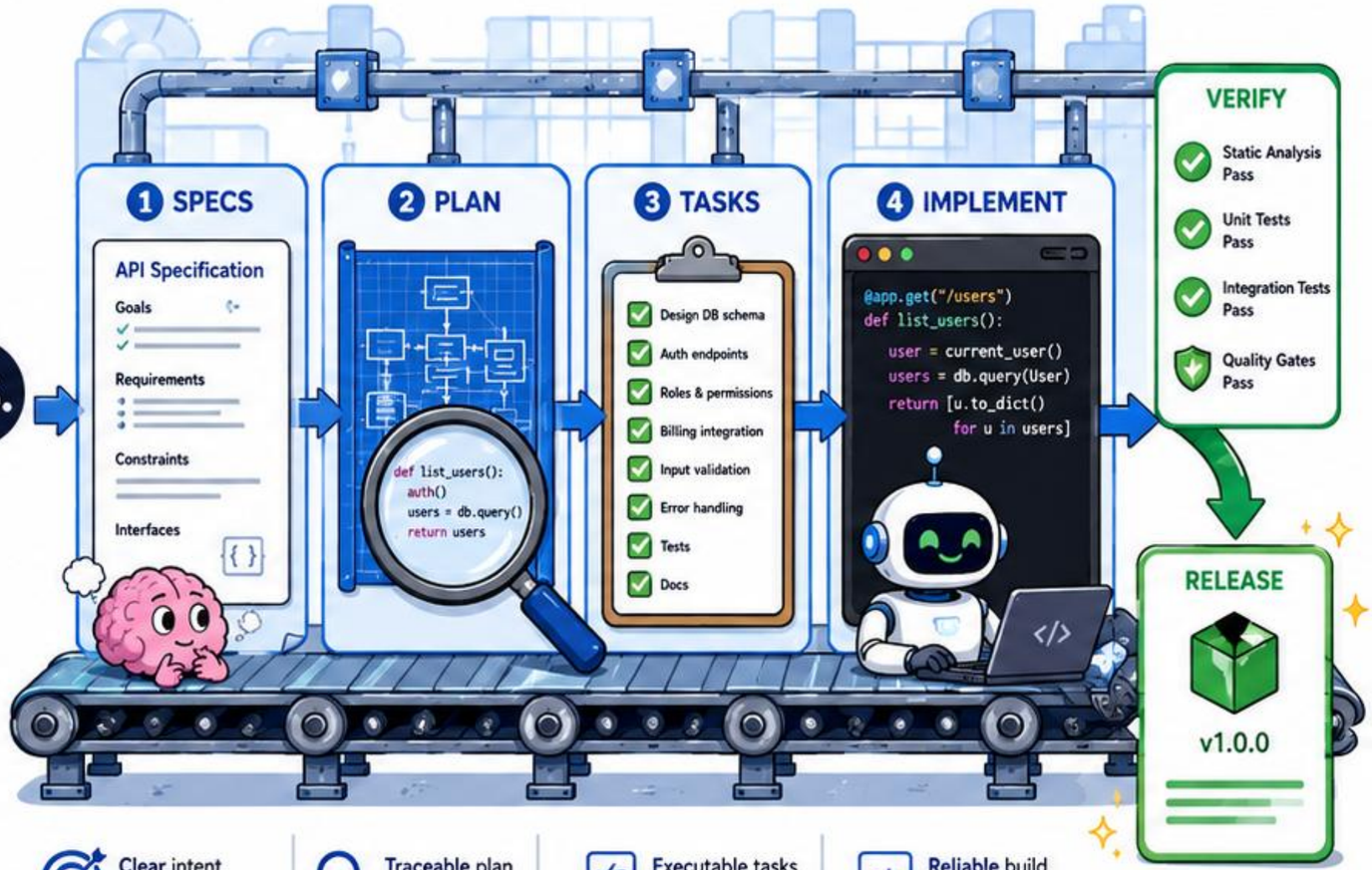


From prompt-and-pray to specify-and-verify

Prompt-and-pray

VS.

Specify-and-verify



Clear intent and requirements

Traceable plan and design

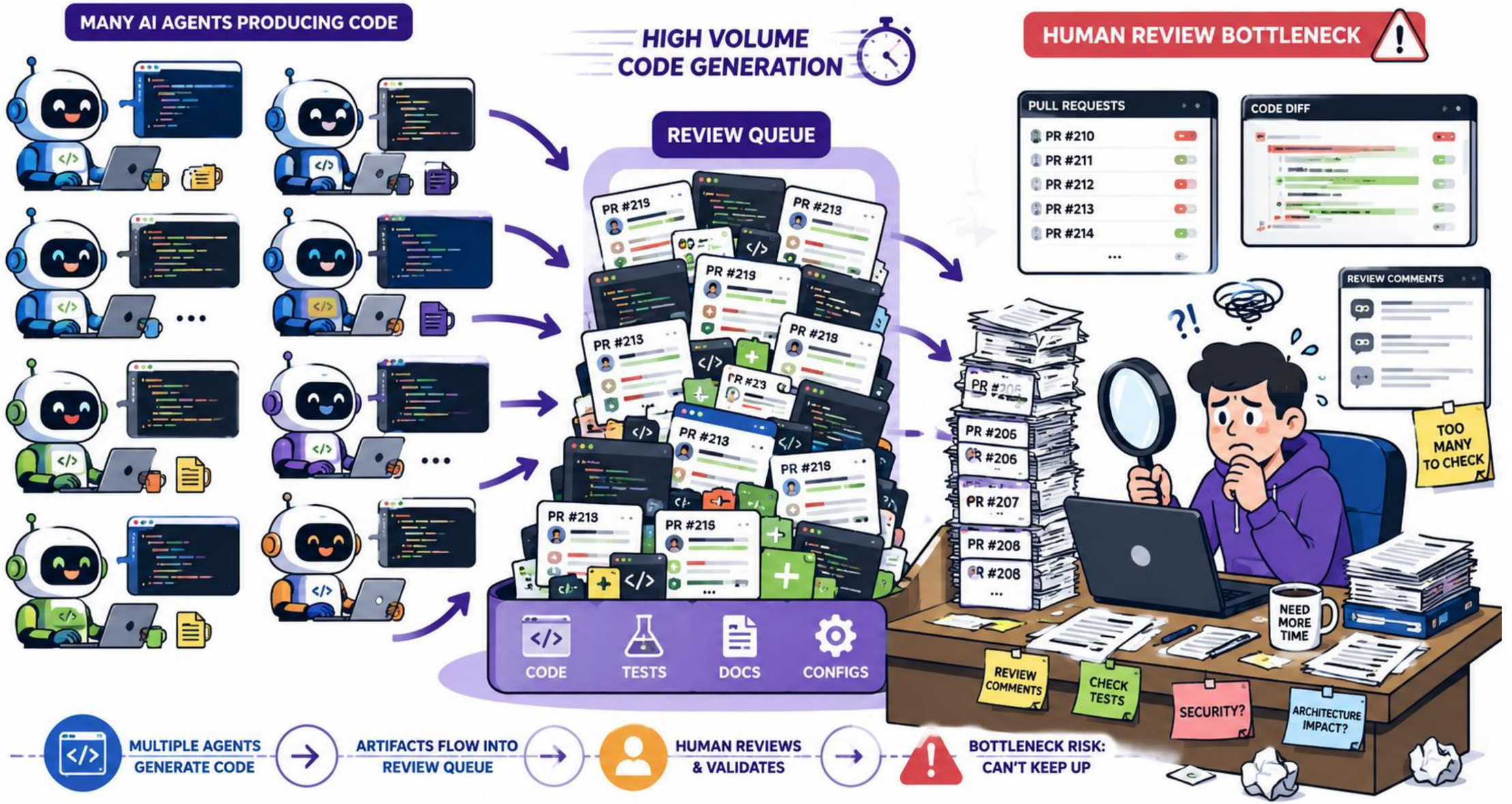
Executable tasks and progress

Reliable build and implementation

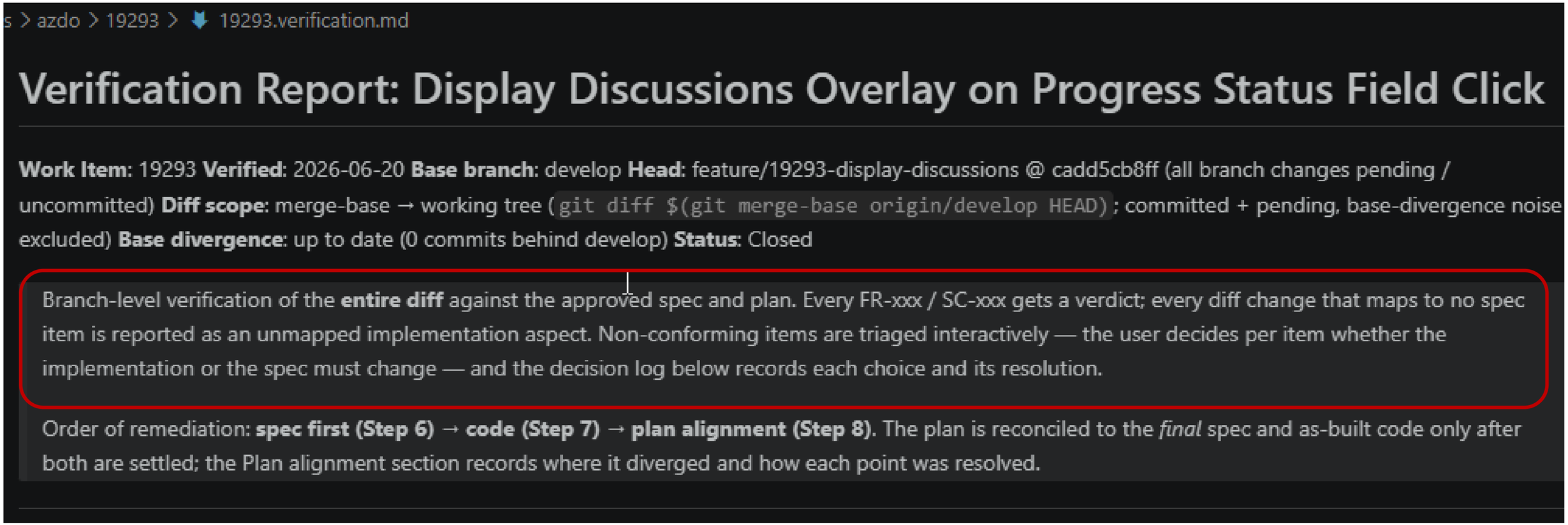
Hope is not a strategy.

Specification and verification are a strategy.

Verifica


MANY AI AGENTS PRODUCING CODE
HIGH VOLUME CODE GENERATION
HUMAN REVIEW BOTTLENECK
REVIEW QUEUE
PULL REQUESTS
PR #210
PR #211
PR #212
PR #213
PR #214
CODE DIFF
REVIEW COMMENTS
TOO MANY TO CHECK
NEED MORE TIME
REVIEW COMMENTS
CHECK TESTS
SECURITY?
ARCHITECTURE IMPACT?
CODE
TESTS
DOCS
CONFIGS
MULTIPLE AGENTS GENERATE CODE
ARTIFACTS FLOW INTO REVIEW QUEUE
HUMAN REVIEWS & VALIDATES
BOTTLENECK RISK: CAN'T KEEP UP

Verifiche automatiche fatte da modelli



Avendo la lista dei requisiti diventa possibile generare prompt specifici che vanno a controllare se il codice scritto nella branch copre realmente tutti i requisiti funzionali approvati

Verifichiamo i requisiti funzionali precedentemente approvati

Spec coverage matrix

Spec item	Verdict	Finding
FR-001 (overlay opens on click, all 6 surfaces)	✅	—
FR-002 (chronological order matches modal)	✅	—
FR-003 (read-only overlay)	⚠️	F-03
FR-004 (FF routing — new vs legacy path)	✅	—
FR-005 (backdrop close)	✅	—
FR-006 (close button)	✅	—
FR-007 (single overlay at a time)	✅	—
FR-008 (second click on open overlay = no effect)	❌	F-02
FR-009 (terminal error / empty state)	✅	—

Prima che un umano vada a controllare il codice scritto, è possibile già capire quali sono le mancanze.

Lo scopo è ritardare il più possibile l’intervento umano e richiederlo quando tutto è già stato verificato da uno o piu agenti / prompt

L’intervento umano è richiesto solo quando tutti i requisiti sono OK

Spec item	Verdict	Finding
FR-001 (enable/disable per task type)	✅	—
FR-002 (reminder list w/ count+unit)	✅	—
FR-003 (async bulk scan on save)	✅	—
FR-004 (create records for eligible tasks)	✅	—
FR-005 (skip tasks w/ existing records)	✅	—
FR-006 (skip closed tasks in bulk)	✅	—
FR-007 (overdue at due+1)	✅	—
FR-008 (reminders, skip past)	✅	—
FR-009 (cancel prior pending on change)	✅	—
FR-010 (cancel on cleared due date)	✅	—
FR-011 (Assignee To, Collaborators Cc)	✅	—
FR-012 (no Assignee → Collaborators primary)	✅	—
FR-013 (no recipients → no email)	✅	—
FR-014 (Close → no email)	✅	—
FR-015 (Close → mark processed)	✅	—
FR-016 (FF off → no processing)	✅	—

Avendo chiarito e verificato inizialmente i requisiti funzionali, la verifica del codice scritto è puntuale e facilmente eseguibile da un agente.

La qualità della Pull Request aperta è solitamente molto alta.

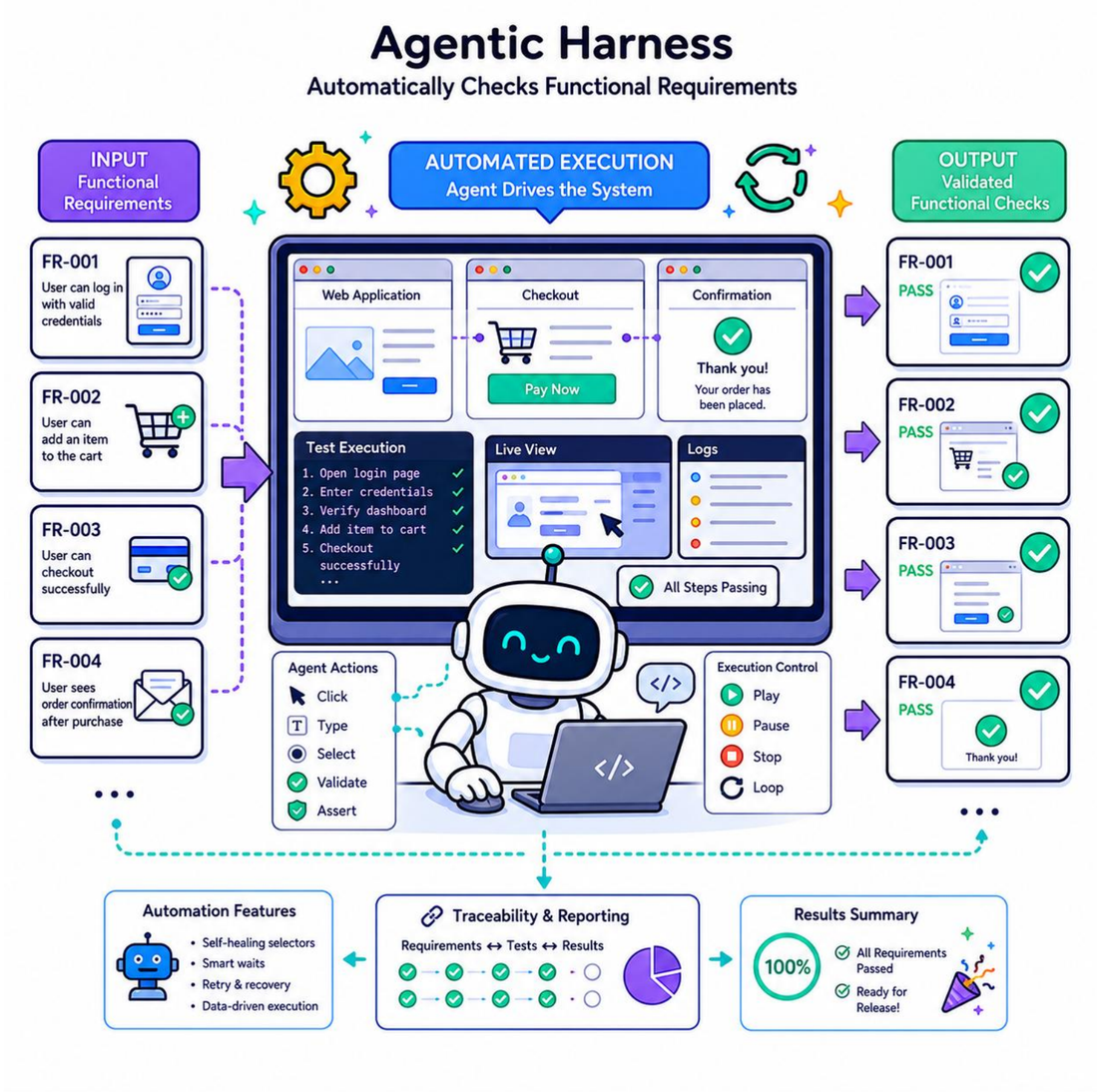
L’integrazione tra Spec Driven Development agentico e Azure DevOps sposta il focus sul **vero valore** del team: an**alisi, design, coerenza, esperienza, qualità e controllo del prodotto**.

Il ciclo iterativo dello SDD, unito a commenti e review su Azure DevOps, **migliora la collaborazione** e rende più strutturato il confronto sulle scelte progettuali.

E' un uso dell'AI che potenzia il contributo umano, non lo sostituisce, non lo debilita o impigrisce (al contrario). Ha **l'obiettivo della qualità, non della velocità**, la velocità è un side-effect.

Oggi al daily mi è tornato alla memoria un motto latino che in questo contesto calza molto: **festina lente**. Affrettati lentamente. E' esattamente quello che questo approccio consente di fare

Verifiche automatiche End to End dei requisiti funzionali



Avendo chiaramente scritto i requisiti funzionali, un agente può interagire con il software completo per verifica

Questo funziona grazie ai modelli vision ed alla integrazione delle app native con il sistema e con i componenti come il browser.

Demo: Verifica tramite app Codex con browser integrato

Cosa abbiamo appreso

La AI non rimuove il software engineering, anzi permette di concentrarsi sulle parti importanti (requisiti)

La capacità di identificare incoerenze e generare testo permette all’utilizzatore di creare User Stories dettagliate in poco tempo.

Rende la scrittura del software un processo più ripetibile

Thank you!

👉 slides & videos: https://www.improove.tech/videos

>>AI CONF 2026

improove