

COME USO L'AI

Dalla domanda al compito

Come lavoro con ChatGPT e con gli agenti di coding,
e quali pattern si possono riusare.

Cristian Ghezzi, 1 ottobre 2026

COSA È CAMBIATO

L'unità di lavoro: dalla domanda al compito

	Prima: la domanda	Ora: il compito
Unità di lavoro	una domanda	un compito: un piano, una feature, una skill
Contesto	incollato a pezzi nella chat	il repository, la knowledge base, le skill
Memoria	la conversazione	i file: piani, knowledge base, AGENTS.md
Il mio ruolo	chiedo e copio la risposta	definisco, critico, verifico, decido

Non chiedo più uno snippet: affido un compito, con un piano, una skill o un prompt-file, e intervengo per decidere.

GLI STRUMENTI

Cosa uso, e dove

BROWSER

ChatGPT

La chat: sistemista passo passo, ricerche, pareri. E consulente per configurare gli agenti.

TERMINALE

Claude Code

L'agente principale: piani, skill, orchestrazione, con un advisor per i passaggi difficili.

TERMINALE E IDE

Codex

Implementa piani e fa review, anche quando è Claude Code a delegargli il lavoro.

IDE

Devin (ex Windsurf)

Agente nell'IDE. Con “fusion” un modello economico esegue; modelli avanzati gratis nelle promozioni.

VS CODE

Cline

Usato poco: soprattutto per provare modelli nei periodi promozionali.

Il piano su file mi permette di passare lo stesso lavoro da uno strumento all'altro e di confrontarli.

PATTERN 1

Il piano su file fa da contratto

1

Un'AI scrive il piano in aiplans/

2

Io lo critico, a volte con una revisione avversaria

3

Una sessione pulita lo implementa

4

Un'altra verifica che codice e piano coincidano

14% → 26%

Workflow a piani: quota delle sessioni → quota delle azioni degli agenti

1% → 9%

Orchestrazione multi-piano: poche sessioni, tantissimo lavoro

PATTERN 2

Costruisco gli strumenti dell'AI

Skill

Quando l'AI sbaglia una procedura, correggo la skill e non il caso singolo. Una libreria unica, gestita con skills-manager.

Knowledge base

Documentazione scritta per gli agenti, non per le persone. Si aggiorna a fine sessione.

Prompt-file

Specifiche in file versionati: si fa il diff, si “eseguono”, si rigenerano.

AGENTS.md globale

Regole comuni a Codex e Claude Code. I prefissi fast e superfast regolano lo sforzo.

“fix all errors ... then amend the skill so that the errors don't repeat next time”

PATTERN 3

Agenti al sicuro, in remoto, di notte



Sandbox

YOLO mode solo in WSL o in container LXD. Chiavi SSH fuori portata, account git dedicati con limiti lato server.



Remoto

Hardware affittato come cloud personale, raggiunto con SSH o con VS Code remoto.



Di notte

airesumer.sh riprende la sessione quando si resetta il limite d'uso: i task lunghi vanno avanti fino al mattino.



Costi

Modello caro come advisor, economico come esecutore. Prompt cache tenuta calda, promozioni sfruttate.

Il rischio da cui parto è la **prompt injection**: un agente autonomo legge testo che non controllo, quindi lo isolo e gli do credenziali limitate. Altro rischio: accesso autonomo alla produzione usando le mie chiavi.

DA PORTARSI VIA

Cinque pattern riutilizzabili

- 1** Il lavoro grosso parte da un piano su file e lo implementa una sessione con il contesto pulito.
- 2** Quando l'AI sbaglia, correggi la skill o la knowledge base e non solo il caso singolo.
- 3** Il contesto si butta, la memoria sta nei file: piani, skill e knowledge base sopravvivono alla sessione.
- 4** Tu definisci, critichi e decidi; l'AI esegue, e un'altra AI verifica.
- 5** Autonomia sì, ma in sandbox con credenziali dedicate e limitate.

Approfondimenti

Dati, numeri e pratiche, per chi vuole i dettagli.

Dove va davvero il lavoro degli agenti

Modalità (agenti)	Sessioni	% sessioni	% azioni
Richiesta diretta di feature/bugfix	85	17%	16%
Workflow a piani	71	14%	26%
Sistemista	50	10%	4%
Setup degli strumenti AI	42	8%	2%
Prompt-file come specifica	36	7%	4%
Diagnosi di errori e log	34	7%	5%
Costruire skill	27	5%	7%
Delega AI → AI	18	4%	3%
Test visuali con l'umano nel ciclo	14	3%	6%
Orchestrazione multi-piano	6	1%	9%
Altre 9 modalità	130	25%	18%

Le righe evidenziate valgono **più in azioni che in sessioni**.

Piani e orchestrazione sono il 15% delle sessioni ma il 35% delle azioni.

APPROFONDIMENTO

Il prompt come codice sorgente

Scrivo

La specifica di un modello 3D, in italiano, in un file *.prompt.txt nel repository.

Eseguo

L'agente la legge e genera lo script Python per Blender: “esegui le istruzioni in ...prompt.txt”.

Confronto

Cambio il prompt, faccio il diff con la versione precedente e rigenero.

Automatizzo

Una skill rigenera in parallelo solo i modelli il cui prompt è più recente dello script.

Pattern: quando una richiesta si ripete, diventa un file versionato e si tratta come codice.

Insegnare una volta sola

Skill

- 27 sessioni per costruirle, 25 per usarle come comandi
- “the skill says X but you didn't do it. Is the skill ambiguous?”
- Si compongono: una skill riusa l'altra invece di duplicarla
- Anche per l'ufficio: riepilogo delle fatture XML, manutenzione del PC
- Una libreria unica gestita con **skills-manager** invece di link simbolici a mano, condivisa tra gli strumenti

Knowledge base per l'AI

- Scritta per gli agenti: niente snippet, niente duplicati, niente prolissità
- Indici gerarchici e un dizionario del codice del framework, mantenuto da una skill
- Quando l'AI sbaglia: “How can we change the knowledge base so that...”
- Il precedente: un GPT personalizzato sulla documentazione del framework (ottobre 2024)

APPROFONDIMENTO

Un'AI che lavora per un'altra AI

6 → 9%

sessioni di orchestrazione → quota delle azioni degli agenti

26

subagent lanciati in una sola sessione, con ruoli implement e review

18

sessioni in cui Claude Code delega review o implementazioni a Codex

Il protocollo, in un solo prompt: leggi lo stato dal README → scegli il piano successivo → crea un branch → un subagent implementa → un altro rivede → aggiorna lo stato → push.

| *“Check if your plan is equivalent to ... tell me the differences stating if your plan is better or worse”*

APPROFONDIMENTO

Sicurezza contro la prompt injection

Un agente in YOLO mode esegue comandi senza chiedere e legge testo che non controllo: pagine web, issue, dipendenze. Do per scontato che prima o poi qualcuno provi a manipolarlo.



Sandbox

YOLO solo dentro WSL o in un container LXD. L'agente vede il suo ambiente e basta; lo raggiungo via tmux, SSH e VNC.



Chiavi protette

Le chiavi SSH personali restano fuori dalla sandbox: l'agente non le può leggere né usare.



Account git dedicati

Gli agenti usano account propri, con restrizioni applicate lato server e non dalla loro buona volontà.

APPROFONDIMENTO

Lavorare in remoto, anche di notte (e in vacanza)

Cloud personale

- Hardware affittato invece di saturare il PC
- Connessione con SSH semplice o con VS Code remoto
- Agenti nei container, raggiunti via tmux; VNC per il browser dei test

Di notte

- **airesumer.sh** controlla quando si resetta il limite d'uso e riprende la sessione dell'agente
- I task complessi, come l'orchestrazione multi-piano, vanno avanti tutta la notte
- Di giorno intervengo solo per decidere

Risparmiare senza rinunciare ai modelli migliori

Advisor

Un modello costoso solo per consigliare, mentre quello normale lavora.

Esecutore economico

Al contrario, “fusion” di Devin: un modello economico esegue.

Limiti d'uso

I comandi nascosti `/low-priority` e `/limit-reset` di Claude Code; i reset dei crediti di ChatGPT.

Promozioni

Modelli avanzati gratis nei periodi promozionali di Devin e Cline.

Prompt cache calda

Riprendo prima che scada; uno script personalizza la riga di comando e mostra lo stato della cache.

Sforzo su misura

fast e superfast, `/effort`, `/model`: il modello grande solo quando serve.

La chat come sistemista

Cosa ci faccio

- Server Linux: web server, application server, database
- Aggiornamenti di sistema, anche di più versioni alla volta
- Chiavi SSH, backup e sincronizzazione verso il cloud
- WSL, container, rete, porte occupate

Come procedo

- Incollo l'output dei comandi e vado avanti un passo alla volta
- Per i lavori lunghi, un file di handoff caricato a ogni ripartenza
- Ramifico la conversazione per provare un'alternativa, invece di proseguire
- Contesto quando sbaglia: “you're wrong, I can run crontab -l but not crontab -e”

Handoff: il contesto si butta, la memoria sta nei file

2,9

prompt miei, in media, per sessione d'agente

33

volte /clear in Claude Code

Un documento di passaggio

- Obiettivo, stato, file, prossimi passi, punti di attenzione
- Con gli agenti: i piani in aiplans/
- Su ChatGPT: l'aggiornamento di un server, su più conversazioni, con un file di handoff a ogni ripartenza
- In questa analisi: agenti → ChatGPT → queste slide, da un handoff all'altro

APPROFONDIMENTO

Lo stesso stile su tutti gli strumenti

Tratto	Come si vede
Regolare lo sforzo	fast e superfast, /effort, modelli “thinking” (321 conversazioni ChatGPT)
Vincoli espliciti	“Do not change code yet”, “wait for my approval”
Contestare	“your analysis is flawed...”, “you're wrong...”
Niente retrocompatibilità inutile	“we are not in production yet”
Lavorare su materiale reale	246 conversazioni ChatGPT con allegati, 130 con immagini
Due lingue	circa 17–18% dei prompt in italiano: dominio e temi personali; inglese per la tecnica

Come è nata questa presentazione

1

Raccolta

Export di ChatGPT, trascrizioni di Codex e Claude Code, un hook che registra i prompt.

2

Classificazione

Ogni sessione riceve una modalità d'uso, con l'aiuto dell'AI.

3

Analisi

Un documento con modalità, esempi anonimizzati e pattern ricorrenti.

4

Handoff

Ogni fase si chiude con un documento di passaggio; la successiva riparte da lì, a contesto pulito.

5

Slide

Un agente genera il deck online; io lo rileggo e chiedo modifiche.

Anche questa presentazione applica i pattern che racconta: dati nei file, handoff tra sessioni, l'AI che esegue e io che decido.