Wolff, Anthropic: “La figura del programmatore è al capolinea”. Come cambierà in futuro lo sviluppo del software

di **Roberto Pezzali** - 01/12/2025 12:01 16

Dopo il lancio del nuovo modello di Anthropic all’interno di Claude Code uno degli ingegneri ammette l’inevitabile: non servirà più scrivere codice. L’ingegneria del software è alle porte di una rivoluzione.

Posta

Le parole di Adam Wolff, sviluppatore software di Anthropic, hanno diviso la community alla fine della scorsa settimana degli sviluppatori. “*Credo che questo nuovo modello di Claude Code sia un assaggio del futuro verso cui stiamo correndo, forse già nella prima metà del prossimo anno: l’ingegneria del software è finita*” ha detto Wolff, aggiungendo che “*Presto, non ci preoccuperemo più di controllare il codice generato per le stesse ragioni per cui non controlliamo l'output di un compilatore.*”

Wolff si riferisce al nuovo Claude Opus 4.5, e quando parla di “finita” in relazione all'ingegneria del software intende dire che la parte meccanica e ripetitiva dell'ingegneria del software è al capolinea: presto sarà l'AI a scrivere codice con la stessa affidabilità con cui i compilatori scrivono codice macchina senza che nessuno controlli quello che fanno.

Nella diatriba che si sta consumando ormai da mesi, ovvero se e quando l'IA sostituirà gli sviluppatori quello che dice Wolff è supportato da una tesi che molti programmatori possono condividere. Secondo l’ingegnere di Anthropic **presto smetteremo di controllare il codice generato dall'intelligenza artificiale per le stesse ragioni per cui oggi non controlliamo l’output di un compilatore**.

Quando uno sviluppatore oggi scrive codice e usa un compilatore come quelli per C, C++, Rust o Java, non verifica mai manualmente l'assembly o il bytecode risultante, ovvero il linguaggio macchina. Questo perché i compilatori sono diventati una tecnologia estremamente affidabile e matura, sono strumenti che applicano trasformazioni matematiche rigorose, validate da decenni di sviluppo, milioni di casi reali e test di regressione continui.

La fiducia è così alta che la possibilità di un errore sistemico del compilatore non rientra più tra le preoccupazioni quotidiane degli sviluppatori. Da molto tempo ormai, l'output del compilatore è considerato un qualcosa che funziona e che nessuno ha bisogno, tranne in rarissimi casi, di toccare.

Wolff sostiene quindi che l’intelligenza artificiale, in particolare con modelli specializzati nella generazione di codice come Claude Code, già dalla metà del prossimo anno si comporterà in modo simile al compilatore che oggi converte i linguaggi “human style” in linguaggio macchina, questo però su un piano più alto: non convertirà codice in codice, ma convertirà specifiche in codice.

La generazione automatica del software verrebbe così vista come una “trasformazione” e non più, come avviene oggi, come un processo creativo dall’esito imprevedibile che potrebbe generare bug. L’idea è che l'AI non si limiterà a produrre frammenti di codice, ma diventerà un sistema capace di generare interi moduli, servizi, API e architetture coerenti, seguendo pattern consolidati e best practice riconosciute. Secondo Wolff per la prima volta con Opus 4.5 si è trovato davanti ad un modello AI che non si limita a generare codice, ma costruisce anche test unitari, test di integrazione, sistemi di verifica logica e controlli di sicurezza. In pratica, una IA che controlla se stessa fino and arrivare a soddisfare le richieste.

Ci sono diversi altri elementi a supporto della tesi di Wolff: l'industria del software si sta muovendo da anni verso architetture molto più uniformi rispetto al passato: framework consolidati, microservizi prevedibili, pattern ricorrenti e librerie estremamente simili tra loro. Questa omogeneità favorisce enormemente l’intelligenza artificiale: più il software reale assomiglia a un insieme di strutture standardizzate, più diventa semplice per un modello generarlo in modo corretto e affidabile. Quando esiste un solo modo “giusto” e diffusamente accettato per costruire un componente, l'AI può replicarlo senza deviazioni.

Se oggi ci preoccupiamo e guardiamo il codice scritto dall'IA per capire se ha fatto bene tra pochi mesi **non ci sarà più bisogno di leggerlo e neppure di preoccuparsi che sia corretto**, perché la sua correttezza deriva dal modo in cui si è arrivati a quel codice: i programmatori smetteranno di verificare l’output esattamente come oggi nessuno apre l’assembly generato dal compilatore.

“*Adoro programmare, e fa un po’ paura pensare che potrebbe non essere più una parte così importante del mio lavoro. Ma scrivere codice è sempre stata la parte facile. La parte difficile sono i requisiti, gli obiettivi, il feedback: capire cosa costruire e verificare se sta funzionando*” conclude Wolff. “*C’è ancora moltissimo da fare, e ci sono tante cose su cui i modelli non sono ancora nemmeno vicini: architettura, progettazione dei sistemi, comprensione degli utenti, coordinamento tra i team. Continuerà a essere divertente e molto interessante ancora per molto tempo*”.

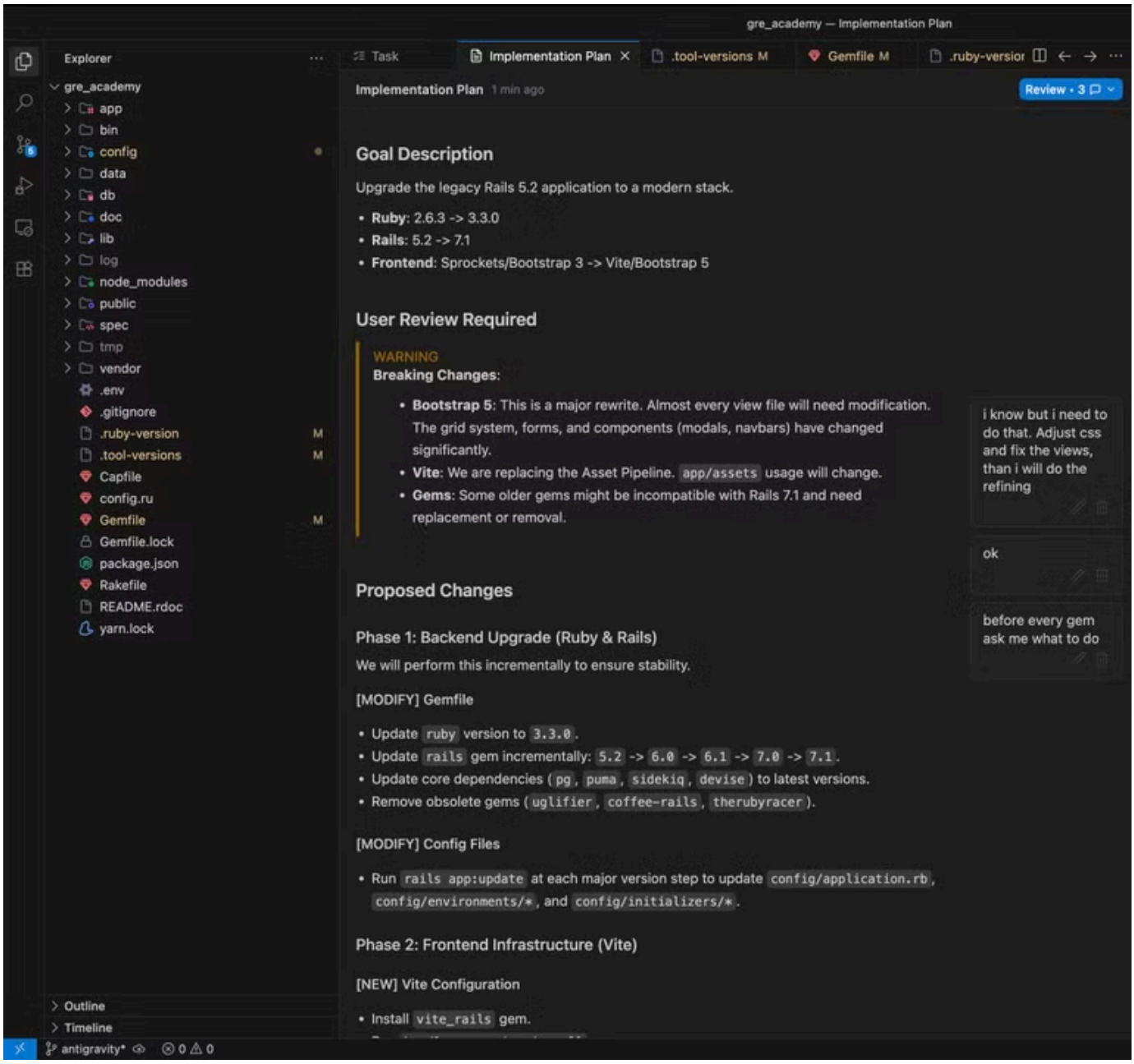
Il livello di astrazione si sposta quindi verso l'alto: gli sviluppatori definiranno obiettivi, requisiti e vincoli e l'AI produrrà il codice in modo affidabile, ripetibile e verificato. **Questa trasformazione non implicherò la scomparsa dei programmatori ma un cambiamento radicale del loro ruolo**: la parte “umana” non risiederà più nella scrittura manuale del codice ma nella capacità di definire problemi, valutare soluzioni, garantire sicurezza e orchestrare sistemi complessi.

Secondo molti, però, questa rivoluzione non sarà indolore perché sono pochissime le figure capaci di trasformarsi da “scribacchini” di codice a quelli che dovrebbero essere architetti o ingegneri del software.

Abbiamo provato Antigravity di Google e in molti casi l'IA è già pronta a sostituire un junior

Negli ultimi giorni abbiamo usato il nuovo editor di codice agentico Antigravity di Google usando proprio Claude come AI, e dobbiamo dire che effettivamente il codice generato dall’AI sta diventando più pulito di quello umano: gli ultimi modelli producono righe più leggibili e più coerenti di uno sviluppatore medio.

Quello che ci ha colpito è il modo in cui funziona bene anche su refactor di grandi dimensioni: ci ha aggiornato intere codebase di progetti legacy creando un piano ben dettagliato e seguendo le nostre indicazioni. Ovviamente non può fare tutto: nel caso di progetti legacy molti componenti non sono più stati aggiornati e vanno sostituiti, e in questo caso serve intervento umano per decidere una strategia: aggiornare il vecchio componente o sostituirlo con uno nuovo?



Ad oggi questa è un po’ l'ancora di salvezza che terrà molti programmatori ancorati al loro lavoro: molti dei software usati dalle grandi aziende sono pieni di librerie e componenti legacy ingestibili da una IA, ci sono in produzione sistemi aziendali con migliaia di patch antiche, dipendenze rotte, assenza di documentazione e di test che rendono più complesso il lavoro dell’IA.

Questo non sposta tuttavia il nodo della questione: oggi gran parte del lavoro fatto da chi scrive fisicamente codice è composto da compiti automatizzabili come boilerplate, conversioni, chiamate API, validazioni, test, refactor meccanici, e **tutte queste attività sono oggi la vera routine di un dev sono perfettamente replicabili dall'AI.**

Se davvero il coding sarà automatizzato e l’ingegneria software avrà sempre maggiore importanza, **in futuro serviranno molti meno programmatori e molti più architetti e manager di progetto**.

Non tutti coloro che scrivono oggi codice e che stanno studiando per diventare programmatori però potrebbero avere le competenze, ma soprattutto l’esperienza, per fare il salto richiesto. Sarà un bel problema.