PEGA

WHITE PAPER

La tassa sui token: perché gli agenti IA autonomi stanno riducendo i margini che avrebbero dovuto proteggere

Guida per COO, CXO e leader operativi

L'opportunità è reale

L'IA ha cambiato lo sviluppo di software: è più veloce, accessibile e competente di quanto non sia mai stata prima.

Gli sviluppatori possono lavorare a una velocità che sembrava incredibile anche solo un anno fa. L'84% degli sviluppatori impiega l'IA per accelerare il proprio lavoro (Studio sugli sviluppatori di Stack Overflow, 2025). Interi prototipi di applicazioni che un tempo richiedevano settimane vengono strutturati in pochi giorni. La democratizzazione dello sviluppo software è reale e sta accelerando. I CIO vedono dove sta portando tutto questo. Entro il 2030 Gartner stima che il 75% del lavoro IT sarà svolto da esseri umani affiancati dall'IA, il 25% sarà eseguito esclusivamente dall'IA, e il lavoro IT privo di IA sarà pari a zero (Gartner, novembre 2025). Non si tratta di una previsione lontana. È l'orizzonte di pianificazione con cui la maggior parte delle organizzazioni IT aziendali sta operando proprio ora.

L'investimento è giustificato e la direzione è quella giusta, ma mettere in produzione più software non si è ancora tradotto nei risultati aziendali trasformativi che i consigli di amministrazione si aspettano. E i leader IT sono stati chiamati in causa per spiegare perché.

Non si tratta di un problema tecnico, ma di un problema di architettura. Al centro della questione vi sono tre domande, e la maggior parte delle aziende non ha ancora risposte chiare a nessuna di esse.

1. ***Lo sviluppo basato sull'IA di fatto ha superato i costi della semplice assunzione di altri ingegneri informatici?***
2. ***Ci si può fidare del codice generato dall'IA per gestire le attività operative aziendali su vasta scala?***
3. ***Come si mantengono la visibilità e il controllo su ciò che l'IA sta creando e mettendo in produzione?***

La domanda sui costi: si paga di più per accumulare più debito?

La promessa dello sviluppo assistito dall'IA era chiara: ridurre i costi e i tempi di sviluppo del software. La realtà, per molte aziende, è che la struttura dei costi è cambiata in modi che non erano previsti nel business case originario.

Scrivere codice non è la parte costosa dello sviluppo software, poiché rappresenta circa il 10% del ciclo di vita totale dell'applicazione. Il restante 90% è costituito da revisione, test, applicazione di patch di sicurezza, debugging, refactoring e manutenzione continua al variare dei requisiti aziendali. Gli assistenti di codice basati sull'IA hanno reso quel 10% più veloce. Ciò che i dati mostrano sempre più chiaramente è che hanno anche reso il restante 90% più costoso.

L'analisi del 2025 di GitClear su 211 milioni di righe di codice nei repository di Google, Microsoft, Meta e altre grandi aziende ha riscontrato un aumento di otto volte dei blocchi di codice duplicati nei progetti assistiti dall'IA rispetto a due anni prima. Il code churn (le righe riviste o scartate entro due settimane dalla scrittura) è raddoppiato dal 2020 al 2024. Come ha affermato il fondatore di GitClear, Bill Harding, il codice generato dall'IA funziona come una nuovissima carta di credito per accumulare debito tecnico: la spesa sembra produttiva al momento, ma il conto arriva più tardi. Il report State of Software Delivery 2025 di Harness evidenzia come la maggior parte degli sviluppatori dedichi più tempo al debugging del codice generato dall'IA rispetto allo sviluppo di nuove funzionalità.

La tassa sui token peggiora la situazione. La produzione di ogni singola riga di codice da parte dell'IA ha un costo in token. A differenza di un token speso per un'interazione con un cliente, un token speso per generare codice applicativo produce una risorsa che richiede investimenti continui per la manutenzione. Se tale risorsa accumula debito più velocemente di quanto produca valore, l'impatto economico si aggrava nella direzione sbagliata fin dal primo giorno.

E la struttura dei costi degli stessi strumenti di sviluppo dell'IA non è più prevedibile. GitHub Copilot è passato alla fatturazione a consumo nel giugno 2026, sostituendo il modello a tariffa fissa basato su unità di richiesta premium con crediti IA basati su token. GitHub stessa ha ammesso che con il vecchio modello una rapida domanda in chat e una sessione di programmazione autonoma di più ore possono costare all'utente la stessa cifra e che assorbire i crescenti costi dell'inferenza "non è più sostenibile" (blog di GitHub, aprile 2026). Le imprese che avevano preventivato i costi di sviluppo dell'IA sulla base di tariffe fisse operano ora in un ambiente a costi variabili, con minore visibilità su quanto spendono per singolo flusso di lavoro.

La vera domanda non è se valga la pena utilizzare gli strumenti di sviluppo basati sull'IA. Questo è indubbio. La domanda è se l'approccio attuale, ovvero utilizzare l'IA per generare più codice più velocemente, stia effettivamente riducendo i costi totali di sviluppo o se stia semplicemente anticipando la velocità di rilascio, aumentando al contempo gli oneri di manutenzione che si presenteranno nel budget del prossimo anno.

La domanda su prestazioni e sicurezza: ci si può fidare di ciò che l'IA crea?

Un rilascio più rapido crea valore solo se ci si può fidare di ciò che viene prodotto per essere eseguito su scala aziendale. In questo senso, le prove stanno fornendo ai leader IT legittimi motivi di cautela.

L'analisi di CodeRabbit su 470 pull request reali ha rilevato che il codice generato dall'IA introduce 1,7 volte più problemi totali rispetto al codice scritto da esseri umani e, nello specifico, 2,74 volte più vulnerabilità di sicurezza. (CodeRabbit State of AI vs. Human Code Generation, dicembre 2025). Una ricerca IEEE presentata all'ISSRE 2025 ha rilevato che il codice generato dall'IA presenta tassi più elevati di vulnerabilità di sicurezza ad alto rischio nonostante produca risultati strutturalmente più semplici, il che suggerisce che il rischio risieda nel modo in cui la logica viene concepita, non solo nella sua complessità.

Gli strumenti di IA generano soluzioni a problemi locali e immediati. Non hanno piena consapevolezza dell'architettura più ampia del sistema, delle astrazioni interne, delle utilità condivise o dei vincoli di sicurezza che regolano il modo in cui i dati si muovono all'interno dell'ambiente. Il risultato è un codice che supera i test delle unità e passa la revisione iniziale, ma introduce vulnerabilità che emergono solo in casi limite o sotto carico di produzione.

Il 76% degli sviluppatori che utilizzano strumenti di programmazione basati sull'IA dichiara di aver generato codice che non comprendeva appieno, almeno in alcune occasioni (Stack Overflow, 2026). Questa statistica ha un peso operativo reale. Quando a un CIO viene chiesto da un organismo di regolamentazione, da un team legale o dal consiglio di amministrazione perché si sia verificato un determinato comportamento del sistema, rispondere che a generarlo è stata l'IA e che lo sviluppatore non era sicuro di come funzionasse non è accettabile. Eppure questa è la posizione implicita di responsabilità di qualsiasi organizzazione che distribuisca codice generato dall'IA in produzione senza un livello di governance strutturale.

Le principali previsioni di Gartner per il 2026 includono una stima secondo cui le cause legali globali per incidenti di tipo "death by AI" supereranno quota 2.000, a causa di una governance insufficiente sui risultati generati dall'IA (Gartner, ottobre 2025). Non si tratta di rischi futuri ipotetici, ma di un'esposizione attiva per le imprese che oggi assegnano a un codice generato dall'IA una significativa autorità sui flussi di lavoro rivolti ai clienti o regolamentati.

La domanda sull'affidabilità: sappiamo cosa sta sviluppando l'IA?

La velocità e la sicurezza sono problemi operativi. La verificabilità è un problema strutturale e per i CIO e gli architect aziendali potrebbe essere il più difficile da integrare a posteriori.

Quando un'applicazione prende una decisione, che si tratti di indirizzare una pratica, determinare l'idoneità, concedere un accesso o generare un risultato visibile al cliente, responsabile di tale decisione è l'IT, non l'IA che ha scritto il codice. Il reparto IT ha distribuito l'applicazione e i leader l'hanno approvata.

In un ambiente di sviluppo tradizionale, tale responsabilità è gestibile poiché la logica è ispezionabile. Quando un auditor chiede perché sia stata presa una decisione, uno sviluppatore traccia il percorso del codice pertinente e trova la risposta. In un ambiente in cui la logica applicativa è stata generata dall'IA, tale tracciabilità non è garantita. Il codice generato dall'IA produce risultati funzionanti, ma la logica alla base di specifiche scelte di implementazione non è documentata come avviene nella progettazione intenzionale umana. Quando l'auditor chiede perché il sistema abbia preso una determinata decisione in un caso limite, rispondere che l'IA lo ha scritto in quel modo non è una posizione motivabile.

Questo è il problema di affidabilità che conta di più a livello aziendale. Non si tratta di stabilire se il codice funzioni correttamente in media, ma di poter spiegare qualsiasi risultato specifico a chiunque lo chieda, in qualsiasi momento futuro, con un linguaggio comprensibile e fruibile.

La logica applicativa visiva e dichiarativa risolve questo problema in modo strutturale. Quando le regole aziendali, le tabelle decisionali e l'instradamento dei flussi di lavoro sono espressi come configurazioni ispezionabili anziché come codice generato, ogni percorso decisionale diventa tracciabile senza dover ricorrere a un'analisi forense. La logica risiede in un livello che possono leggere anche i team di conformità, i team legali e gli stakeholder aziendali, non solo i tecnici. La revisione diventa una verifica della configurazione, anziché un esercizio di archeologia del codice.

L'architettura che risponde a tutte e tre le domande

La svolta architetturale che risolve contemporaneamente i problemi di costi, sicurezza e affidabilità non rappresenta un limite all'uso dell'IA, ma comporta il reindirizzamento del livello dello stack di sviluppo in cui l'IA viene applicata.

L'IA è davvero efficace nel velocizzare la generazione di strutture applicative di alto livello: modelli di dati, definizioni di flussi di lavoro, tabelle decisionali e logiche di instradamento espresse in forma dichiarativa. Questi sono artefatti che descrivono ciò che un'applicazione fa senza codificare l'implementazione in codice imperativo. Sono leggibili, verificabili e gestibili come non lo è il codice generato. Quando i requisiti aziendali cambiano, il modello cambia e il comportamento dell'applicazione si adegua di conseguenza, senza le modifiche al codice a cascata e i rischi di regressione che tali cambiamenti introducono.

Questo è l'approccio di sviluppo incentrato sul modello nella pratica. L'IA svolge il lavoro in cui riesce davvero meglio: tradurre l'intento in struttura, generare la configurazione a partire dal linguaggio naturale e velocizzare la definizione dell'architettura. Il risultato è un modello, non del codice. La generazione di codice, laddove necessaria, avviene al di sotto di esso come dettaglio implementativo e non come artefatto principale.

Le implicazioni operative sono concrete. I team di sviluppo dedicano meno tempo alla manutenzione perché la superficie di manutenzione è il modello, non la base di codice. La revisione della sicurezza è più gestibile perché la logica aziendale è espressa in modo dichiarativo. Le verifiche di conformità sono più veloci perché il livello decisionale è ispezionabile senza il supporto dei tecnici. E la spesa per i token si concentra sulla parte a più alto valore dello sviluppo: definire che cosa l'applicazione debba fare, invece di generare i meccanismi di come lo fa.

Secondo il sondaggio I&O di Gartner della fine del 2025, solo il 28% dei casi d'uso delle infrastrutture IA soddisfa pienamente le aspettative di ROI. Il divario tra aspettativa e realtà non è un fallimento delle capacità, ma un problema di architettura. Le aziende che ottengono un ROI dall'IA nello sviluppo non utilizzano necessariamente modelli più avanzati. Utilizzano l'IA per creare artefatti che non si deprezzano al ritmo del codice generato.

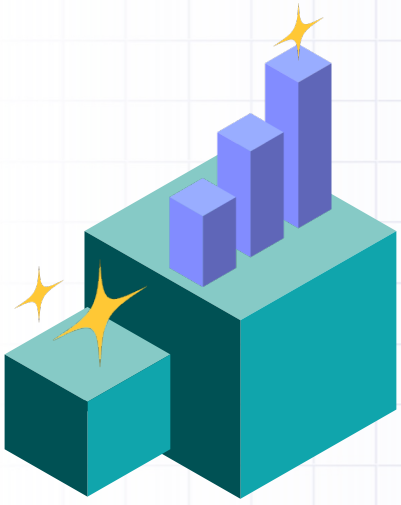
La decisione circa l'architettura

La scelta che i leader IT devono compiere non è tra velocità e sicurezza, anzi è proprio questo tipo di impostazione a far sembrare la situazione attuale un vincolo impossibile da superare. La vera scelta è tra due diverse architetture con profili di rischio e costi a lungo termine profondamente differenti.

Un'architettura incentrata sul codice generato dall'IA offre una rapida produzione iniziale a fronte di una coda di manutenzione che si allunga, una verificabilità limitata e un'esposizione a problemi di sicurezza che cresce con la base di codice. Un'architettura incentrata sullo sviluppo basato sul modello con IA integrata offre una produzione iniziale che è verificabile fin dalla progettazione, gestibile a livello di modello e genera una spesa per i token concentrata sulla parte dello sviluppo che crea effettivamente valore duraturo.

I leader IT non devono dire di no alla richiesta di un rilascio accelerato dall'IA. La risposta non è andare più lenti. La risposta è una filosofia di sviluppo diversa che l'azienda possa sostenere e l'IT possa motivare.

Il giusto investimento nell'IA non consiste nello scrivere codice più velocemente, ma nel creare modelli applicativi che non hanno bisogno di essere riscritti.



Ringraziamento

Informazioni su Pega

Pega fornisce la principale piattaforma basata sull'IA per la trasformazione aziendale. Le organizzazioni più influenti al mondo si affidano alla nostra tecnologia per reinventare il modo in cui viene svolto il lavoro automatizzandone i flussi, personalizzando le customer experience e modernizzando i sistemi legacy. Dal 1983, la nostra architettura scalabile e flessibile promuove l'innovazione continua, aiutando i clienti ad accelerare il loro percorso verso l'impresa autonoma.
PEGA.COM/IT