INSTANT PAPER

Copilots for Coding

Un’esplorazione dell’impatto dell’AI Generativa nel Software Development.

Autori

Massimo Del Vecchio
Eng Modernize
Executive Director

ENGINEERING
Massimo.DelVecchio@eng.it
in Massimo Del Vecchio

Dario Fabbri
Modernization Factory
Senior Technical Manager

ENGINEERING
Dario.Fabbri@eng.it
in Dario Fabbri

Elena Marchisa
Strategic Marketing
& Content Senior Specialist

ENGINEERING
Elena.Marchisa@eng.it
in Elena Marchisa

00

Summario

01 / **La GenAI nel Software Development** 2
02 / **Il nostro approccio alla sperimentazione di GitHub Copilot** 3
03 / **L'aumento della produttività** 6
04 / **Il miglioramento della qualità del software** 9
05 / **Verso una più ampia adozione** 12
06 / **Perchè scegliere noi** 13

01 La GenAI nel Software Development

L'**Intelligenza Artificiale Generativa** (GenAI) consente di creare contenuti - codice, immagini, audio e video - che non solo replicano, ma spesso superano le capacità umane in termini di velocità e precisione. Le aziende hanno trovato nell'AI generativa un potente strumento per ottimizzare e migliorare l'efficienza in diverse aree aziendali.

Nel campo dello **sviluppo software**, l'AI generativa sta rivoluzionando l'approccio tradizionale al codice.
Da semplici strumenti che forniscono suggerimenti fino alla creazione di contenuti e codici, questa tecnologia sta ridefinendo ogni fase del ciclo di vita del software: raccolta dei requisiti, progettazione, sviluppo, test e distribuzione, nonché produzione di documentazione.

Anche se siamo solo all'inizio, il futuro sembra promettente. Con la continua evoluzione di questa tecnologia, un numero sempre maggiore di software developer si rivolge agli strumenti di GenAI per **snellire il processo di produzione, aumentare la produttività e migliorare la qualità** del codice. Tra gli strumenti più utilizzati ci sono quelli che affiancano chi sviluppa software, come GitHub Copilot, basati sulla tecnologia dei Large Language Models. Questi sistemi agiscono come assistenti intelligenti, in grado di generare codice, suggerire miglioramenti e persino automatizzare intere fasi del processo di produzione.
Questo paper presenta la sperimentazione sull'adozione di **GitHub Copilot** che abbiamo condotto coinvolgendo diversi team di sviluppo in progetti aziendali reali. I nostri risultati dimostrano chiaramente sul campo i benefici che la GenAI sta apportando nello sviluppo del software. Il successo dell'integrazione della GenAI negli ambienti di sviluppo segna uno step significativo nella trasformazione digitale in corso, sfruttando le peculiarità della collaborazione uomo-macchina per creare soluzioni software più efficienti, scalabili e innovative.

02 Il nostro approccio alla sperimentazione di GitHub Copilot

GitHub Copilot fornisce assistenza contestualizzata durante tutto il ciclo di vita dello sviluppo del software, fornendo supporto in termini di completamento automatico del codice, assistenza via chat direttamente nell'IDE (Integrated Development Environment), spiegazioni sul codice in lavorazione e risposte alle domande di ogni software developer.
Abbiamo scelto di utilizzare Copilot per la nostra sperimentazione in ragione della nostra forte partnership con Microsoft e del fatto che, a gennaio 2024, era l'unico assistente AI in grado di integrare il servizio di chat negli IDE più adottati (Jetbrains e Visual Studio).

Il nostro viaggio con GitHub Copilot è iniziato con un obiettivo chiaro: **comprendere l'impatto effettivo dell'AI sulla produttività** delle persone che sviluppano il software **e sulla qualità** di esso. A differenza degli studi di laboratorio, che spesso non riescono a cogliere le sfumature dei progetti reali, abbiamo scelto di condurre la nostra sperimentazione in condizioni autentiche di business-as-usual, selezionando in modo random alcuni tra i diversi progetti in corso e definendo un approccio semplice, ma efficace per il contesto di utilizzo.
Abbiamo innanzitutto definito la nostra **strategia** per raccogliere dati e misurare la produttività.

Copilot adoption

TRIAL POPULATION

9 MESI

70+ SOFTWARE DEVELOPER

10+ TEAM

Tipo di progetto

- Green field
- Brown field
- Application maintenance
- Portfolio handovers
- Modernization
- Security & quality remediations

Tecnologie

- Java legacy
- Java microservices (Spring Boot / Quarkus)
- .NET legacy
- .NET microservices
- Angular / Typescript
- ...

Livello di seniority

- Technical managers
- Team leaders
- Architects
- Senior developers
- Junior developers

In qualità di system integrator, le nostre metodologie di misurazione della produttività e dello sviluppo del software possono essere molto diverse tra i vari clienti e contesti: siamo in presenza di una situazione molto variegata (scattering) che può includere misure quali Function Point, Story Point, Lines of Code ed in certi casi l'assenza stessa di un quadro di misurazione.
Abbiamo sviluppato un **approccio standard cross-team** che ci permette di testare i benefici della GenAI nello sviluppo del software in qualsiasi contesto aziendale.
Si articola in tre fasi successive:

1. Categorizzazione delle attività di sviluppo
Viene creata una tassonomia condivisa dei tipi di attività per allineare tutto il gruppo con definizioni coerenti.

2. Stima manuale dei tempi
Prima di scrivere il codice, si chiede ad ogni software developer di stimare il tempo (in ore-persona) necessario per svolgere attività specifiche.
È richiesta la misurazione di attività di piccole/medie dimensioni (ore, non giorni) per ridurre le distorsioni e gli errori di stima.

3. Confronto e regolazione
Il tempo stimato è poi confrontato con il tempo realmente impiegato per le attività svolte con Copilot.

Può essere necessario applicare un fattore di correzione per rettificare gli imprevisti sulle stime iniziali, anche attraverso interviste dirette con ogni software developer.

In riferimento alla categorizzazione delle attività di sviluppo, nella nostra sperimentazione abbiamo suddiviso i compiti in aree specifiche, pensate per far emergere i punti di forza di Copilot:

- **Quick and dirty feature development**: la creazione della versione iniziale di una funzionalità per soddisfare i requisiti funzionali di base. Non sono inclusi commenti, dichiarazioni di log, test automatici e ottimizzazioni;
- **Code analysis**: tutte le attività relative alla comprensione delle codebase esistenti;
- **Refactoring**: la ristrutturazione o la riscrittura del codice per migliorare le prestazioni, la qualità, la leggibilità e la manutenibilità senza alterare la funzionalità;
- **Logging**: l'aggiunta o il miglioramento delle istruzioni di logging al codice esistente per il monitoraggio del sistema e per favorire la risoluzione dei problemi;
- **Test automation**: la creazione o il miglioramento di qualsiasi forma di test automatizzato, senza distinzione tra quelli unitari, di integrazione, end-to-end, etc;
- **Technical documentation**: la predisposizione di note tecniche estratte da porzioni di codice, al fine di accelerare i processi di documentazione;
- **Comments**: la scrittura o il miglioramento dei commenti del codice sorgente per migliorare la manutenibilità;
- **Hotfix**: l'attività di correzione di bug nel codice rilasciato in tempi stretti.

Lo studio è stato suddiviso in **tre wave** di tre mesi ciascuna, a partire da gennaio 2024, per avere un tempo sufficientemente ampio per raccogliere dati significativi.

Prima di entrare nella terza wave, abbiamo deciso di effettuare un'**ampia rimodulazione** della popolazione della sperimentazione. Questo ci ha permesso anche di rivalutare la curva di apprendimento di ogni software developer e di verificare i dati raccolti nella seconda wave.

Questo processo mirava a monitorare l'onboarding di Copilot progetto per progetto, per massimizzare l'efficacia dell'investimento.

03 L'aumento della produttività

I dati raccolti hanno dimostrato l'efficacia di Copilot in diversi task, con miglioramenti misurabili sia nella produttività che nella qualità. Di seguito presentiamo i risultati della **prima** (gennaio, febbraio e marzo 2024) e della **seconda wave** (aprile, maggio e giugno 2024). I dati della terza wave sono ancora in fase di consolidamento alla data di stesura di questo paper.

Ogni stima di attività raccolta durante la wave è stata denominata "Data Point". L'aumento di produttività stimato è stato calcolato sotto forma di percentuale adottando la seguente formula: (tempo stimato senza copilota - tempo effettivo con copilota) / (tempo stimato senza copilota) * 100.

Gli aumenti di produttività sono stati tracciati su una distribuzione di Gauss per tutte le categorie di attività di sviluppo. I grafici seguenti mostrano la distribuzione effettiva, dove:

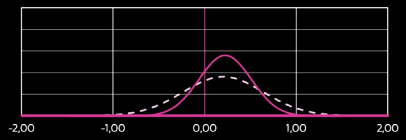
- il valore **Data Points** indica il numero di osservazioni (singole attività di sviluppo) per la wave di riferimento (prima o seconda);
- il valore "**μ**" rappresenta il **valore medio** della metrica durante il periodo di osservazione;
- il valore "**σ**" rappresenta la **deviazione standard**. Un σ più alto indica una maggiore variabilità, ovvero quanto i Data Points si allontanano dal valore medio;
- le curve mostrate nel grafico sono una distribuzione di probabilità dei dati intorno alla media. La **linea tratteggiata** rappresenta i valori della prima wave e la **linea continua** quelli della seconda.

Ad esempio: il valore medio del 22,28% nel grafico del Quick & Dirty Features Development significa che il tempo necessario per svolgere questo tipo di attività utilizzando Copilot è inferiore del 22,28% rispetto a quello necessario senza di esso (con una deviazione standard dalla media del 28,46%).

Risultati di produttività della prima e della seconda wave

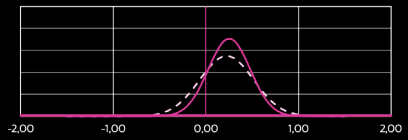
Quick & dirty feature dev

Data points	55	**102**
μ	20.15%	▲ **22.28%**
σ	43.96%	▲▲ **28.46%**



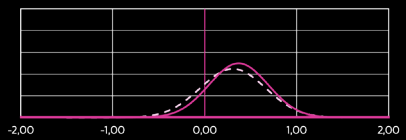
Refactoring

Data points	29	**54**
μ	22.58%	▲ **25.36%**
σ	29.08%	▲ **22.60%**



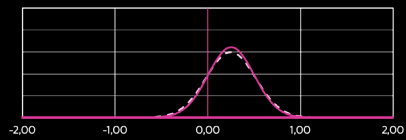
Test automation

Data points	20	**34**
μ	30.63%	▲ **36.83%**
σ	35.44%	▲ **31.97%**



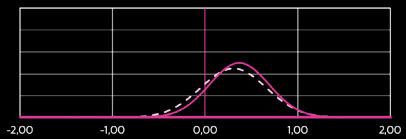
Comments

Data points	19	**30**
μ	25.00%	≈ **25.36%**
σ	26.69%	▲ **24.76%**



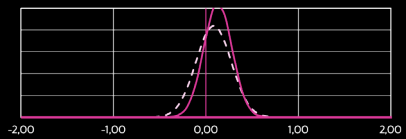
HotFix

Data points	4	**24**
μ	16.67%	▼ **11.06%**
σ	34.74%	▲ **14.03%**



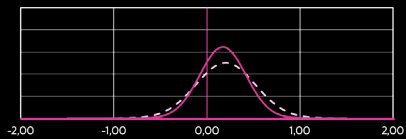
Code analysis

Data points	8	**9**
μ	8.33%	▲ **12.96%**
σ	19.01%	▲ **15.20%**



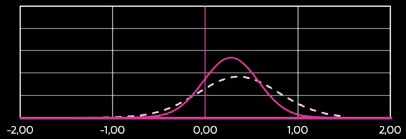
Logging

Data points	5	**7**
μ	20.00%	▼ **17.01%**
σ	31.84%	▲ **24.72%**



Technical documentation

Data points	5	**7**
μ	36.19%	▼ **27.89%**
σ	43.72%	▲ **29.67%**



▲▼ : miglioramento o peggioramento del valore

Durante la seconda wave, la deviazione standard si è ridotta di circa il **10%** in tutte le categorie di attività di sviluppo. Questo dato può essere il risultato sia della curva di apprendimento del gruppo di software developer nell'adozione dello strumento, sia del miglioramento della loro capacità di stima delle attività.

Tra i miglioramenti segnalati, il **7-10%** si riferisce direttamente a un aumento della produttività, grazie ai suggerimenti e alle spiegazioni in tempo reale di Copilot (ad esempio: i commenti sul codice sono stati classificati come miglioramenti della qualità, non della produttività). Questo aumento potrebbe sembrare abbastanza grande, ma si riferisce solo alle ore realmente dedicate al coding durante l'intero ciclo di vita del progetto IT.

La giornata lavorativa di ogni software developer, infatti, comprende diverse attività non legate al coding, come riunioni, revisione di documenti e formazione. Il coding non supera quasi mai il 70% del tempo di lavoro effettivo e rappresenta solo il 30-40% dell'impegno totale in un tipico progetto IT. Infine, nella nostra sperimentazione abbiamo analizzato un sottoinsieme molto ampio, ma non completo, di attività di sviluppo (ad esempio, mancano la definizione e l'implementazione dell'architettura generale, la suddivisione dei compiti, ecc...).

In sintesi, GitHub Copilot supporta l'operatore umano solo quando ha le mani sulla tastiera e l'IDE aperto, quindi l'impatto complessivo sulle prestazioni del progetto deve essere calcolato su tutto lo stack di consegna e richiede ulteriori approfondimenti.

Infine, va menzionato un esperimento parallelo, condotto durante la prima e la seconda wave, in cui alcuni team sono stati coinvolti anche in attività di remediation a problemi rilevati dallo strumento di analisi statica del codice SonarQube. L'uso di Copilot per le correzioni ha portato un **aumento medio della produttività del 40%**, mettendo in evidenza il **grande potenziale dell'automazione completa di compiti ripetitivi** e a basso livello di creatività nelle attività di coding quotidiane.

04 Il miglioramento della qualità del software

Copilot ha facilitato l'aderenza alle best practice, ha consentito di ridurre i bug e di migliorare la manutenibilità del codice, portando così un **miglioramento di circa il 10%** anche nella qualità del software.
Le metriche fornite da GitHub Copilot (ultimo utilizzo, numero di suggerimenti proposti e accettati, ...) si sono rivelate molto utili, ma abbiamo anche condotto un sondaggio diretto con il gruppo di software developer coinvolto nella sperimentazione per ulteriori approfondimenti.

In generale, **c'è stata una risposta positiva all'introduzione di GitHub Copilot**. Le persone che operano nel campo dello sviluppo software tendono a essere appassionate di tecnologia. Per questo motivo la maggior parte del team si è dimostrata desiderosa di provare l'AI Generativa, ritenendo che avrà un ruolo trasformativo nell'evoluzione della professione.
I profili **junior** hanno sottolineato come Copilot fatichi a identificare le allucinazioni e possa talvolta fornire suggerimenti insufficienti. I profili **senior** hanno evidenziato

la necessità di possedere una significativa esperienza nel ruolo per essere in grado di valutare meglio i suggerimenti proposti dal Copilot, ma riconoscono come questo assistente possa dimostrarsi molto efficace se sfruttato tramite prompt specifici e ben formulati nella chat.

- **Code Analysis** ★★★★★
- **Refactoring** ★★★★
- **Logging** ★★★
- **Test Automation** ★★★★
- **Technical Documentation** ★★★★★
- **Comments** ★★★★
- **Hotfixes** ★★★★

GitHub Copilot è promettente nella sua capacità di trasformare le pratiche di sviluppo del software, ma la sua utilità varia a seconda del compito da svolgere.

In particolare, la nostra sperimentazione ha evidenziato alcuni suoi limiti attuali.

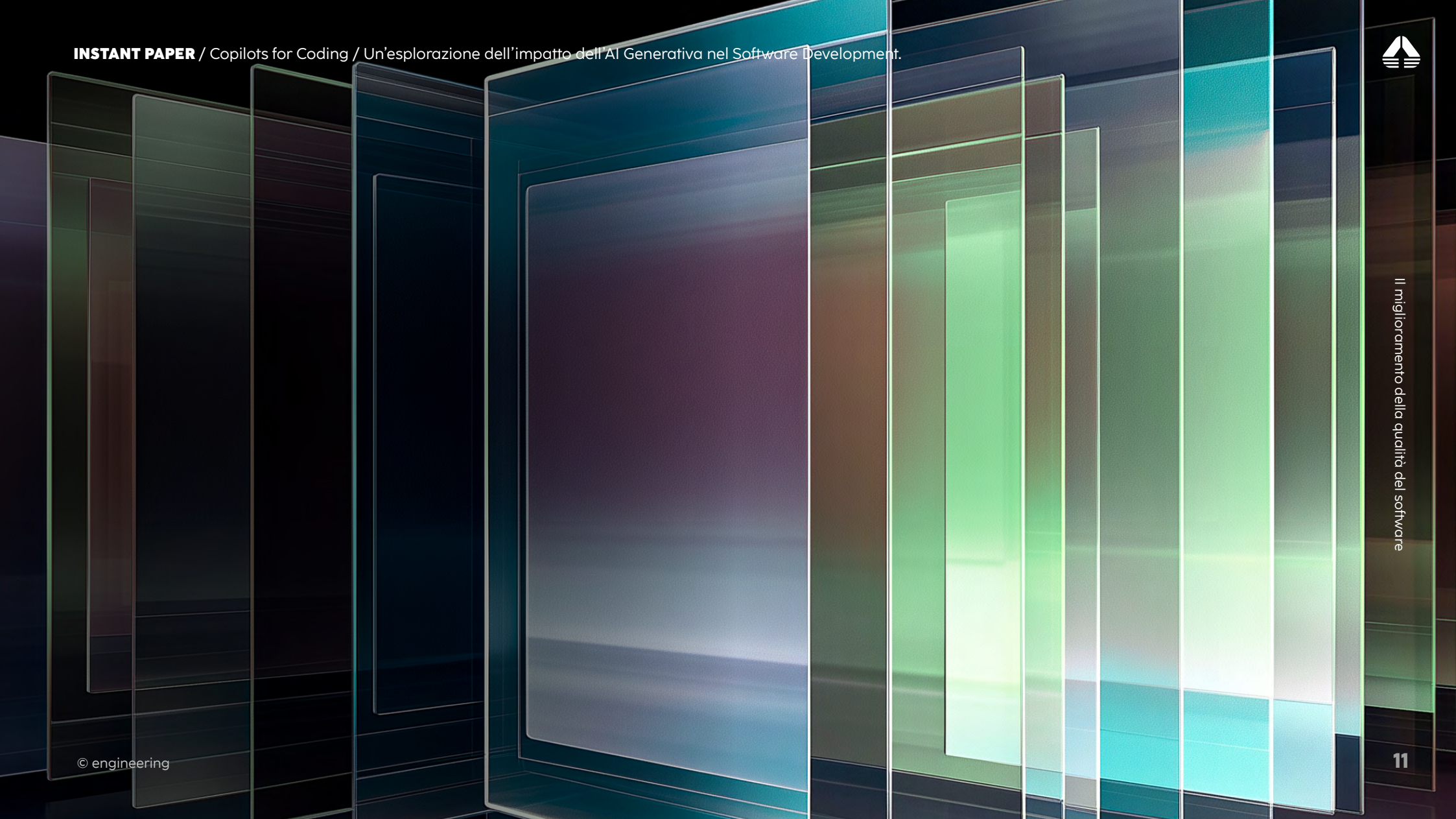
Quando si chiede di generare **documentazione** per le codebase esistenti, Copilot produce una documentazione semplice e ben strutturata, ma le sue prestazioni diminuiscono con il codice legacy più complesso o mal scritto, in cui fatica a dedurre l'intento o a identificare logiche più articolate.

I test di **traduzione di codice** da linguaggi molto vecchi, come il COBOL, a quelli moderni hanno dato risultati insoddisfacenti: ad oggi, lo strumento ha difficoltà con il COBOL a causa della sua verbosità, della natura procedurale e del contesto altamente specifico del dominio, producendo traduzioni poco accurate. Allo stesso modo, per framework come Spring Boot, anche se può aiutare nel refactoring o nella configurazione di base, la sua comprensione dei modelli architetturali più profondi e delle best practice specifiche del framework è limitata.

Per quanto riguarda il COBOL e la documentazione del codice, come Gruppo Engineering possiamo invece vantare un progetto di successo realizzato utilizzando il nostro Assistente di Private GenAI, EngGPT.

Tornando alle **sfide di contesto**, la mancanza di comprensione dello scenario specifico del dominio da parte del Copilot, unita alla complessità intrinseca delle vecchie tecnologie, limita la sua capacità di fornire assistenza completa nei progetti di modernizzazione.

Queste difficoltà evidenziano la necessità di un **significativo coinvolgimento umano** in questi sforzi di traduzione e modernizzazione e la centralità della figura di personale esperto nel guidare la trasformazione dei sistemi legacy. Sebbene il futuro sia promettente, considerando il clamore del mercato e gli investimenti su larga scala previsti nel campo della "GenAI for coding", attualmente le attività di sviluppo in copilottaggio con la GenAI apportano miglioramenti in compiti ben definiti e ripetitivi.



05 Verso una più ampia adozione

La nostra sperimentazione suggerisce che il Copilot funzioni **meglio se affiancato** a personale esperto che possa fornire il contesto e la competenza che attualmente mancano all'assistente, **aumentando al contempo la velocità delle attività ripetitive** per ogni software developer.
L'AI Generativa è sicuramente uno strumento che libera del tempo, permettendo al personale umano di concentrarsi su attività di maggior valore.

Per ora, il Copilot può supportare le **analisi preliminari** o le attività a basso rischio nell'ambito della modernizzazione del codice legacy, ma a nostro avviso non è ancora pronto a sostituire completamente il lavoro manuale in questi ambiti. Ulteriori aree di indagine potrebbero riguardare il **code churn**, combinando le metriche di Copilot con i dati di GitHub. Sarebbe necessario anche esplorare il supporto di questo assistente nella raccolta dei requisiti, nella documentazione del software, nel reverse engineering e nelle conversioni (semi)automatiche.

Man mano che l'utilizzo della GenAI nelle attività di coding aumenterà, potrebbero emergere anche miglioramenti nella sua **capacità di gestire il codice legacy** e le **attività complesse di documentazione**, in particolare se le future iterazioni saranno addestrate su **set di dati più ampi e diversificati** che includano vecchi paradigmi e framework di programmazione.

Allo stato attuale, non crediamo che l'obiettivo dell'adozione di GenAI sia quello di produrre il più velocemente possibile grandi quantità di codice non controllato. Ad oggi, pensiamo che la GenAI possa portare ad un significativo miglioramento della qualità (unit testing, commenti, refactoring, tech doc, ...) più che della produttività in sé.

In ogni caso, riteniamo che questo sia solo l'inizio di una trasformazione molto più ampia, guidata da investimenti significativi nelle tecnologie di GenAI in tutto il mercato IT. In uno scenario caratterizzato da un'abbondanza di proof of concept e di ricerche sull'AI generativa, l'uso di nuove tecnologie in **progetti aziendali reali** ci ha permesso di identificare chiaramente la strada da seguire per avere un impatto reale nel mondo in cui viviamo e lavoriamo.

Quest'ultima considerazione ci ha portato a prendere l'audace decisione di **distribuire GitHub Copilot a ogni software developer nella nostra struttura di Eng Modernize**: solo con un approccio di **industrializzazione completa** saremo in grado di esplorare tutto il potenziale dei prodotti di GenAI in questo campo.

Il piano mira a raggiungere **migliaia di software developer in tutto il 2025**, verificando periodicamente i risultati relativi ai miglioramenti della qualità e della produttività lungo il percorso.

Parallelamente alla distribuzione "industriale" del GitHub-Copilot stiamo studiando altri strumenti di mercato per assicurarci di impiegare sempre la soluzione giusta per il caso d'uso giusto.

L'adozione industrializzata conferma la posizione di Eng come partner strategico per l'utilizzo dei servizi di GenAI nello sviluppo software, portando efficienza e miglioramento complessivo a tutti i tipi di aziende.

06 Perchè scegliere noi

Eng Modernize

Trasformare gli ecosistemi dei clienti in un ambiente completamente digitale, guidandoli e supportandoli nel loro percorso verso il cloud.

- Soluzioni cloud-native distribuite sui principali fornitori del mercato: AWS, Azure, Google.
- Soluzioni on prem e private cloud implementate su tecnologie "cloud ready".
- Un Cloud Modernization Process per guidare il cliente nella trasformazione attraverso il cloud.

10+
Anni di esperienza in progetti **cloud-native**

STRUTTURA DISTRIBUITA
Con capacità on-shore e near-shore

MILIARDI
Di transazioni giornaliere sulle nostre risorse cloud consegnate ai nostri clienti

1500+
Sviluppatori specializzati in soluzioni cloud-native e nella modernizzazione dei software aziendali

MILIARDI
Di linee di codice in cui stiamo applicando soluzioni di GenAI per migliorare la qualità e la produttività

CLOUD DELIVERY IN MOLTEPLICI MERCATI
Government, Industry, Energy & Utilities, Media & Communication, Transportation, Finance

Strutture specializzate
in EAI (Enterprise Application Integration) e GIS (Geographic Information System)

Advisory | **Cloud native implementations** | **Deploy automation** | **PAAS services**

I dati visualizzati sono una nostra elaborazione da fonti multiple

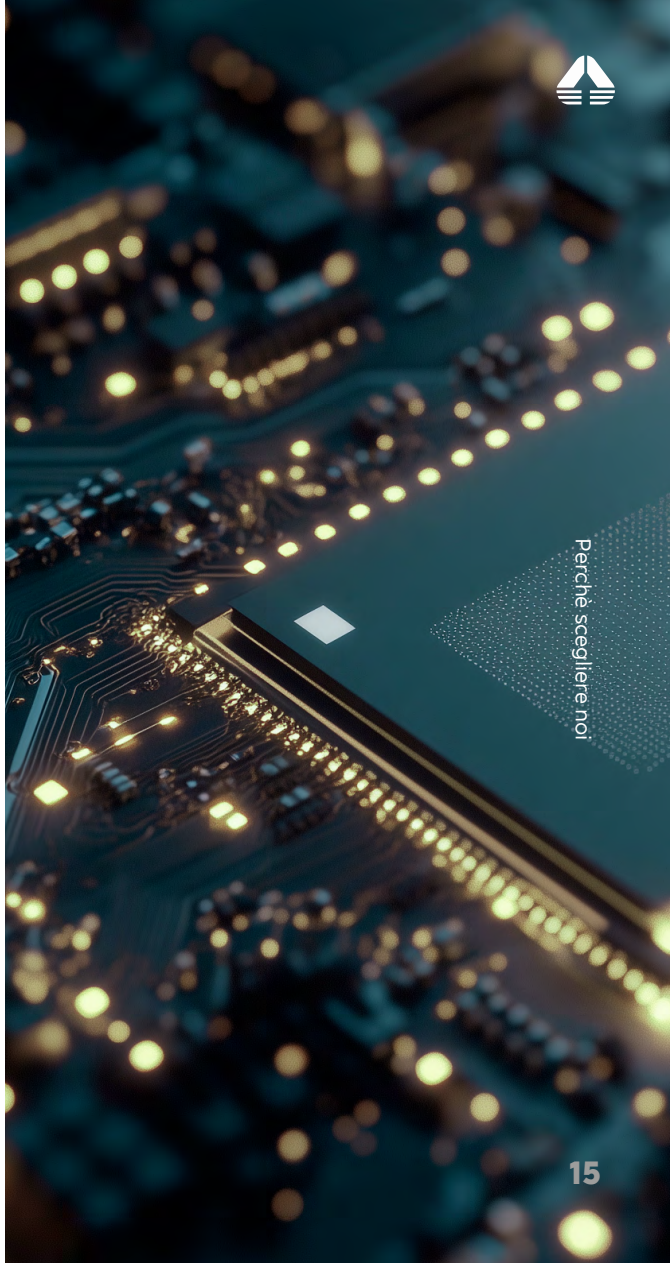
I **sistemi legacy** possono limitare la crescita aziendale e richiedono l'adozione di soluzioni più agili e veloci.
Le **architetture cloud-native**, che supportano scalabilità e flessibilità, consentono lo sviluppo di applicazioni moderne con un time-to-market più rapido, alimentando l'innovazione e l'efficienza aziendale.

Eng Modernize è la technology business line all'interno della nostra Business Unit **Digital Technologies** con una significativa **esperienza in grandi progetti di migrazione** e forti capacità tecniche nella trasformazione di applicazioni legacy in soluzioni scalabili e sicure.

Gestiamo l'**intero ciclo** del processo di modernizzazione, condividendo **best practice e framework**, offrendo un approccio basato sulla consulenza, per garantire transizioni fluide e soluzioni cloud e on-premise ottimizzate.

Forniamo servizi di outsourcing su misura e personalizzati per massimizzare l'uso delle risorse, mitigare i rischi e ridurre i costi, anche grazie a partnership consolidate con i migliori fornitori.
In Eng Modernize, sfruttiamo l'AI Generativa con quattro obbiettivi principali:

- **migliorare la qualità**, riducendo i costi di manutenzione e facilitando l'evoluzione della codebase con un'offerta specifica sull'automazione dei test;
- **ridurre i bug**, poiché l'individuazione precoce di potenziali problemi riduce i costi tipicamente associati alla risoluzione dei bug in produzione;
- **migliorare la sicurezza**, garantendo conformità alle normative e limitando potenziali problemi per azienda e clienti;
- **migliorare la produttività**, perché la riduzione del tempo dedicato alle attività più banali può liberare risorse di sviluppo per attività di maggior valore.



www.eng.it

Engineering Group

@LifeAtEngineering

@EngineeringSpa