

Programmare serve ancora? L'Intelligenza Artificiale cambia le regole del gioco

DI MASSIMO VAILATI

26/10/2026

INTELLIGENZA ARTIFICIALE

SCIENTIFICO S.A. , TT INFORMATICA



L'Intelligenza Artificiale sta cambiando il mondo della programmazione, generando codice e sviluppando applicazioni in pochi secondi. Ma imparare a programmare è ancora importante? Il testo spiega perché le competenze informatiche, il pensiero critico e la capacità di progettare e verificare le soluzioni restano fondamentali. Attraverso un'attività pratica, gli studenti scopriranno come utilizzare l'IA come assistente, riconoscendone i limiti e comprendendo il valore del contributo umano.

ChatGPT scrive codice, GitHub Copilot completa i programmi e l'Intelligenza Artificiale sembra diventare ogni giorno più brava. Allora perché studiare ancora Informatica?

Qualche anno fa, imparare a programmare era considerato una delle competenze più richieste dal mondo del lavoro. Oggi lo scenario sembra essere cambiato. Basta descrivere un problema con poche frasi e strumenti come Codex o Claude Code sono in grado di generare intere applicazioni in pochi secondi.

Di fronte a questa rivoluzione è normale che molti studenti si pongano una domanda: **«Se l'Intelligenza Artificiale scrive il codice, quale sarà il lavoro del programmatore?»**

La risposta è semplice: **il programmatore non sta scomparendo, sta evolvendo.**

Negli ultimi mesi anche le principali aziende del settore hanno confermato questa evoluzione. Microsoft descrive un futuro nel quale gli sviluppatori non si limiteranno più a utilizzare semplici assistenti per completare alcune righe di codice, ma coordineranno veri e propri agenti di Intelligenza Artificiale capaci di realizzare intere funzionalità, eseguire test, individuare bug e aggiornare componenti software esistenti. In questo scenario il ruolo dello sviluppatore si sposta sempre più dalla scrittura del codice alla supervisione, alla progettazione e al coordinamento dell'intero processo di sviluppo, mantenendo però la responsabilità finale sulla qualità e sull'affidabilità del software.

Immaginiamo di dover realizzare un'applicazione per gestire una biblioteca scolastica. Fino a pochi anni fa sarebbe stato necessario progettare il database, creare le classi, sviluppare le schermate e scrivere centinaia di righe di codice.

Oggi possiamo chiedere a un assistente basato sull'Intelligenza Artificiale: *"Realizza un'applicazione web per gestire prestiti e restituzioni dei libri utilizzando Python."*

Nel giro di pochi secondi otteniamo una prima versione del progetto. L'IA può anche suggerire correzioni, spiegare il funzionamento di un algoritmo, tradurre un programma da Java a Python oppure generare automaticamente test e documentazione. Insomma, è come avere accanto un assistente sempre disponibile.

Pensate all'Intelligenza Artificiale come a un giovane sviluppatore appena entrato in azienda. Lavora rapidamente, conosce moltissimi linguaggi di programmazione e produce codice senza stancarsi mai. Ma ha bisogno di qualcuno che controlli il suo lavoro, corregga gli errori e prenda le decisioni più importanti. È proprio questo il ruolo del Software Engineer.

Le risposte generate dall'IA sono spesso sorprendenti, ma non sempre corrette. Può utilizzare librerie che non esistono, proporre algoritmi poco efficienti oppure produrre

codice vulnerabile agli attacchi informatici. Ancora più importante, **non conosce davvero il contesto nel quale il software verrà utilizzato.**

Per esempio, sviluppare un videogioco e progettare un software per un ospedale sono due attività completamente diverse. Nel secondo caso entrano in gioco sicurezza, privacy, affidabilità e responsabilità. Questi aspetti non possono essere affidati esclusivamente a una macchina. Gli strumenti di IA non “ragionano” come gli esseri umani. Analizzano enormi quantità di dati e prevedono quale risposta sia più probabile. Per questo motivo possono sembrare molto convincenti anche quando commettono errori. La regola d’oro è semplice: **non fidarti mai di un programma che non hai verificato.**

Questa necessità di verifica è confermata anche da numerosi esperti di ingegneria del software. Ridurre la programmazione alla semplice capacità di scrivere prompt efficaci significa trascurare la vera complessità dello sviluppo: progettare architetture robuste, gestire sistemi complessi, comprendere il funzionamento della memoria, garantire sicurezza, prestazioni e manutenibilità del codice. La generazione automatica di codice non sostituisce l’ingegneria del software, ma ne rende ancora più importante la conoscenza, perché solo chi possiede solide competenze può valutare criticamente ciò che l’IA produce.



Il lavoro del programmatore sta cambiando.

Per molti anni si è pensato che programmare significasse soprattutto scrivere codice. In realtà il codice è soltanto una parte del lavoro. Prima di iniziare a sviluppare un'applicazione bisogna capire quali problemi risolvere, progettare il sistema, scegliere le tecnologie più adatte e collaborare con molte altre figure professionali.

Un Software Engineer dialoga con clienti, designer, esperti di sicurezza, responsabili di progetto e altri sviluppatori. L'IA può aiutare nella realizzazione del software, ma non può sostituire la capacità di comprendere le esigenze delle persone e trasformarle in soluzioni concrete.

Se il codice può essere generato automaticamente, quali competenze saranno richieste nel futuro?

Cosa dovrebbero imparare gli studenti di oggi?

La prima risposta è sorprendente: **le basi dell'informatica diventano ancora più importanti.**

Algoritmi, strutture dati, basi di dati, reti e sistemi operativi continuano a rappresentare il linguaggio con cui comprendiamo il funzionamento dei computer. Chi possiede queste conoscenze è in grado di capire se il codice prodotto dall'IA sia corretto oppure no.

Accanto alle competenze tecniche crescerà anche l'importanza delle cosiddette *soft skills*: saper lavorare in gruppo, comunicare in modo efficace, analizzare problemi complessi e prendere decisioni responsabili.

In futuro il professionista più richiesto non sarà necessariamente quello che scrive più righe di codice, ma quello che saprà progettare le soluzioni migliori.

Il fatto che l'intelligenza artificiale sia oggi in grado di generare codice non significa che imparare a programmare sia diventato superfluo. Al contrario, il valore delle competenze richieste agli sviluppatori è aumentato.

In passato, un programmatore junior poteva contribuire principalmente scrivendo codice. Oggi questa attività è sempre più automatizzabile e, da sola, non rappresenta più un elemento distintivo. Ciò che fa la differenza è la capacità di comprendere i problemi, progettare soluzioni, definire l'architettura di un sistema, valutare i compromessi tecnici e verificare la qualità del codice prodotto, anche quando viene generato dall'intelligenza artificiale.

Per questo motivo, chi desidera lavorare oggi nel settore dell'informatica deve costruire solide basi di programmazione, algoritmi, architettura del software e ingegneria dei sistemi. In altre parole, servono competenze più ampie e più profonde rispetto al passato: non basta più saper scrivere codice, bisogna essere in grado di guidarne la progettazione e comprenderne il funzionamento.

Il Software Engineer del prossimo decennio dovrà essere capace di:

- comprendere problemi complessi;
- progettare sistemi affidabili;
- utilizzare l'IA come strumento di lavoro;
- collaborare con persone di discipline diverse;
- aggiornarsi continuamente.

Un altro aspetto spesso sottovalutato riguarda infatti l'evoluzione continua del software. L'informatica ha sempre introdotto strumenti che hanno automatizzato parte del lavoro degli sviluppatori: si è passati dai linguaggi a basso livello a linguaggi sempre più evoluti e sicuri, senza che il ruolo del programmatore scomparisse. Ogni nuovo livello di astrazione ha semplicemente spostato il lavoro verso problemi più complessi. Anche l'Intelligenza Artificiale rappresenta un ulteriore passo in questa direzione: accelera lo sviluppo, ma la manutenzione dei sistemi, la comprensione dei principi fondamentali dell'informatica e la capacità di governare progetti complessi restano attività profondamente umane. Per questo motivo, soprattutto per chi inizia oggi questo percorso, è ancora fondamentale studiare algoritmi, reti, gestione della memoria e architettura dei sistemi, evitando di delegare completamente il ragionamento agli strumenti automatici.

Anche il modo di studiare Informatica sta cambiando. Utilizzare l'Intelligenza Artificiale durante un laboratorio non significa "copiare". Significa imparare a lavorare con uno strumento professionale.

Naturalmente c'è una differenza enorme tra chiedere all'IA di svolgere tutto il lavoro e utilizzarla come supporto. Il vero apprendimento nasce quando lo studente confronta le soluzioni proposte, individua gli errori, le migliora e comprende il motivo delle modifiche effettuate. È esattamente ciò che avviene nelle aziende di sviluppo software.



Il futuro appartiene a chi sa pensare.

Ogni rivoluzione tecnologica ha modificato il modo di lavorare delle persone. La calcolatrice non ha eliminato la matematica, Internet non ha eliminato la necessità di studiare, l'Intelligenza Artificiale non eliminerà gli informatici. Cambierà semplicemente il loro ruolo.

Chi saprà usare questi strumenti con spirito critico sarà più produttivo e potrà dedicare più tempo alla progettazione, alla creatività e all'innovazione. In fondo, il vero valore di un informatico non è la velocità con cui scrive codice, ma la capacità di comprendere un problema e trovare la soluzione migliore.

L'Intelligenza Artificiale può aiutarci a costruire il software. Ma saranno ancora le persone a decidere **quale software costruire e perché**.

“L'Intelligenza Artificiale può scrivere il codice di un programma. Ma solo una persona può decidere se quel programma migliorerà davvero la vita di chi lo userà.”

Forse questa è la competenza più importante che uno studente di Informatica possa sviluppare oggi.

Attività pratica

Chi è il vero Software Engineer: tu o l'Intelligenza Artificiale?

Obiettivo

Sperimentare come l'Intelligenza Artificiale possa essere un valido assistente nello sviluppo software e comprendere quali competenze rimangono indispensabili agli esseri umani.

Fase 1 - Il problema

Il docente presenta un caso reale. Ad esempio: **Una scuola desidera sviluppare un'applicazione per prenotare i laboratori di Informatica.**

Il sistema deve permettere ai docenti di verificare la disponibilità delle aule, effettuare prenotazioni e annullarle. Inoltre deve impedire doppie prenotazioni e conservare lo storico delle attività.

The screenshot shows the LabSchool web application interface for a teacher, Prof. Rossi. The interface is divided into several sections:

- Header:** Welcome message "Benvenuto, Prof. Rossi!" and a notification bell icon.
- Summary Cards:** Four cards showing key statistics: 12 bookings in the next 7 days, 5 confirmed bookings this week, 2 bookings pending approval this week, and 6 available laboratories today.
- Calendar:** A weekly calendar view for May 20-26, 2024, showing a grid of time slots (08:00-17:00) and color-coded booking blocks for various laboratories (e.g., Lab. Informatica 1, Lab. Elettronica).
- Upcoming Bookings:** A list of upcoming bookings with details like laboratory name, date, time, and status (e.g., "Lab. Informatica 1 2A Informatica" on Thursday, May 23, 08:00-10:00).
- Recent Bookings:** A table of recent bookings with columns for Laboratory, Class/Group, Date, Time, and Status.
- Quick Actions:** A sidebar on the right with buttons for "Nuova prenotazione", "Le mie prenotazioni", and "Tutte le prenotazioni".
- Footer:** Copyright notice: "© 2024 LabSchool - Istituto Tecnico Tecnologico - Tutti i diritti riservati".

Laboratorio	Classe/Gruppo	Data	Orario	Stato
Lab. Rete	4B Informatica	21/05/2024	11:00 - 13:00	Confermata
Lab. Multimediale	3A Grafica	20/05/2024	14:00 - 16:00	Confermata
Lab. Informatica 2	1A Informatica	21/05/2024	09:00 - 11:00	In attesa
Lab. Elettronica	4A Elettronica	21/05/2024	15:00 - 17:00	Confermata

Il problema viene discusso collettivamente senza parlare di codice.

Gli studenti devono rispondere a domande come:

- Chi utilizzerà il programma?
- Quali informazioni devono essere memorizzate?
- Quali problemi potrebbero verificarsi?
- Quali funzionalità sono indispensabili?

L'obiettivo è far comprendere che il primo lavoro del Software Engineer consiste nel capire il problema.

Fase 2 - Chiediamo aiuto all'IA

Ogni gruppo prepara un prompt dettagliato e lo sottopone a Codex o Claude Code. Ad esempio:

“Progetta un'applicazione web per la gestione delle prenotazioni dei laboratori scolastici utilizzando Python e Flask. Descrivi anche il database e proponi una semplice interfaccia grafica.”

L'IA produrrà una soluzione.

Fase 3 - La revisione

Qui inizia il vero lavoro. Ogni gruppo analizza criticamente la proposta dell'IA. Gli studenti devono individuare:

- funzionalità mancanti;
- errori progettuali;
- problemi di sicurezza;
- casi non considerati;
- miglioramenti possibili.

Il docente può stimolare la riflessione con alcune domande. Ad esempio:

- Cosa succede se due docenti prenotano contemporaneamente la stessa aula?
- Chi può modificare una prenotazione?
- È previsto un sistema di autenticazione?

- Lo storico può essere cancellato?
- Il database è progettato correttamente?

A questo punto gli studenti capiscono che il codice è soltanto una parte del progetto.

Fase 4 - Miglioriamo la soluzione

Ogni gruppo modifica il progetto suggerito dall'IA. Non è necessario riscrivere tutto il codice e possiamo chiedere le modifiche direttamente a Codex o Claude Code.

L'importante è spiegare:

- cosa è stato cambiato;
- perché è stato cambiato;
- quale principio di Ingegneria del Software è stato applicato.

Fase 5 - Presentazione finale

Ogni gruppo espone il proprio lavoro in cinque minuti. La presentazione dovrebbe rispondere a tre semplici domande:

- Cosa ha fatto bene l'IA?
- Dove ha sbagliato o è stata manchevole?
- Quale contributo umano è stato indispensabile?

Conclusione

Alla fine dell'attività il docente propone una riflessione collettiva. Se tutti i gruppi hanno utilizzato la stessa Intelligenza Artificiale, perché nessuno ha ottenuto esattamente lo stesso progetto?

Gli studenti scopriranno che la differenza non è stata l'IA. La differenza è stata la qualità del ragionamento umano. Ogni gruppo espone il proprio lavoro in cinque minuti.