
aracne

Classificazione Decimale Dewey:

005.13028563 (23.) LINGUAGGI DI PROGRAMMAZIONE. Intelligenza artificiale

GERARDO IOVANE

AI AND VIBE CODING

A NEW FRONTIER FOR DEVELOPING APPLICATIONS

aracne

ISBN
979-12-218-2017-1

PRIMA EDIZIONE
ROMA 18 GIUGNO 2025

Indice

Introduzione

Nel cuore delle trasformazioni tecnologiche che stanno ridisegnando l'orizzonte della produzione software, questo testo si propone di affrontare in modo sistemico, critico e propositivo il fenomeno emergente del low-code e del no-code, non come semplici strumenti di facilitazione tecnica, ma come vere e proprie tecnologie abilitanti in grado di ridefinire l'ontologia stessa del "fare software". Il testo è pensato per professionisti, studiosi e innovatori con una solida formazione, desiderosi di comprendere, utilizzare e ripensare in chiave critica ed etica il ruolo del codice, della narrazione e della progettazione esperienziale nell'ecosistema contemporaneo dello sviluppo digitale. Partendo dall'idea che il codice possa essere concepito come linguaggio dell'esperienza — e non solo come linguaggio formale di interazione uomo-macchina — si indaga come le piattaforme low-code e no-code siano oggi strumenti per l'espressione di una nuova cultura progettuale, democratica, accessibile, creativa. Il volume si struttura come un vero e proprio manifesto concettuale e operativo del "Vibe Coder", una nuova figura emergente che unisce capacità tecnica, sensibilità narrativa, consapevolezza sociale e intelligenza progettuale, capace di dare forma a ecosistemi digitali che parlano la lingua della contemporaneità. L'intero libro si articola in modo da offrire una prospettiva integrata e trasversale: si parte da una trattazione teorica e filosofica del concetto di codice come espressione culturale e semiotica, per poi passare all'indagine delle logiche abilitanti delle piattaforme di sviluppo semplificato e una mappatura analitica delle principali tecnologie impiegate. A seguire, il lettore troverà un'esposizione articolata dei dieci principi fondamentali del Vibe Coding, che vanno dalla centralità dell'esperienza utente alla dimensione etica e partecipativa della progettazione, fino all'adozione del low-code come pratica espressiva e trasformativa. Ogni principio è corredato da esempi, riflessioni critiche, riferimenti epistemologici e implicazioni pratiche, in un percorso che fonde riflessione teorica e concretezza operativa. Uno spazio rilevante è riservato allo storytelling applicato allo sviluppo, inteso come strategia non solo comunicativa ma cognitiva, in cui le app diventano ambienti narrativi, capaci di costruire significato, appartenenza e valore condiviso. In questo contesto, vengono presentati numerosi casi d'uso — oltre settanta — che rappresentano una vera e propria galleria di applicazioni progettate secondo i principi del Vibe Style: app per antropologi, nutrizionisti, operatori olistici, esperti di cybersecurity, artisti, scrittori, analisti finanziari, sviluppatori, educatori, critici d'arte, social media manager, figure aziendali come CEO, CFO, CTO, CMO e molte altre figure professionali emergenti o consolidate, tutte accomunate dalla volontà di utilizzare la tecnologia come mezzo per dare forma a un'esperienza trasformativa. Ogni app è descritta secondo un format preciso, che include: concept generale, struttura tecnica in low-code, strumenti consigliati, moduli principali, storytelling

integrato, elementi distintivi VIBE ed estensioni possibili. Questo format rigoroso consente al lettore non solo di comprendere la filosofia progettuale alla base di ciascuna app, ma anche di replicarla o adattarla al proprio contesto operativo. L'obiettivo non è solo fornire un repertorio di buone pratiche, ma offrire una grammatica trasformativa, una metodologia fluida e riutilizzabile per chi desidera progettare soluzioni digitali che mettano al centro l'essere umano, le sue emozioni, le sue comunità di senso. Un'attenzione particolare è riservata all'aspetto etico e politico dello sviluppo low-code: se da un lato tali strumenti permettono un accesso più inclusivo e diffuso alla progettazione digitale, dall'altro sollevano interrogativi su standardizzazione, privacy, sostenibilità e disintermediazione del sapere tecnico. Per questo motivo, il testo insiste sulla necessità di un approccio riflessivo, consapevole e soprattutto situato: la tecnologia non è mai neutra e il modo in cui viene usata (o evitata) riflette scelte culturali, visioni del mondo, modelli di relazione. In tal senso, l'approccio Vibe Coding si propone come risposta integrata e critica alle derive di ipertecnicismo o di deresponsabilizzazione cognitiva che rischiano di affliggere parte della produzione software contemporanea. Un altro aspetto chiave del testo è l'enfasi sulla didattica trasformativa: sono presenti esercizi pratici per aiutare il lettore a trovare il proprio "vibe", ovvero il proprio stile progettuale, la propria sintonia tra competenza tecnica e visione etica. Tali esercizi sono pensati per stimolare l'autoriflessione, l'immaginazione progettuale, l'empatia e la creatività, in una forma laboratoriale che può essere applicata anche in contesti formativi, universitari, aziendali o comunitari. In sintesi, questo libro è pensato come una piattaforma multidimensionale: teorica e pratica, critica e generativa, tecnica e poetica. È un invito a superare le separazioni tra codice e cultura, tra sviluppo e significato, tra innovazione e umanesimo. È una chiamata a progettare ambienti digitali che non siano solo funzionali, ma anche significativi, accoglienti, trasformativi. In un mondo attraversato da crisi ecologiche, sociali e cognitive, il codice — se riletto come linguaggio dell'esperienza — può diventare uno degli strumenti più potenti per costruire futuri desiderabili. Ma per farlo, è necessario un cambio di paradigma: un passaggio dal puro codice al Vibe Coding, dall'algoritmo alla narrazione, dalla funzione al senso. Questo libro è il primo passo in quella direzione.

PARTE I – Inquadramento Cognitivo

Capitolo 1 – Introduzione al Vibe Coding

Il concetto di Vibe Coding si colloca all'intersezione tra scienze cognitive, progettazione esperienziale e ingegneria del software a bassa complessità. Esso rappresenta una declinazione evolutiva delle metodologie di sviluppo low-code e no-code, in cui l'accento viene posto non esclusivamente sulla riduzione della scrittura manuale del codice, bensì sull'armonizzazione dei flussi creativi, delle interfacce e delle architetture attraverso principi di coerenza esperienziale, empatia progettuale e fluidità operativa. Questo capitolo introduttivo mira a delineare i fondamenti concettuali, filosofici e operativi del Vibe Coding, fornendo al lettore una visione sistemica delle sue implicazioni e delle potenzialità applicative in contesti professionali avanzati.

La prima sezione analizzerà l'etimologia e l'evoluzione semantica del termine "vibe" nel contesto tecnologico, tracciando un parallelismo tra le "vibrazioni cognitive" dell'utente e la risposta dinamica dei sistemi software. Verranno inoltre esplorate le radici interdisciplinari di questo paradigma, con riferimenti alla semiotica visiva, all'ergonomia digitale e alla psicologia delle interfacce. Tale analisi consentirà di porre le basi teoriche per comprendere come il Vibe Coding si distingua dai tradizionali approcci ingegneristici, superando la dicotomia tra funzionalità ed estetica.

La seconda sezione sarà dedicata al profilo del programmatore vibrazionale, un nuovo tipo di sviluppatore che agisce secondo principi di coerenza intuitiva e sintonia funzionale. Questa figura professionale combina competenze tecniche e sensibilità progettuale, collocandosi come ponte tra sviluppo e design. L'approfondimento comprenderà una tipizzazione cognitiva di tali profili, una valutazione delle competenze soft complementari e una descrizione dei contesti in cui tali capacità possono essere valorizzate in ambienti multidisciplinari e orientati all'innovazione.
Nella terza sezione si procederà alla classificazione e analisi comparativa degli strumenti low-code e no-code che supportano il Vibe Coding, con attenzione particolare agli ambienti che permettono una personalizzazione visuale profonda, una gestione modulare delle logiche di interazione e una trasparenza architetturale adeguata a contesti enterprise. L'approccio sarà quello di una disamina tecnica ma anche qualitativa, prendendo in considerazione criteri di UX, interoperabilità e curva di apprendimento. Questo permetterà di definire linee guida pratiche per la selezione dello strumento più adatto in base agli obiettivi di progetto e al profilo del team.
La quarta sezione si concentrerà sugli aspetti di design vibrazionale, ovvero sull'elaborazione di interfacce che non solo funzionano ma "risuonano" con l'utente.

Verranno trattati i principi cognitivi che guidano la percezione visiva e l'interazione fluida, incluse le nozioni di sinestesia digitale, ritmo visivo e feedback armonico. Questa parte del capitolo offrirà anche modelli euristici e schemi operativi per guidare la progettazione in ambienti low-code, con un focus sull'usabilità emozionale e la coerenza narrativa dell'esperienza utente.

Infine, la quinta sezione introdurrà il concetto di "Vibe Canvas", uno strumento metodologico e operativo per orchestrare lo sviluppo di un progetto in chiave vibrazionale. Tale canvas consente di rappresentare in modo integrato le componenti funzionali, emotive e narrative di un'applicazione, favorendo una visione olistica del processo di sviluppo. Verranno illustrate le componenti costitutive del canvas, i metodi per la sua implementazione nei diversi contesti e le modalità con cui può essere integrato nei processi agili e nei framework di progettazione evolutiva. In conclusione, questa sezione fungerà da ponte verso i capitoli successivi, dove tali elementi verranno sviluppati in modo esteso e applicativo.

1.1 – Origine e significato del termine 'vibe' nel contesto dello sviluppo software

Il termine "vibe", nella sua accezione originaria, proviene dal linguaggio informale anglofono, dove rappresenta una contrazione di "vibration", ovvero vibrazione. In ambito sociolinguistico e semiotico, tale concetto ha assunto nel tempo connotazioni sempre più raffinate, divenendo una forma sintetica di riferimento a percezioni soggettive, atmosfere e risonanze emotive. Sebbene inizialmente relegato a contesti informali o artistici, il termine è stato progressivamente adottato da numerosi domini, tra cui il design interattivo, l'arte generativa e, più recentemente, lo sviluppo software orientato all'esperienza. In tale contesto, "vibe" non è più solo una metafora emozionale, ma diviene un costrutto operativo utile a descrivere l'interazione qualitativa tra sistemi digitali e utenti umani. Esso incarna il tentativo di trasferire nella dimensione computazionale alcune caratteristiche della percezione analogica, in particolare quelle legate alla fluidità, all'armonia e alla sintonia contestuale. Tale processo è particolarmente rilevante nel paradigma del Vibe Coding, che si propone di superare la dicotomia rigida tra logica e sensibilità, fondendo l'efficienza algoritmica con l'intuizione percettiva.

Dal punto di vista etimologico, la diffusione del termine "vibe" nelle discipline computazionali rappresenta una forma di trasposizione semantica che accompagna l'evoluzione della progettazione software verso forme sempre più user-centered e affettive. In effetti, con l'emergere dell'UX design come disciplina autonoma e transdisciplinare, termini come "tone", "mood" e "vibe" hanno iniziato a popolare il lessico tecnico, ridefinendo i criteri di qualità dei prodotti digitali. Il concetto di

"buona vibe" di un'applicazione non si limita alla sua funzionalità, ma include elementi come la coerenza cromatica, la reattività dei componenti, la musicalità dell'interazione e la capacità del sistema di rispondere anticipatamente alle esigenze dell'utente. In questo scenario, il Vibe Coding si propone come una forma di codifica esperienziale, in cui le scelte implementative sono guidate da euristiche percettive e da pattern emozionali ricorrenti, spesso ispirati a modelli cognitivi provenienti dalle neuroscienze e dalla psicologia comportamentale.

L'introduzione della nozione di "vibe" all'interno dello sviluppo software trova giustificazione teorica anche nell'evoluzione delle interfacce utente verso modalità sempre più immersive e sensoriali. La transizione da interfacce puramente testuali a quelle grafiche e da queste ultime a interfacce vocali, tattili e multimodali ha ampliato il campo semantico degli strumenti concettuali utilizzabili per descrivere l'esperienza utente. In tale contesto, la "vibe" rappresenta una sintesi di elementi percettivi che contribuiscono alla formazione di un giudizio intuitivo e immediato sulla qualità dell'interazione. Essa non è quantificabile attraverso parametri puramente tecnici, ma può essere descritta, analizzata e replicata tramite l'uso di modelli narrativi, archetipi visuali e strutture armoniche. La progettazione di software orientato al vibe coding implica dunque una sensibilità aumentata verso queste componenti, rendendo necessaria una formazione incrociata che includa elementi di estetica computazionale, semantica funzionale e teoria dell'embodiment.

Dal punto di vista metodologico, il Vibe Coding si inserisce in una linea di sviluppo concettuale che include il design emozionale, il design comportamentale e l'interaction design esperienziale. La sua specificità consiste nell'adozione di strategie di codifica che privilegiano la coerenza percettiva rispetto all'ottimizzazione strutturale. Questo significa, ad esempio, che una soluzione visivamente più armonica può essere preferita a una soluzione tecnicamente più performante, qualora la prima generi una "vibe" più convincente o coerente con il contesto d'uso. Tale approccio impone una revisione dei tradizionali criteri di valutazione del software, introducendo metriche qualitative che includono il comfort cognitivo, la fluidità esperienziale e la capacità di generare coinvolgimento. In questo senso, la "vibe" diviene un indicatore progettuale e non solo un epifenomeno dell'interfaccia. L'integrazione del concetto di vibe nei processi di sviluppo è ulteriormente facilitata dai moderni ambienti di programmazione low-code e no-code. Questi strumenti, concepiti per abbassare la soglia di ingresso alla progettazione software, offrono interfacce visive che rendono immediatamente tangibile la qualità esperienziale del prodotto. In tali ambienti, la progettazione diviene un atto compositivo e quasi musicale, dove ogni componente può essere manipolato in tempo reale e valutato secondo criteri sensoriali oltre che funzionali. La manipolazione diretta degli

elementi grafici, la possibilità di iterare rapidamente su versioni multiple e la disponibilità di librerie visive avanzate costituiscono un terreno fertile per l'espressione vibrazionale del codice. Il Vibe Coding trova qui un habitat naturale, in quanto la struttura visuale stessa del linguaggio di sviluppo favorisce l'emergere di pattern coerenti e facilmente sintonizzabili con l'utente finale.

Infine, è necessario evidenziare come l'adozione consapevole del paradigma Vibe Coding implichi un cambiamento culturale all'interno delle organizzazioni che producono software. In luogo di una visione meramente ingegneristica e funzionale, il Vibe Coding propone una cultura dell'ascolto, dell'iterazione estetica e della responsabilità percettiva. Le scelte progettuali non sono più guidate esclusivamente da logiche prestazionali, ma anche da considerazioni etiche e affettive, in un'ottica che valorizza l'empatia digitale. Questo comporta, tra le altre cose, la necessità di ridefinire i processi di formazione, di introdurre nuove figure professionali intermedie tra il design e la programmazione e di sviluppare ambienti di sviluppo che supportino esplicitamente l'espressione e la valutazione delle vibrazioni cognitive. In tale direzione, il Vibe Coding non è solo una tecnica, ma un cambiamento di paradigma nel modo di intendere il rapporto tra uomo e macchina.

1.2 – Il profilo del programmatore vibrazionale

Il profilo del programmatore vibrazionale si configura come una figura emergente nel panorama della progettazione software contemporanea, caratterizzata da un approccio olistico e multisensoriale allo sviluppo. Contrariamente alla concezione tradizionale del programmatore come entità strettamente logico-deduttiva, il programmatore vibrazionale integra nella propria prassi operativa elementi di sensibilità estetica, empatia cognitiva e capacità di anticipazione emotiva. L'identità epistemologica del programmatore vibrazionale si fonda su una triplice competenza: tecnica, percettiva e riflessiva. La competenza tecnica riguarda la padronanza degli strumenti di sviluppo, in particolare degli ambienti low-code e no-code, nonché la conoscenza dei paradigmi di interazione uomo-macchina. La dimensione percettiva, invece, si riferisce alla capacità di cogliere sfumature sensoriali e microsegnali provenienti dall'interfaccia, mentre quella riflessiva implica la disponibilità a concepire in modo coerente e conclusivo.

Il programmatore vibrazionale si distingue anche per la sua capacità di modulare l'ambiente di sviluppo in funzione del contesto esperienziale desiderato. Ciò implica una costante attenzione ai pattern visivi, ai tempi di risposta delle interfacce, alla fluidità delle transizioni e alla coerenza semiotica degli elementi grafici. L'atto del programmare, in questo quadro, si trasforma in un processo interpretativo.

Dal punto di vista cognitivo, questa figura è spesso associata a profili neurodivergenti o altamente sensibili, capaci di elaborare simultaneamente input multipli e di cogliere correlazioni tra domini apparentemente disgiunti. Tale caratteristica rende il programmatore vibrazionale particolarmente adatto a lavorare in contesti di innovazione radicale, in cui la capacità di anticipare scenari di uso e reazioni utente è cruciale. A livello organizzativo, l'introduzione del programmatore vibrazionale nei team di sviluppo richiede una ridefinizione dei processi collaborativi e delle metriche di valutazione delle performance. Le sue modalità operative, infatti, mal si conciliano con modelli rigidamente sequenziali o basati esclusivamente sulla produttività numerica. Un ulteriore elemento distintivo del programmatore vibrazionale è la sua predisposizione alla co-creazione, ovvero alla collaborazione sinergica con designer, storyteller, psicologi cognitivi e altri attori del processo progettuale. In tal senso, egli agisce da catalizzatore di connessioni interdisciplinari, contribuendo a costruire ambienti di sviluppo inclusivi, adattivi e orientati all'esperienza. La co-creazione non è intesa come semplice divisione dei compiti, bensì come processo di mutua influenza. Infine, va sottolineato come il riconoscimento formale di questa figura professionale sia ancora in fase embrionale, eppure sempre più richiesto nei contesti aziendali che puntano all'eccellenza esperienziale. Le grandi piattaforme digitali, i laboratori di interaction design e i centri di ricerca sull'intelligenza artificiale stanno progressivamente valorizzando profili capaci di operare con sensibilità "vibrazionale". Ciò prefigura un'evoluzione dei curricula accademici e delle certificazioni professionali.

1.3 – Differenza tra codice logico e codice vibrante

Nell'ambito della progettazione e dello sviluppo software, la distinzione tra codice logico e codice vibrante rappresenta una nuova frontiera epistemologica e metodologica. Il codice logico, nella sua accezione tradizionale, si fonda su strutture sintattiche e semantiche orientate alla risoluzione efficiente di problemi computazionali. Esso privilegia la correttezza formale, la coerenza algoritmica e la prevedibilità funzionale.

Il codice vibrante, invece, si definisce come una modalità di programmazione che incorpora dimensioni estetiche, affettive e percettive all'interno dell'atto computazionale. Piuttosto che limitarsi alla risoluzione di compiti logici, esso mira a generare esperienze significative, risuonanti con le dinamiche emotive e cognitive dell'utente finale. Tale codice non è caratterizzato unicamente da ciò che fa, ma soprattutto da come lo fa e da come viene percepito nel contesto d'uso.

Un elemento distintivo fondamentale del codice vibrante risiede nella sua capacità di trasmettere coerenza percettiva attraverso le microinterazioni. Questi elementi, spesso trascurati nella programmazione logico-funzionale, diventano centrali nella progettazione vibrazionale. La reattività di un pulsante, la transizione fluida tra due schermate, l'armonia cromatica dell'interfaccia: sono tutti aspetti che, se coerentemente orchestrati, generano una vibrazione positiva nel fruitore. Dal punto di vista strutturale, il codice logico tende a massimizzare la chiarezza e l'ottimizzazione algoritmica, ricorrendo a pattern consolidati, modularità e separazione delle responsabilità. In ambito enterprise, questa impostazione garantisce robustezza, scalabilità e manutenibilità del codice. Tuttavia, essa tende a marginalizzare le componenti sensoriali dell'esperienza d'uso, relegando tali aspetti a livelli superiori di astrazione (UI/UX design), spesso gestiti da altri attori nel processo.

Il codice vibrante rompe questa gerarchia tradizionale, reintegrando l'estetica e la sensorialità all'interno del core del processo implementativo. In ambienti low-code e no-code, dove la logica e l'interfaccia sono spesso rappresentate visivamente in un unico spazio compositivo, tale integrazione risulta non solo possibile, ma auspicabile. Il programmatore non agisce più solo come ingegnere, ma anche come compositore, curatore e regista di esperienze digitali.

L'adozione del codice vibrante richiede una mutazione del paradigma valutativo dei progetti software. I criteri tradizionali, basati esclusivamente su indicatori quantitativi (tempo di risposta, numero di errori, complessità computazionale), vengono affiancati o sostituiti da metriche qualitative quali la fluidità percettiva, la coerenza esperienziale, il grado di coinvolgimento emotivo. Questa trasformazione richiede anche nuove metodologie di testing, in cui il feedback dell'utente assume un ruolo centrale.

Infine, è importante sottolineare che il codice vibrante non esclude il codice logico, ma lo ingloba in una struttura semantica e sensoriale più ampia. In altri termini, il codice vibrante è una sovrastruttura che mantiene l'efficienza computazionale del codice logico, arricchendola con dimensioni narrative, affettive e percettive. Tale approccio si rivela particolarmente efficace in ambiti in cui l'esperienza utente è un fattore critico di successo: applicazioni educative, ambienti immersivi, sistemi d in modo coerente e conclusivo.

1.4 – Le emozioni nel design del software

Nel contesto contemporaneo dello sviluppo software, l'integrazione delle emozioni nella fase di design rappresenta una delle frontiere più avanzate della progettazione centrata sull'utente. Questo approccio si basa sulla consapevolezza che le applicazioni non sono semplici strumenti funzionali, bensì ambienti interattivi che mediano esperienze complesse, dove le emozioni giocano un ruolo centrale nella percezione di valore, utilità e desiderabilità.

L'inclusione delle emozioni nel design software si fonda su una vasta letteratura interdisciplinare che comprende contributi dalla psicologia cognitiva, dalle neuroscienze affettive, dall'estetica computazionale e dal design interattivo. In particolare, il modello OCC (Ortony, Clore & Collins) delle emozioni ha trovato applicazione nella modellazione delle risposte affettive all'interazione uomo-macchina, consentendo di categorizzare le emozioni evocate da specifici eventi computazionali.

La progettazione emozionale comporta l'identificazione e l'attivazione di specifici trigger emotivi, che possono essere visivi, uditivi, tattili o comportamentali. L'utilizzo di transizioni fluide, feedback sonori armonici, animazioni coerenti con l'identità semantica dell'applicazione e un linguaggio interfacciato empatico, concorre alla costruzione di un'esperienza che non solo informa o guida l'utente, ma lo coinvolge affettivamente. Ciò è particolarmente rilevante nelle interfacce conversazionali. Un aspetto fondamentale dell'approccio emozionale al design software è la coerenza narrativa e simbolica dell'interfaccia. Gli utenti sono particolarmente sensibili alla congruenza tra ciò che vedono, ciò che fanno e ciò che provano. Disallineamenti percettivi – ad esempio tra un'interfaccia apparentemente "amichevole" e un comportamento rigido del sistema – possono generare reazioni negative, come frustrazione, ansia o senso di inadeguatezza.

In ambito enterprise, l'integrazione delle emozioni nel design del software assume una valenza strategica, poiché incide direttamente sulla produttività, sulla fidelizzazione degli utenti e sulla riduzione dei costi di supporto. Applicazioni che suscitano emozioni positive tendono a essere percepite come più semplici, più affidabili e più efficaci, indipendentemente dalla loro reale complessità.

La progettazione emozionale richiede, tuttavia, un cambiamento profondo nei processi di sviluppo, che devono incorporare metodi qualitativi di valutazione dell'esperienza, come le interviste fenomenologiche, i test con valenza affettiva e le metriche di engagement emozionale. Strumenti come gli emotional journey map e i sensori biometrici offrono oggi la possibilità di oggettivare il vissuto soggettivo dell'utente, rendendo accessibili dati prima considerati esclusivamente qualitativi.

Infine, è opportuno riconoscere che il design emozionale non si limita a rendere il software "più piacevole", ma contribuisce a ridefinire l'ontologia stessa dell'interazione uomo-macchina. La macchina non è più solo un esecutore di comandi, ma un partner relazionale, capace di rispondere, adattarsi e persino anticipare gli stati d'animo dell'utente. In questo senso, l'integrazione delle emozioni nel design si configura come una forma di intelligenza empatica distribuita.

1.5 – Il cambio di paradigma tra programmazione tradizionale, AI coding, low-code e no-code

Il panorama dello sviluppo software sta attraversando una trasformazione strutturale che investe non soltanto gli strumenti e le tecnologie utilizzate, ma anche le fondamenta epistemologiche della programmazione stessa. L'avvento dell'intelligenza artificiale generativa, l'affermazione di strumenti low-code e no-code, e il crescente bisogno di accessibilità tecnologica hanno favorito un ripensamento radicale del ruolo dello sviluppatore, della natura del codice e della relazione tra essere umano e macchina.

La programmazione tradizionale, così come si è consolidata a partire dagli anni '60 del Novecento, si fonda su principi di formalizzazione logico-matematica, controllo deterministico del flusso di esecuzione e codifica testuale esplicita. In questo modello, il programmatore è responsabile della definizione completa delle regole che governano il comportamento del sistema, secondo linguaggi formali che devono essere appresi e gestiti con rigore. Con l'introduzione delle tecnologie di intelligenza artificiale generativa, si assiste a un decentramento progressivo dell'autorità sintattica e semantica del programmatore. Gli strumenti di AI coding, come gli assistenti intelligenti basati su modelli di linguaggio di grandi dimensioni, non si limitano a suggerire porzioni di codice, ma partecipano attivamente alla strutturazione semantica del progetto, operando in regime di co-creazione. Parallelamente, l'emergere dei paradigmi low-code e no-code ha aperto nuove possibilità di accesso allo sviluppo software da parte di utenti non tecnici, noti come citizen developers. Questi strumenti consentono la creazione di applicazioni complesse mediante interfacce visive, compositori drag-and-drop, e workflow predefiniti, abbattendo le barriere di ingresso e democratizzando la capacità di innovazione.

Il cambio di paradigma evidenziato si riflette anche nella trasformazione dei criteri di valutazione delle soluzioni software. Nella programmazione tradizionale, i principali indicatori di qualità sono di tipo tecnico: efficienza computazionale, robustezza, scalabilità e sicurezza. Nei nuovi contesti, invece, si affiancano dimensioni esperienziali: intuitività, accessibilità, personalizzazione e capacità di adattamento