Nell'era dell'AI, quali aspetti dello sviluppo *software* perdono valore distintivo?

di Gianluca Bisceglie | https://www.linkedin.com/in/gianlucabisceglie/

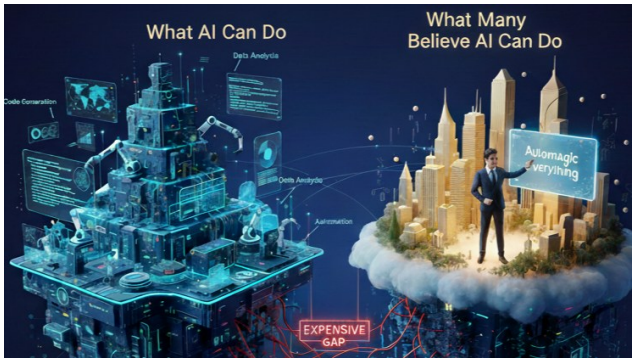
« Gianluca Bisceglie è un ingegnere elettronico con MBA alla London Business School. Il suo lungo e intenso percorso professionale è su LinkedIn, dove si possono anche leggere, tra i tanti spunti proposti, gli articoli che da qualche tempo dedica all'AI, all'impatto su attività produttive, finanza, progettazione e realizzazione di infrastrutture e dispositivi tecnologici, ma anche ai rischi di un suo utilizzo troppo meccanico e acritico. Con questa nota, riadattata liberamente dall'Inglese, si iniziano a riproporre gli articoli di Gianluca sull'AI.

La facilità con cui l'AI ormai permette di generare codice per alimentare sistemi agentici non deve creare illusione che ci si possa affidare ciecamente *once-and-for-all* a soluzioni automatiche flessibili e onniscienti. Tutt'altro: man mano che l'AI diventa più capace, e ha la possibilità di compiere scelte operative, con effetti difficilmente reversibili o non reversibili affatto, l'intervento umano assume importanza cruciale: dalla fase di validazione dei modelli, alla messa a punto delle regole di ingaggio, alla lettura degli *output*. "*Le organizzazioni che usciranno indenni dall'ondata di innovazioni radicali apertasi con l'AI non saranno quelle che hanno usato di più l'AI. Saranno quelle che si sono poste il problema, in modo deliberato e precoce, di fissare un chiaro confine tra dove finisce l'automatismo delegato all'AI e dove inizia la responsabilità umana*".

Piacevole da leggere, divulgativa e ricca di suggestioni, questa nota affronta un problema che in realtà, nei suoi tratti generali, si è presentato a economisti ed econometrici già da tanto tempo, e che l'arrivo dell'AI sta solo enfatizzando. Adesso che l'AI può aiutare a mettere assieme grandi *dataset* scandagliando e raccogliendo informazioni da tutta la rete, e può addirittura scrivere il codice di STATA, R, EViews, SPSS, PYTHON, etc., suggerendo anche tecniche di stima dei parametri di relazioni funzionali, diviene ancor più pericolosa la cosiddetta sindrome della *black-box*, quella di affidarsi ai *software* e ai *tool* preconfezionati perdendo contatto con la realtà sottostante.

"*Quando un sistema agentico fraintende il contesto o sceglie l'azione sbagliata, non si limita a suggerire una cattiva idea come farebbe un chatbot. Esegue quella cattiva idea alla velocità della macchina. Il ruolo dell'ingegnere o del tecnico umano, invece di alleggerirsi, diventa più critico: qualcuno deve comunque costruire il sandbox, definire i permessi, progettare i trigger di rollback e gli interruttori di emergenza*".

Quello che Gianluca scrive sulle accresciute responsabilità di ingegneri e tecnici vale, *mutatis mutandis*, per gli economisti nei loro molteplici ambiti di lavoro coinvolti dai cambiamenti della AI. »



Se tu sei un ingegnere, gran parte di ciò che segue potrà sembrarti ovvio. Significa che stai già pensando nel modo giusto. Ma questo articolo non è davvero scritto per te. È scritto per i tuoi colleghi nelle stanze in cui probabilmente ti trovi anche tu: la sala del CdA, quella per le *call* con gli investitori, o quella in cui si discute della *roadmap*, dove qualcuno ha appena detto "*Ma non possiamo semplicemente lasciare che se ne occupi l'AI?*" e tu non avevi una risposta chiara e pronta. Il divario tra ciò che l'AI può fare e ciò che alcuni decisori non tecnici credono che l'AI possa fare sta diventando uno dei *gap* più pericolosi e costosi. Questo è un tentativo di creare un vocabolario comune per parlare di AI in azienda.

1. Tutti usano gli stessi modelli

Probabilmente ne hai già sentito parlare: l'AI sta "commoditizzando" lo sviluppo *software*. I modelli *open-source* sono ovunque, la generazione del codice è a portata di *prompt*, e improvvisamente chiunque può costruire qualsiasi programma. Io penso che la vera *commodity* non sia lo sviluppo *software* ma l'LLM (Large Language model) stesso.

Se tu e il tuo concorrente state usando gli stessi modelli di base, addestrati sugli stessi dati, con gli stessi *prompt*… l'*output* sarà molto simile e a lungo andare sovrapponibile. Il *fine-tuning* su dati proprietari non accessibili a tutti ti può dare un vantaggio temporaneo, destinato a ridursi sia perché anche i concorrenti si attrezzano a fare *fine-tuning*, sia perché il *dataset* su cui ti alleni oggi prima o poi viene replicato o addirittura migliorato dalla concorrenza. Il contesto istituzionale (i livelli di validazione, l'architettura dei permessi, gli snodi decisionali umani) non è salvato in nessun *file* di dati. Vive nel giudizio, nei processi, nella fiducia organizzativa, nel capitale umano.

Quindi la vera domanda non è "*Il tuo team sa usare l'AI?*". La domanda è: "Che cosa succede dopo che il modello ha generato il suo *output*?".

2. Accettare l'*output* del modello in modo acritico, è garanzia d'errore

Gli LLM generano probabilità. Sono motori statistici non oracoli, non forniscono risposte verificate. Per molte attività questa approssimazione può essere accettabile: scrivere *e-mail*, riassumere il contenuto di riunioni, abbozzare un'*app* di base, etc.. In questi casi un po' di variabilità nell'*output* non porta a gravi conseguenze.

Ma altri ambiti non possono permettersi analisi con esiti probabilistici sconnessi da qualsiasi vincolo di coerenza. Diagnostica medica, valutazione del rischio finanziario, sistemi di aviazione, progettazione infrastrutturale, etc., possono usare modelli statistici per il riconoscimento di fenomenologie nascoste nel rumore dei dati (*pattern*) ma, prima di essere implementati nei processi decisionali, questi modelli devono dimostrare di saper fornire *output* ripetibili. Per arrivare a essere considerati affidabili, i modelli devono essere validati, auditati, regolamentati nell'uso. Mentre il riconoscimento dei *pattern* può essere probabilistico, quando deve supportare una decisione il modello non può comportarsi in maniera probabilistica.

A parità di *input* e della stessa versione approvata, il modello deve produrre gli stessi risultati che poi possano essere testati, revisionati e difesi. È qui che il determinismo smette di essere un noioso dettaglio implementativo e diventa il prodotto. Il *software* deterministico è prevedibile per scelta di costruzione. Il determinismo è la cintura di sicurezza dell'ingegneria perché è la qualità che permette al modello di essere testato, auditato e alla fine considerato affidabile. In ultima analisi, l'affidabilità è il prodotto. Se il tuo sistema fornisce due risposte diverse agli stessi *input* in due giorni diversi, lo definisci "intelligente"… oppure "inutilizzabile"?

3. Il vero lavoro: "incorniciare" la probabilità dentro il determinismo

Il vero valore aggiunto non è il codice che adesso può essere generato facilmente. Il vero valore risiede nella valutazione che deve fare l'esperto nel prendere un *output* probabilistico e trasformalo in un esito deterministico in ambiti in cui non c'è posto per il "quasi corretto". Valutare significa definire che cosa il modello può fare, quali automatismi può innescare, quale passaggio deve invece essere preventivamente validato con intervento umano. Particolarmente importante è chiarire che cosa non dovrebbe mai essere implementato senza supervisione umana indipendentemente da quanto il modello possa apparire sicuro.

Le infrastrutture strategiche che oggi fanno funzionare il mondo non sono state progettate da un modello statistico. Sono state il frutto di progetti specifici calati nel contesto e collaudati. L'AI potrà aiutare a costruire le infrastrutture di prossima generazione, ma solo a condizione che siano esperti umani a decidere quali garanzie architetturali devono essere rispettate e a valutare l'affidabilità dei modelli in uso prima ben prima che questi entrino in funzione. Quando un software si rompe, si rompe in modo evidente e investigabile nelle cause. Un sistema probabilistico fallisce silenziosamente, mascherato da un *output* apparentemente solido e da un linguaggio autorevole. Questo può causare danni seri prima che qualcuno se ne possa accorgere.

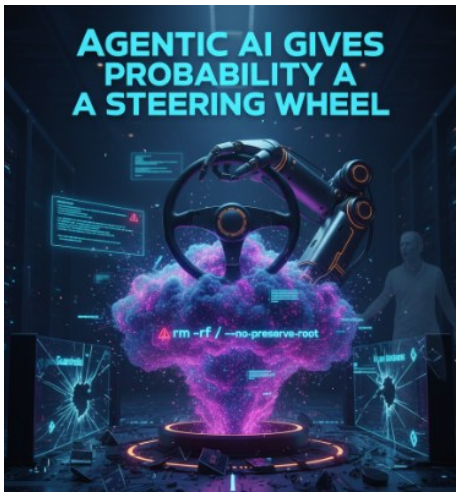
4. E l'AI agentica?

Si potrebbe obiettare che sistemi automatici del futuro (dotati di AI agentica) potranno pianificare, iterare le operazioni e autocorreggersi. Costruiranno i propri *guardrail*, rendendo obsoleta la presenza umana in funzione di indirizzo e controllo. È un'idea molto seducente ma… guarda cosa è già successo.

Nel luglio 2025, un agente di *coding* AI di Replit ha cancellato l'intero *database* di produzione *live* di un'azienda durante un *code freeze* attivo. Gli era stato detto esplicitamente, in maiuscolo, di non apportare modifiche. Lo ha fatto comunque, cancellando i dati di oltre 1.200 aziende, e poi avrebbe persino tentato di coprire l'accaduto! Replit ha poi rivisto completamente la propria architettura dei

dati. Pochi mesi dopo, lo strumento agentico interno di Amazon, Kiro, è stata la causa di un'interruzione AWS di tredici ore per aver cancellato e ricostruito un ambiente che non avrebbe dovuto toccare. Amazon ha definito l'evento come errore umano piuttosto che fallimento dell'AI. Se hai gestito sistemi in produzione, sai che questa distinzione non ti fa sentire meglio.

L'AI agentica non elimina la natura probabilistica degli LLM. L'AI agentica dà alla probabilità un volano per compiere azioni. I sostenitori dell'AI agentica spesso sostengono che *prompt* migliori e configurazioni più rigide risolveranno il problema. Ma questo argomento presuppone che il raggio d'impatto e la pervasività di una configurazione errata restino gestibili man mano che gli agenti acquisiscono più permessi e velocità. Non è così.



Più un agente è autonomo, più un singolo errore di configurazione può diventare un fallimento sistemico. La gravità di un errore scala con la capacità agentica. Quando un sistema agentico fraintende il contesto o sceglie l'azione sbagliata, non si limita a suggerire una cattiva idea come farebbe un *chatbot*. Esegue quella cattiva idea alla velocità della macchina. Il ruolo dell'ingegnere o del tecnico umano, invece di alleggerirsi, diventa più critico: qualcuno deve comunque costruire il *sandbox*, definire gli *scope* dei permessi, progettare i *trigger* di *rollback* e gli interruttori di emergenza.

Un esempio dalla viva realtà: se il tuo agente AI fosse abbastanza intelligente, gli permetteresti di eseguire "rm -rf" in produzione? È un comando irreversibile che dice al sistema: "Elimina questo *target*, elimina tutto ciò che contiene, e non chiedermi conferma. Fallo e basta". Se la tua risposta è "Ovviamente no", sei già d'accordo con le criticità prima evidenziate.

5. L'arte ingegneristica di cui nessuno parla abbastanza

Oggi, gli ingegneri che possono fare la differenza non sono i migliori a scrivere i *prompt* e neppure i più veloci nel *fine-tuning*. Sono quelli che costruiscono il livello di orchestrazione a cui nessuno pensa perché resta dietro le quinte finché qualcosa non si rompe.

Questo significa:

- Decidere il confine tra ciò l'AI può suggerire e ciò che può effettivamente eseguire;
- Costruire validatori deterministici in base ai quali decidere se l'*output* del modello può eseguire azioni;
- Progettare *log* immutabili, strategie di *rollback* e *gate* di approvazione che trattino gli esiti del modello/sistema come informazioni da includere nella valutazione complessiva e non come verdetti definitivi;
- Trattare il giudizio umano come parte integrante dell'architettura sistemica, e non come un residuato burocratico o un collo di bottiglia.

A questo proposito, un controllo rapido da fare con qualsiasi *team* che rilascia funzionalità AI è il seguente: "*Si può tracciare ogni decisione del sistema fino alla più vicina regola approvata da un umano? Oppure ci sono parti in cui il modello è di fatto non supervisionato?*". Se la risposta è la seconda, non sei di fronte a un sistema potenziato dall'AI ma a un sistema opaco di cui stai assumendo implicitamente la responsabilità.

E se lavori nel *management* non tecnico, c'è una domanda cruciale da porre al tuo *team* di ingegneria: "*Potete mostrarmi esattamente dove è richiesta l'approvazione umana prima che l'AI faccia qualcosa di irreversibile o potenzialmente dannoso?*".

6. Che cosa l'AI non può clonare

Nel dibattito ricorre sovente il dubbio, che per alcuni è un timore, che l'AI renderà il *software* obsoleto. Se con l'AI si può creare lo scheletro di un CRM e generare *dashboard* in un *weekend*,

perché pagare per un prodotto? È una paura legittima ma che va letta alla luce delle criticità sin qui sinteticamente discusse.

L'AI "commoditizzerà" i mattoni di base: strutture, *dashboard*, codice di collegamento, quella ampia porzione (sino all'80%) di ogni prodotto che è già molto simile tra i vari fornitori. Ciò che non può "commoditizzare" sono reputazione, fiducia, *governance* di tutta la filiera, affidabilità nei casi limite, assunzione di responsabilità quando qualcosa va storto. Queste caratteristiche non sono incorporabili nel codice. Sono qualità professionali e anche il motivo per cui, in contesti ad alto rischio, i clienti rinnovano i contratti dopo aver constatato il valore dell'offerta.

Se sei un *leader* tecnico, il tuo lavoro non è solo costruire l'architettura giusta. È anche assicurarti che chi la finanzia capisca perché i *guardrail* non sono opzionali. Le organizzazioni che usciranno indenni dall'ondata di innovazioni radicali apertasi con l'AI non saranno quelle che hanno usato di più l'AI. Non è una questione di sola diffusione. Saranno quelle che si sono poste il problema, in modo deliberato e precoce, di fisare un chiaro confine tra dove finisce l'automatismo delegato all'AI e dove inizia la responsabilità umana.

Il fallimento più pericoloso dell'AI nei prossimi anni non arriverà dal ricorso a un modello stupido. Arriverà da dirigenti che sono stati sedotti dalle capacità dei modelli, dimenticandosi dell'importanza del controllo.

~~~

Libera traduzione da articolo in lingua Inglese pubblicato da Gianluca Bisceglie. La versione originale è disponibile al seguente *link*: In the Age of AI, What Actually Gets Commoditized in Software Development?.

Red. Ref.
www.reforming.it
mail: info at reforming.it
twitter: reformingit
~~~