

The background is a vibrant yellow. It is decorated with a complex network of black and white dots of various sizes. These dots are interconnected by thin black lines, some of which are solid and others dashed. The lines and dots form a series of overlapping circles and arcs, creating a sense of movement and connectivity, reminiscent of a network diagram or a stylized orbital system.

LAURENT DÉSÉCHALLIERS

CONTESTO

Create un contesto che permetta all'IA
di comprendere ciò che conta,
di collegare le vostre conoscenze
e di metterle al servizio dei vostri obiettivi.

Le Contexte

Come costruire un contesto che permetta all'IA di capire ciò che conta, collegare le tue conoscenze e far fruttare ciò che impari al servizio dei tuoi obiettivi.

Laurent Deséchalliers

Nota sulla traduzione

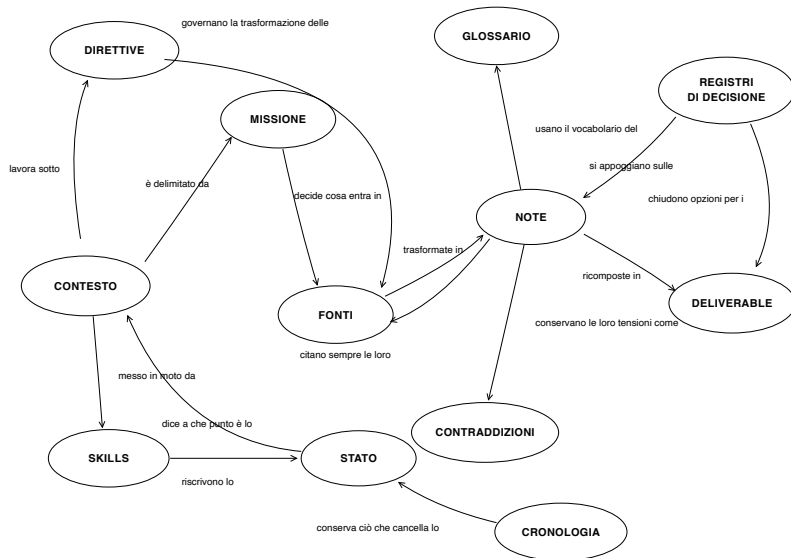
Questa edizione è una traduzione automatica, prodotta da un'intelligenza artificiale a partire dal testo francese.

Non è stata riletta da un traduttore umano. Le idee, la struttura e gli esempi sono fedeli all'originale; alcune formulazioni, invece, possono suonare goffe, e qualche passaggio conserverà la traccia della lingua di partenza.

È il prezzo da pagare perché quest'opera esista in più lingue anziché soltanto in francese. In caso di dubbio su una formulazione, l'edizione francese fa fede.

Definizione del Contesto

Un contesto è un sistema delimitato da una missione umana, nel quale un'IA organizza e mette in campo ciò che si sa, da dove lo si sa, come gli elementi si collegano fra loro e a che punto è il lavoro, per aiutare l'essere umano a raggiungere il suo obiettivo senza ripartire da zero. Così facendo, il sistema costituisce e fa fruttare un capitale informativo duraturo: la conoscenza prodotta resta disponibile, si arricchisce nel tempo e può essere riutilizzata per altri obiettivi.



Introduzione

Non abbiamo mai avuto tante informazioni a disposizione. Eppure continuiamo a spendere gran parte delle nostre energie per ritrovarle, rimetterle in ordine e ricostruire quello che avevamo capito qualche giorno prima.

Un'idea utile arriva di rado in una forma pulita. Nasce da uno scambio, prende forma in un documento, incontra un'obiezione in riunione, produce una decisione, poi diventa un articolo, una specifica o un progetto. I nostri strumenti conservano ognuna di queste tracce. Quello che fanno fatica a conservare è il filo che le lega.

L'intelligenza artificiale cambia una parte del problema. Sa leggere, avvicinare, riformulare e trasformare materiali molto diversi. Ma una conversazione con un'IA resta un momento fragile: l'argomento avanza, la discussione si allunga, e poi bisogna ricordare lo scopo, le decisioni prese, le fonti usate e quello che resta da fare.

È per rispondere a questa difficoltà che ho costruito il Contesto.

Un Contesto è uno spazio di lavoro dedicato a un argomento. Un umano e un'IA vi conservano lo scopo perseguito, le fonti, le idee, le regole e lo stato del lavoro, per produrre risultati senza dover ricostruire tutto a ogni sessione.

L'IA può allora lavorare a partire da una storia reale. Ritrova l'origine di un'idea, ricorda una scelta, segnala una contraddizione, rileva una prova mancante, riprende una bozza dove si era fermata. **Aiuta a organizzare il pensiero e a metterlo alla prova. L'umano conserva la responsabilità del senso e delle decisioni.**

Questo sistema è nato nel mio lavoro quotidiano, da un solo file e da una richiesta molto semplice rivolta a un'IA: organizza quello che sto per dirti. Si è sviluppato per tentativi successivi, e ogni difficoltà rendeva visibile una mancanza. Continua a evolvere: alcune delle soluzioni presentate qui hanno tenuto in più usi, altre restano risposte provvisorie a problemi che incontro ancora.

Questo libro non parte dunque da una teoria compiuta. Parte da problemi vissuti: cinquecento mail diventate impossibili da seguire, idee sparse fra più strumenti, documenti di clienti a cui rispondere senza perdere il lavoro già svolto, testi ripresi dopo un'interruzione.

Si rivolge a chi trasforma informazione in comprensione, in decisioni e in deliverable: product manager, consulenti, redattori, ricercatori, analisti e, più in generale, lavoratori della conoscenza. Propone un modo di fare dell'IA un partner di lavoro nel tempo, mantenendo il controllo sul proprio materiale.

Il libro è organizzato in due libri complementari. **Libro I — Comprendere il** contesto racconta come è nato il sistema e

che cosa cambia nel rapporto con l'IA. **Libro II — Costruire e far vivere il contesto** spiega come costruirlo e usarlo nel tempo.

Si possono leggere uno dopo l'altro oppure separatamente. Il capitolo **In che ordine leggere questo libro** presenta i diversi percorsi possibili.

Nota sugli esempi

Le riflessioni e gli esempi di questo libro nascono da oltre trent'anni di esperienza professionale nel settore informatico. Provengono da situazioni incontrate in progetti e ambienti di lavoro diversi, e da numerosi scambi con product manager, CTO, engineering manager e altri professionisti di varie aziende. Riguardano quindi più organizzazioni e diverse fasi del mio percorso.

Per renderle utili al libro, ho modificato, generalizzato o ricomposto queste situazioni. I casi d'uso, i dati numerici e le metriche illustrano problemi ricorrenti e ciò che ne ho imparato. Sono costruzioni: nessuna descrive un cliente, un prodotto o un'organizzazione reali, né riflette i dati, le pratiche o le posizioni del mio datore di lavoro.

Gli esempi tratti dalla mia vita personale — il condominio, i miei articoli e i miei testi — si basano su situazioni che ho vissuto davvero. Alcuni dettagli che avrebbero potuto identificare delle persone sono però stati modificati.

Le analisi, i metodi e le opinioni espressi in questo libro sono personali e riflettono esclusivamente il mio punto di vista.

Sull'autore

Lavoro nel settore del software da diversi decenni e oggi faccio il product manager. Il mio percorso mi ha portato dallo sviluppo e dall'architettura all'organizzazione dei team, poi al prodotto, alle sue operazioni e al suo marketing.

Quei cambi di mestiere hanno spostato il mio rapporto con l'informazione. Non bastava più conoscere una soluzione tecnica o ritrovare un documento. Bisognava mettere insieme segnali sparsi, capire che cosa dicessero nel loro insieme, scegliere fra più direzioni e produrre testi adatti a lettori molto diversi.

Il mio interesse per l'intelligenza artificiale e per i sistemi distribuiti viene da lontano. Oggi incontra una domanda molto concreta: **come lavorare con un'IA senza lasciarle il controllo delle proprie informazioni, delle proprie decisioni e del proprio pensiero?**

Ho collaborato con la stampa specializzata e ho lavorato come revisore tecnico per M2 Édition ed Eyrolles. Oggi scrivo su malorean.net, di product management, di IA, di metodi di lavoro e di sistemi informativi.

Collaboro anche con *Proposition Marketing*, una rivista indipendente dedicata al marketing strategico e operativo. Scrivere per lettori che vengono dal marketing e dalla direzione d'impresa mi obbliga a spiegare le questioni di

prodotto senza dare per scontato un vocabolario comune. È la stessa esigenza di questo libro.

Mi farebbe molto piacere confrontarmi con altre persone che costruiscono sistemi simili, diversi o complementari:

l'argomento merita di essere discusso, messo alla prova e arricchito da altre pratiche. E anche con chi lavora su questioni vicine all'università — ho passato qualche anno in due centri di ricerca del CNRS, e ritrovare quel mondo sarebbe un piacere.

Laurent Deséchalliers

- laurent@malorean.net
- <https://malorean.net>

Ringraziamenti

I miei primi ringraziamenti vanno a te.

A te che hai avuto l'idea un po' folle di aprire questo libro: un libro un po' più spesso del previsto, che contiene senz'altro qualche ripetizione e parla di un argomento astruso.

Finora ho lavorato al *Contesto* in modo piuttosto solitario.

Spero però che questa solitudine non duri: che l'argomento faccia venire ad altri la voglia di esplorarlo, di discuterlo, di criticarlo e di farlo crescere. Forse i prossimi articoli pubblicati su malorean.net saranno allora l'occasione per ringraziare molti collaboratori e collaboratrici.

Ringrazio soprattutto mia moglie, Christine.

Ha visto le serate e i fine settimana allungarsi davanti a questo libro. Mi ha sostenuto per tutta la scrittura, sapendo al tempo stesso ricordarmi che era tardi e che era davvero ora di chiudere quel maledetto computer.

Promesso, era solo una parentesi.

E, ben oltre questo libro, grazie per accompagnarmi in tutti questi anni nella bella esistenza che condividiamo tutti e due.

In che ordine leggere questo libro

Il libro segue l'ordine in cui le cose mi sono capitate: un problema vissuto, una soluzione raffazzonata e infine un sistema. La descrizione completa del **Contesto** arriva perciò solo nel Libro II.

Il percorso «capire» è proprio quest'ordine. Leggi dall'inizio alla fine, e ogni pezzo del sistema compare nel momento in cui è appena stata mostrata la mancanza che lo ha prodotto.

Il percorso «costruire» si rivolge a chi ha già vissuto il problema e vuole la meccanica, non il racconto.

1. **Definizione del** **Contesto**, qualche riga per sapere di che cosa si parla.
2. **La parte 5 — Sotto il cofano di un** **Contesto**, nell'ordine dei suoi sei capitoli: la mappa, aprire, alimentare, trasformare, produrre e decidere, salvare.
3. **La parte 6 — Il** **Contesto nel suo ecosistema**, per collegare quello che hai appena costruito al resto dei tuoi sistemi informativi.
4. **Il Libro I**, quando una scelta di progettazione ti sembra arbitraria: lì trovi il problema che l'ha provocata.

Questa scorciatoia ha un prezzo. Il Libro II è stato scritto per un lettore che ha già in mente le situazioni del Libro I, e vi fa riferimento. Incontrerai quindi esempi a cui mancherà un po'

di contesto: è il prezzo da pagare, e resta conveniente se sai già perché sei qui.

Provare prima di leggere

Questo libro descrive un metodo, e un metodo si racconta male. Puoi vederne uno funzionare prima di leggerne la teoria, o durante: spesso è il momento migliore per capire a cosa serve un singolo pezzo.

Tre depositi sono aperti su github.com/laurent-malorean.

Contexte Nano Notion è il più rapido da installare. Non richiede né un ambiente di sviluppo né competenze tecniche: bastano un ChatGPT e un Notion. ChatGPT porta l'intelligenza del `Contesto`, Notion gli fa da memoria: è lì che le fonti, le note, le decisioni e lo stato si scrivono e si rileggono da una conversazione all'altra. In pochi minuti hai un `Contesto` che regge.

Contexte V2 è l'implementazione completa, quella che uso tutti i giorni e con cui è stato scritto questo libro. Si installa con Claude Code o con Codex, a scelta, e chiede un po' più di preparazione della precedente.

LLM Wiki non è un `Contesto`. È una biblioteca di conoscenze costruita a partire da documenti — libri, norme, articoli, corsi — da cui l'IA ricava schede brevi e collegate fra loro, che poi si interrogano in lingua corrente. Il `Contesto` tiene il filo di una missione; l'LLM Wiki conserva in massa ciò che è stato letto. La parte 6 spiega come i due lavorano insieme.

Nulla obbliga a cominciare da lì. Ma se a un certo punto la lettura ti sembra astratta, è la via più breve perché torni concreta.

Dello stesso autore



Malorean Corp — Fino a poco tempo fa, trasformare un’idea in un prototipo poteva richiedere settimane. Oggi, con l’aiuto dell’intelligenza artificiale, una persona può riuscirci in pochi giorni. È un cambiamento straordinario. Ma la vera trasformazione potrebbe avvenire altrove: nel modo in cui lavoriamo insieme.

Uscita prevista: fine novembre 2026.

Maggiori informazioni su <https://malorean.net>.

Indice

LIBRO I

Comprendere il Contesto

Parte 1 — La nascita di un bisogno	21
1.1 Cinquecento mail e una storia che sparisce	22
1.2 Una prima definizione	31
Parte 2 — Che cosa cambia il Contesto	39
2.1 Dalla conversazione a un lavoro che si può riprendere	40
2.2 Due clienti, un white paper e una contraddizione da conservare	48
Parte 3 — Tre usi concreti	59
3.1 Rispondere a un questionario del cliente senza perdere il lavoro fatto	60
3.2 Scrivere un articolo a partire da un pensiero ancora informe ...	66
3.3 Far maturare un tema di prodotto in mezzo al caos	74
Parte 4 — Che cosa si lasciano dietro i deliverable	85
4.1 La risposta sta in una cella, il ragionamento no	86
4.2 Sei mesi dopo, la risposta non basta più	90

LIBRO II

Costruire e far vivere il Contesto

Parte 5 — Sotto il cofano di un Contesto 97

5.1 La mappa completa 98

5.2 Aprire: ritrovare la missione e il presente 109

5.3 Alimentare: far entrare il materiale 120

5.4 Trasformare: passare dalle fonti alle idee 127

5.5 Produrre e decidere 137

5.6 Salvare e riprendere 147

Parte 6 — Il Contesto nel suo ecosistema 155

6.1 Interrogare le conoscenze strutturate 156

6.2 Interrogare il prodotto reale 163

6.3 Il giorno in cui ho avuto tre Contesti 172

6.4 Leggere dal vicino senza trasferirsi da lui 178

6.5 Quanto costa la circolazione 185

Prima di chiudere

Chiusura 193

7.1 Epilogo 194

7.2 Parentele e vicinanze 197

LIBRO I

Comprendere il Contesto

- 1 La nascita di un bisogno
- 2 Che cosa cambia il Contesto
- 3 Tre usi concreti
- 4 Che cosa si lasciano dietro i deliverable

Parte 1

La nascita di un bisogno

Il Contesto non è nato da un'idea sull'intelligenza artificiale. È nato da una casella di posta diventata ingestibile e da una consapevolezza maturata poco a poco: il vero problema non era il numero dei messaggi.

Questa parte racconta quell'origine e ne ricava una prima definizione: abbastanza semplice da ricordare e abbastanza completa da guidarci nel resto del libro.

Cinquecento mail e una storia che sparisce

La prima volta che ho creato un `Contesto` non cercavo di inventare un metodo di lavoro con l'intelligenza artificiale. Cercavo soltanto di non perdere più il filo del mio condominio. Faccio parte del consiglio di condominio. Per questo mi arrivano i messaggi dell'amministratore, dei condòmini, degli inquilini e delle ditte:

- una mail annuncia un problema nuovo,
- quella dopo definisce un preventivo,
- una terza rinvia un intervento,
- una settimana più tardi qualcuno risponde nello stesso scambio parlando di un altro edificio.

Nel frattempo altri tre argomenti sono andati avanti.

Il volume non era l'unico problema. Circa cinquecento mail si distribuivano su una quindicina di argomenti:

- lavori,
- conti,
- assicurazioni,
- assemblee,
- morosi,
- gare,
- guasti e richieste varie

- ...

Ognuno andava avanti al proprio ritmo, a volte per mesi.

Chi segue un cantiere ogni giorno ne porta in testa la storia senza sforzo. Sa quale ditta è intervenuta, perché il calendario è cambiato e quale decisione aspetta ancora una risposta.

Quando aprivo un messaggio nuovo, quella continuità non ce l'avevo sempre. Dovevo risalire lo scambio, ritrovare le mail precedenti e rimettere insieme la situazione prima ancora di poter capire l'informazione nuova.

Volevo poter fare domande semplici:

- a che punto è il tetto?
- quale ditta ha mandato l'ultimo preventivo per le grondaie?
- che cosa abbiamo deciso su quell'infiltrazione?
- che cosa aspettiamo ancora dall'amministratore?

La mia casella di posta conteneva le risposte. Quello che non mi dava era la visione d'insieme.

Un file e una frase

Ho cominciato con la sistemazione più semplice possibile: un file e un'istruzione data a un'IA.

▮ *Metti insieme le informazioni e dividile per argomenti.*

Le passavo le mail. Le associava agli argomenti e teneva un riassunto di ciascuno.

- Tutto quello che riguardava i lavori finiva insieme.
- Le questioni di soldi formavano un altro blocco.

- Le ditte, le assicurazioni e le assemblee trovavano via via il loro posto.

1

Già quel primo tentativo dava qualcosa. Potevo chiedere a che punto fosse un argomento senza rileggere tutti i messaggi che lo riguardavano. L'IA non cercava più soltanto una corrispondenza nella casella: mi presentava quello che aveva capito della situazione.

Il file si riorganizzava man mano che arrivavano nuove informazioni.

- Compariva una voce quando un argomento nuovo prendeva peso.
- Due voci si separavano quando cominciavano a mescolare realtà diverse.

L'informazione non si limitava più ad accumularsi. Prendeva una forma con cui si poteva lavorare.

Poi quel primo sistema ha dato una risposta esatta e comunque insufficiente.

Il cantiere era finito, ma la sua storia era sparita

Quando un cantiere arrivava alla fine, il suo riassunto diceva: «lavori conclusi, collaudo regolare».

Era davvero il suo stato presente. Era anche una perdita.

- Quando erano cominciati i lavori?
- In che momento erano stati rinviati?
- Perché il calendario era cambiato?
- Quando era stata firmata la variante?

Tutte quelle tappe avevano contato. L'ultimo aggiornamento le aveva cancellate a favore della situazione finale.

Avevo chiesto un riassunto. L'IA lo aveva tenuto aggiornato sostituendo il vecchio stato con quello nuovo. Aveva seguito l'istruzione, ma l'istruzione non proteggeva la storia.

Mi servivano dunque due viste dello stesso argomento.

- La prima doveva rispondere in fretta alla domanda «a che punto siamo?».
- La seconda doveva conservare i fatti nel loro ordine: inizio dei lavori, ritardo, nuovo accordo, intervento, collaudo.
- Lo stato presente permetteva di agire.
- La storia permetteva di capire come si fosse arrivati a quello stato e di verificare un punto in caso di dubbio.

Quella distinzione è nata da un difetto molto concreto. La prendevo per un miglioramento dell'archiviazione. È diventata uno dei principi più importanti del mio lavoro con l'IA: **ciò che è vero adesso e ciò che è successo non svolgono la stessa funzione.**

L'IA non poteva indovinare le mie regole

Un altro problema è comparso con i nomi:

- due ditte avevano lo stesso cognome.
 - Per distinguerle bisognava capire l'argomento della mail.
- una condòmina compariva ora con il cognome da nubile, ora con quello del marito.

Senza un'indicazione precisa, l'IA poteva creare due persone dove ce n'era una sola.

Altre differenze venivano dal modo di lavorare.

- I morosi sotto i cento euro non venivano seguiti come gli altri.
- Quel limite non veniva da nessuna regola universale che l'IA potesse conoscere.
- Veniva dal nostro modo di fare.

Ho cominciato a formulare quelle conoscenze come regole durature, scritte per esteso nel file:

Regole di trattamento

- Martin citato a proposito del tetto indica il lattoniere;
Martin citato a proposito
del verde indica il giardiniere.
- Dupont-Lemoine e Lemoine indicano la stessa condòmina,
unità 14.
- Presentare i morosi sotto i cento euro in un elenco
separato.

Niente di più sofisticato. Tre frasi che avrei dovuto ripetere a ogni conversazione e che non ho più dovuto dire nemmeno una volta.

Ogni regola veniva da un errore osservato o da un'ambiguità reale. Fissava una conoscenza che tenevo in testa e che l'IA non poteva ricostruire con certezza.

Così il sistema cominciava a conservare due tipi di informazioni:

- le informazioni sul condominio,

- il modo in cui volevo che fossero trattate.

Il file che aiutava ha cominciato a dare fastidio

Con le settimane il file è cresciuto. Ha superato le cinquecento righe, poi le ottocento, poi le mille.

Quella crescita dimostrava la sua utilità:

- gli argomenti erano seguiti meglio,
- le ditte identificate,
- le decisioni ritrovate.

Ma quello che aveva reso possibile il sistema cominciava a limitarne l'uso.

- Troppe informazioni diverse dividevano lo stesso spazio.
- Le regole si mescolavano ai fatti.
- I conti stavano accanto ai lavori, alle persone, alle assicurazioni e alle gare.

Trovare le informazioni giuste richiedeva di nuovo fatica.

Per prima cosa ho separato le regole di trattamento. Non descrivevano il condominio; dicevano come lavorare con le informazioni che lo riguardavano. Isolarle permetteva di ritrovarle e di applicarle più facilmente.

Poi ho distribuito le informazioni fra più argomenti. I lavori hanno avuto il loro spazio, come i conti, le ditte, le delibere d'assemblea o le assicurazioni. Una vista d'insieme diceva che cosa contenesse ciascuno. L'IA poteva allora aprire l'argomento pertinente alla domanda, invece di scorrere ogni volta tutto il contenuto.

Infine ho raccolto il vocabolario particolare: termini giuridici, sigle ed espressioni proprie del condominio. Lo scopo non era costruire un'enciclopedia. Era dare un senso stabile alle parole da cui dipendeva la comprensione degli scambi.

Continuavo a non avere in testa nessuna grande architettura. Aggiungevo un pezzo quando un problema rendeva quel pezzo necessario.

Fidarsi senza perdere la fonte

L'IA a volte sbagliava. Poteva accostare due informazioni che non riguardavano esattamente lo stesso argomento o scegliere la ditta sbagliata fra due omonime. Il sistema doveva dunque accelerare la comprensione senza diventare l'unica versione disponibile dei fatti.

Ho conservato tutte le mail originali. Quando dovevo verificare una risposta o quando la posta in gioco aumentava, potevo tornare alle informazioni nella loro forma originale. Il riassunto dava un accesso rapido. La fonte permetteva di controllare.

Quella scelta è ciò che rendeva possibile l'uso. Potevo accettare qualche errore occasionale nel modo in cui le mail erano organizzate, perché i messaggi originali restavano accessibili. Lo stesso grado di fiducia non andrebbe bene per un documento bancario, per un obbligo normativo o per una decisione irreversibile. Il rischio dipende sempre da ciò che si affida al sistema e dalla possibilità di verificare.

Il Contesto non sostituiva dunque la mia casella di posta. Trasformava il suo contenuto in una memoria di lavoro, lasciandomi una porta di ritorno verso i messaggi originali.

Che cosa era comparso

In poche settimane il punto di partenza — un file e una frase — era diventato un insieme capace di rispondere a più bisogni: mostrare lo stato presente, conservare la storia, applicare regole proprie del condominio, distinguere gli argomenti, fissare il vocabolario e tornare alle fonti. Quando arrivava una mail nuova, potevo occuparmi di ciò che cambiava invece di ricostruire tutto quello che l'aveva preceduta. Quando spuntava una domanda, potevo porla nei termini del problema invece di cercare il messaggio giusto nello scambio giusto. Quando l'IA sbagliava, potevo correggere una regola o tornare al testo d'origine.

Non avevo progettato il Contesto in un colpo solo. L'avevo visto comparire attraverso una serie di insufficienze: il riassunto cancellava la storia, i nomi si confondevano, le regole si perdevano, il file diventava troppo grande. Ogni correzione rendeva il lavoro più duraturo e più facile da riprendere.

Quell'origine conta. Il Contesto non è nato dalla voglia di classificare di più. È nato dal bisogno di capire informazioni che cambiano, di conservare le prove necessarie per verificarle e di portare avanti un lavoro senza ripartire da zero.

DA RICORDARE — DUE VISTE DELLO STESSO ARGOMENTO

Lo stato risponde a «a che punto siamo?». La storia risponde a «come ci siamo arrivati?». Un riassunto aggiornato schiacciando il precedente conserva la prima e perde la seconda. Servono entrambe: una per agire, l'altra per capire e verificare.

Una prima definizione

Il sistema nato da quelle cinquecento mail può adesso ricevere una prima definizione.

Un Contesto è uno spazio di lavoro dedicato a un argomento, dove una persona e un'IA conservano ciò che serve per capire, prendere decisioni e produrre risultati nel tempo.

Questa definizione resta volutamente semplice. Il Contesto descritto in questo libro ha un'organizzazione precisa, che la seconda metà dettaglierà. Per capire che cosa porti, all'inizio bastano sei riferimenti:

- un obiettivo,
- informazioni,
- delle idee,
- uno stato del lavoro,
- delle regole proprie,
- dei risultati da produrre.

Un argomento e un obiettivo

Il Contesto del condominio non cercava di contenere tutto quello che sapevo. Perseguiva un obiettivo: capire gli argomenti in corso, conservarne la storia e ritrovare le informazioni necessarie per agire nel consiglio di condominio. Quell'obiettivo formava un confine. Una mail sul tetto aveva il suo posto. Una lettura interessante senza rapporto con il

condominio no. Quel limite evitava al sistema di diventare un deposito dove si accumula ogni informazione giudicata interessante.

1

Un `Contesto` comincia dunque da un'intenzione. Può servire a:

- scrivere un libro,
- esplorare un problema di prodotto,
- preparare una decisione,
- rispondere a questionari del cliente,
- seguire un'entità di business,
- ...

Informazioni di cui si conserva l'origine

Le informazioni entrano in un `Contesto` in due modi.

Dalla prima passa ciò che esiste già prima della sessione di lavoro:

- una mail,
- un articolo,
- un libro,
- una pagina web,
- la trascrizione di un'intervista
- ...

Sono le **fonti documentali**. Arrivano come oggetti già formati e restano stabili. Nel caso del condominio, la mail che annuncia il rinvio di un intervento appartiene a questa prima categoria.

Dalla seconda passa ciò che si formula durante il lavoro con l'IA. Quando spiego una situazione, correggo

un'interpretazione, affino una sfumatura o esprimo una convinzione nata dalla mia esperienza, quello che dico diventa una **fonte di sessione**. Si costruisce a poco a poco nel corso dello scambio. Quando spiego che due nomi indicano la stessa condòmina, quello che dico entra prima di tutto da questa seconda porta.

La differenza sta dunque nel momento e nel modo in cui le informazioni compaiono.

- Il documento esiste prima della sessione e viene introdotto al suo interno.
- La fonte di sessione nasce durante la sessione e cresce con essa.

Poi entrambe svolgono la stessa funzione: conservare ciò che è stato ricevuto o detto prima di interpretarlo.

- Non garantiscono che tutto quello che contengono sia vero.
- Permettono di ritrovare ciò che è stato ricevuto o detto, e poi di valutarne l'affidabilità.

Da queste fonti il *Contesto* può poi ricavare un'idea, una regola o uno stato del lavoro. La conoscenza ottenuta interpretando e organizzando le informazioni viene conservata altrove. Se una data sembra sbagliata, se due persone sono state confuse o se un'interpretazione diventa discutibile, si può così tornare alla sua origine.

Idee rese utilizzabili

Accumulare fonti non basta. Cinquecento mail ben conservate restano cinquecento mail da leggere.

Il Contesto trasforma dunque a poco a poco quelle fonti in conoscenza utilizzabile. Ne fa emergere:

- argomenti,
- fatti importanti,
- regole,
- distinzioni,
- domande aperte,
- relazioni.

Uno scambio isolato entra nella storia del cantiere a cui appartiene. Più citazioni di una stessa ditta arricchiscono una conoscenza comune. Una correzione impedisce a una confusione di ripetersi.

Questa tappa intermedia conta perché rende utilizzabile l'informazione.

- Le fonti conservano ciò che è stato ricevuto o detto.
- La conoscenza organizzata permette di capire e di agire.

In lavori più complessi, certe idee potranno servire più volte: una stessa conoscenza di prodotto alimenta una decisione, una specifica e una documentazione; uno stesso fatto verificato aiuta a rispondere a più questionari del cliente. Il risultato cambia, mentre l'idea e la sua provenienza restano disponibili.

Uno stato che permette di continuare

Un lavoro in corso ha sempre uno stato attuale: quello che si sa, quello che resta incerto, quello che è stato deciso e quello che deve succedere dopo.

Nel condominio, lo stato rispondeva a una domanda immediata: a che punto è oggi questo cantiere? La sua storia rispondeva a un'altra: come ci siamo arrivati? Il sistema aveva bisogno di entrambe perché potessi davvero riprendere il lavoro.

Questa distinzione va oltre il seguire un cantiere. Quando interrompo la scrittura di un testo o l'esplorazione di un problema di prodotto, voglio ritrovare più di una raccolta di documenti. Voglio sapere:

- che cosa è assodato,
- quale punto resta fragile,
- quale azione può rimettere in moto il lavoro.

Il Contesto conserva quella continuità fuori dalla conversazione.

- La sessione con l'IA può fermarsi.
- Il lavoro conserva uno stato da cui la sessione successiva potrà ripartire.

Regole proprie del lavoro

L'IA dispone di conoscenze generali. Ignora invece gran parte di ciò che rende particolare una situazione:

- le due ditte che hanno lo stesso nome,
- il modo in cui un team classifica un segnale che viene dal cliente,
- il livello di prova richiesto prima di pubblicare un'affermazione
- il tono scelto per un libro,
- ...

Il *Contesto* tiene quelle regole insieme al lavoro che governano. Possono essere corrette quando l'esperienza ne mostra il limite. Si applicano poi senza che l'essere umano debba riformularle in ogni richiesta.

Quelle regole non danno all'IA la responsabilità del lavoro. Rendono esplicito ciò che l'essere umano si aspetta da lei. La persona fissa l'obiettivo, porta o convalida le fonti, corregge le interpretazioni e prende le decisioni. L'IA acquisisce le informazioni, le collega e le organizza, pone domande, riformula e prepara risultati entro quei limiti.

Risultati adatti a un bisogno

Il *Contesto* resta concentrato su ciò che deve rendere possibile. Nel condominio, il risultato poteva essere:

- una risposta a una domanda,
- un quadro dei lavori,
- la preparazione di una discussione.

In altri usi sarà:

- un articolo,
- un questionario del cliente,
- una documentazione.

Lo stesso spazio può produrre più risultati perché conserva le fonti prima che assumano una forma definitiva. Le fonti non restano chiuse nel primo documento che le usa. Le idee restano disponibili per un'altra domanda o per un altro pubblico.

Questa capacità distingue il *Contesto* da un semplice archivio.

- Un archivio permette di conservare.
- Il `Contesto` organizza il modo in cui ciò che è stato conservato viene ripreso e trasformato.

Una forma che può cominciare in modo molto semplice

Nella mia pratica, un `Contesto` prende la forma di file che posso leggere, spostare e conservare indipendentemente dallo strumento che mi aiuta a lavorarci. Quella forma risponde al mio bisogno di controllo e di durata. Altre realizzazioni sono possibili, se preservano le stesse funzioni.

Il primo `Contesto` non aveva però niente di sofisticato. Stava in un file. È cresciuto quando i suoi limiti sono diventati visibili. Quella progressione resta importante: non serve costruire tutto il sistema descritto in questo libro per cominciare a mettere alla prova l'idea.

Bastano un argomento, un obiettivo e alcune fonti iniziali.

- L'IA aiuta a organizzarli.
- L'essere umano osserva che cosa manchi, aggiunge una regola, conserva una fonte o separa due informazioni che non hanno lo stesso ruolo.
- L'architettura compare al ritmo dei problemi incontrati.

Il `Contesto` indica dunque non tanto una particolare struttura di cartelle, quanto un rapporto di lavoro duraturo fra una persona, un'IA e un insieme di informazioni organizzato intorno a un obiettivo.

La parte seguente esaminerà che cosa cambi quel rapporto. Una conversazione con un'IA può già essere feconda. Il

passaggio al *Contesto* comincia quando quella conversazione si svolge in un ambiente capace di conservarne l'obiettivo, la storia e lo stato del lavoro oltre il momento presente.

1

DA RICORDARE — I SEI RIFERIMENTI DI UN CONTESTO

Un obiettivo che traccia un confine, informazioni di cui si conserva l'origine, idee rese utilizzabili, uno stato che permette di continuare, delle regole proprie e risultati da produrre. Questi sei riferimenti bastano per cominciare. L'architettura viene dopo, quando un problema la rende necessaria.

Parte 2

Che cosa cambia il Contesto

Una conversazione con un'IA può già offrire molto. Spiega, riformula, propone, e spesso dà una risposta migliore di quella che si sarebbe trovata da soli.

Questa parte si chiede dunque che cosa aggiunga un Contesto. La risposta non sta nella qualità delle risposte ottenute, ma in ciò che rimane da una sessione all'altra: un lavoro che riparte da dove si era fermato e documenti che mantengono i loro disaccordi invece di smussarli.

Dalla conversazione a un lavoro che si può riprendere

Una conversazione con un'IA può cominciare in modo molto semplice.

Le do un'idea ancora confusa. Lei la riformula. Chiedo una scaletta, poi un'altra. Mi fa un'obiezione a cui non avevo pensato. Preciso la mia intenzione, scarto una pista, ne approfondisco un'altra. In pochi minuti la discussione fa avanzare qualcosa che forse sarebbe rimasto a lungo davanti a una pagina bianca.

È uno dei grandi pregi della chat: basta una domanda per cominciare. Per un compito breve questa forma va benissimo. Chiedo una sintesi, una spiegazione, qualche possibilità. La risposta arriva, la uso, e la conversazione può finire.

Il rapporto cambia quando il lavoro dura.

Cinque frasi prima di poter scrivere

Riapro una conversazione di sei giorni prima per riprendere un testo cominciato la settimana precedente. Prima di poter scrivere una sola riga, digito questo:

Riprendi il testo sulla qualità dei rapporti di intervento. Si rivolge a responsabili delle operazioni, non a sviluppatori. Avevamo scartato l'idea di accorciare il modulo, è troppo presto per decidere. Il dato sugli audit viene da una sola

intervista, trattalo come un'ipotesi. Eravamo alla terza sezione.

Cinque frasi per rimettere l'IA dentro un lavoro che sei giorni prima aveva già davanti.

Tutto quello che ho appena digitato esisteva prima che lo digitassi: l'angolatura, il pubblico, la decisione, il punto fragile e la posizione nel testo stavano nella conversazione precedente. Li ho semplicemente recuperati dalla mia memoria, perché la conversazione li aveva conservati nell'ordine cronologico in cui erano emersi.

Il lavoro segue una logica diversa.

- Un'istruzione data all'inizio deve valere ancora alla fine.
- Una fonte aggiunta ieri deve poter sostenere un paragrafo scritto domani.
- Una decisione vecchia deve impedire che la stessa discussione si riapra.
- Un punto in sospeso deve ricomparire al momento giusto.

Più la conversazione cresce, più l'essere umano deve tenere in testa quell'ordine. Quando comincia una nuova sessione, ricostruisce la situazione per l'IA: ecco l'obiettivo, ecco che cosa è cambiato, ecco il tono, ecco le decisioni, ecco dove eravamo.

La chat resta utile. Il peso della continuità grava però su chi la usa.

Dare un luogo a ciò che deve durare

Un Contesto cambia il rapporto dando al lavoro uno spazio distinto dalla conversazione.

La discussione resta il luogo dello scambio. Lì si interroga, si esplora, si chiarisce e si corregge. Ciò che deve durare passa nello spazio di lavoro:

- l'obiettivo,
- le fonti,
- le regole,
- le idee fissate,
- le decisioni,
- i testi,
- il punto in cui è il lavoro.

La differenza può sembrare sottile. È paragonabile a quella fra una riunione e il progetto di cui parla. Una buona riunione può far avanzare il progetto, ma il progetto non sta tutto nella memoria della riunione. I documenti, le decisioni e le azioni continuano a esistere dopo la sua fine.

Conservare la trascrizione completa della chat non produrrebbe lo stesso effetto. È preziosa quando si vuole ritrovare esattamente quello che è stato detto, ma segue la forma della conversazione: una domanda, una risposta, una correzione, una divagazione, un ritorno all'argomento. Per riprendere il lavoro bisognerebbe ancora rileggere quella sequenza e decidere che cosa conti adesso.

Il Contesto, invece, trasforma quello che è stato detto.

- Una correzione va dove potrà impedire il ritorno dell'errore.
- Una decisione resta visibile come decisione.
- Una fonte resta legata a ciò che sostiene.
- Un testo esiste nella sua ultima versione completa.

La memoria viene così organizzata per facilitare l'azione. Non cerca soltanto di conservare il passato. Prepara l'intervento successivo.

Una pagina per ritrovare trentadue testi

Ho misurato che cosa cambia tornando al lavoro su questo libro.

Il manoscritto contava allora trentadue testi. Alcune parti erano scritte, altre ancora vuote. Una sezione era stata divisa perché occupava uno spazio sproporzionato. Due passaggi aspettavano la mia rilettura. Un'affermazione non poggiava su alcun esempio concreto e andava rifatta.

Presi uno per uno, nessuno di questi elementi è difficile da capire. Insieme formano però una situazione che bisogna ricostruire prima di poter scrivere una sola riga utile.

Potrei aprire i trentadue testi, rileggere la scaletta, scorrere le mie vecchie conversazioni e cercare le osservazioni rimaste in sospeso. Quel lavoro di archeologia prenderebbe tempo e lascerebbe comunque un dubbio: avrò dimenticato proprio il punto che doveva impedire un errore?

Invece apro il *Contesto* del libro. Mi porge un file di ripresa, che quella mattina diceva più o meno questo:

Ripresa

A che punto siamo

Esistono trentadue testi, dalla scaletta alla conclusione.
Le parti tecniche sono scritte; le prime due restano troppo teoriche.

Che cosa sapere prima di continuare

Il lettore a cui pensiamo scopre l'argomento: perde il filo appena un concetto arriva prima di un esempio concreto.
L'affermazione sul fallimento dei sistemi documentali non poggia ancora su nessuna esperienza personale.

La prossima azione

Riscrivere l'apertura intorno al caso delle cinquecento mail del condominio.

Che cosa è rimasto aperto

Due passaggi aspettano la mia rilettura.
Diverse fonti sono entrate senza essere state trasformate in note.

Una pagina, letta in meno di un minuto. So che cosa fare, e non ho aperto nessuno dei trentadue testi.

Continuare non è riprendere

Continuare vuol dire conoscere l'azione successiva.

Riprendere vuol dire ritrovare il senso del lavoro prima di compiere quell'azione.

Un elenco di compiti può dirmi che una fonte va trattata. Non mi spiega perché quella fonte venga prima, quale parte del libro debba illuminare né quale rischio debba permettere di correggere.

Per riprendere mi serve un insieme più ricco:

- l'obiettivo che il lavoro persegue;

- il punto a cui è arrivato davvero;
- le decisioni che orientano ancora il seguito;
- i punti fragili che non vanno dimenticati;
- una prossima azione abbastanza precisa da poter ricominciare.

Il file di ripresa si legge del resto in quest'ordine. «Trentadue testi» dà una misura. «Perde il filo appena un concetto arriva prima di un esempio concreto» indica una direzione. La prossima azione discende da entrambe e nomina il caso preciso che deve aprire il libro.

Quel che è stato recuperato è un ragionamento, non soltanto una posizione in un elenco.

Non per questo cerca di ricreare nella mia testa ogni dettaglio della sessione precedente. Un riassunto troppo corto cancella la ragione di una scelta. Un resoconto troppo lungo rimanda il momento in cui il lavoro ricomincia.

Un'interruzione non sceglie il suo momento

All'inizio immaginavo la ripresa come la conseguenza di una chiusura pulita. Una sessione sarebbe finita, il suo stato sarebbe stato riassunto con cura e la prossima azione annotata.

La pratica segue di rado quel copione.

- Comincia una riunione.
- La batteria si scarica.
- La giornata finisce prima del previsto.
- Un'altra urgenza prende il posto dell'argomento.

Il momento in cui quella memoria è più utile è proprio quello in cui non si è preparata la fine.

Il Contesto mantiene dunque la propria capacità di ripresa durante il lavoro. Le decisioni trovano il loro posto quando vengono prese, i compiti quando vengono formulati, le fonti quando entrano.

2

Riprendere diventa una proprietà del sistema invece che una disciplina eroica dell'utente.

L'essere umano mantiene però un ruolo decisivo. Può giudicare che la prossima azione proposta non sia più la priorità, che una vecchia decisione vada riaperta o che un fatto nuovo cambi l'obiettivo. La memoria non blocca la traiettoria. Rende visibile la traiettoria da cui si decide di cambiarla.

Le cinque frasi che ridigitavo riaprendo la mia conversazione, non le ridigito più. Sono scritte da qualche parte, e qualcuno che non sono io le rilegge al posto mio all'inizio di ogni sessione.

DA RICORDARE — LA CHAT, L'AMBIENTE E LA RIPRESA

La chat è il luogo dello scambio; il *Contesto* è il luogo di ciò che deve durare. Una conversazione conserva le informazioni nell'ordine in cui sono state dette; il lavoro richiede un'organizzazione diversa. Continuare è conoscere l'azione successiva; riprendere è ritrovare il senso del lavoro prima di compierla. Un'interruzione non avvisa: la capacità di riprendere si mantiene durante il lavoro, non al momento di chiuderlo.

Due clienti, un white paper e una contraddizione da conservare

Finora ho descritto il `Contesto` e diversi effetti che produce. Ma com'è fatto davvero quando lo si apre? Che cosa si vede nei file? Come fanno più fonti a far emergere idee che possono essere giuste separatamente e comunque difficili da conciliare?

L'argomento di questo `Contesto` inventato è semplice: capire perché alcuni tecnici consegnano rapporti di intervento incompleti, prima di decidere se il prodotto debba cambiare. La sua missione sta in poche righe:

Capire che cosa peggiori la qualità dei rapporti di intervento. Distinguere le difficoltà di compilazione, i vincoli organizzativi e i bisogni che nascono dall'uso dei dati. Non scegliere una soluzione prima di aver messo a confronto più fonti.

Quella missione dà una direzione al lavoro. Chiede di capire un problema, non ancora di progettare una funzionalità.

Una prima testimonianza rivela un'idea

Il primo documento versato nel `Contesto` è la trascrizione di un'intervista con una responsabile delle operazioni.

Spiega:

«I tecnici compilano spesso i loro rapporti il venerdì. Sul campo il modulo sembra loro troppo lungo. A fine settimana hanno dimenticato una parte dei dettagli.»

L'IA conserva quell'intervista come **fonte**. Poi ne estrae un'idea in una scheda a parte, quella che chiamo una **nota atomica**:

Una compilazione troppo esigente può portare alcuni tecnici a rimandare il rapporto e a perdere dei dettagli.

Quella formulazione non dice che tutti i tecnici si comportino così, né che la lunghezza del modulo sia l'unica causa. Conserva il centro della testimonianza: nell'organizzazione descritta la compilazione viene rimandata e una parte dell'informazione sparisce fra l'intervento e la sua stesura. La fonte resta disponibile. La nota non la sostituisce; rende la sua idea principale più facile da ritrovare e da confrontare con il seguito.

La seconda testimonianza arricchisce la stessa nota

Qualche settimana dopo entra nel `Contesto` una seconda intervista.

Questa volta un responsabile delle operazioni racconta:

«Abbiamo reso obbligatori dodici campi per migliorare la qualità dei dati. Adesso certi tecnici scrivono «nulla da segnalare» o mettono un trattino solo per poter confermare.»

Quella testimonianza non descrive esattamente lo stesso comportamento. I tecnici della prima intervista rimandano la compilazione. Quelli della seconda compilano il modulo, ma ne aggirano l'esigenza.

Le due fonti riguardano però la stessa idea centrale: un vincolo pensato per ottenere più informazioni può finire per ridurre la qualità reale di ciò che viene inserito.

La seconda intervista raggiunge dunque la prima **nota atomica**. Non sostituisce la prima fonte e non crea un'idea concorrente. Arricchisce la nota facendo emergere un'altra forma dello stesso problema: dopo la compilazione rimandata, la completezza di facciata.

La nota può adesso affermare:

Una compilazione troppo esigente può portare alcuni tecnici a rimandare o ad aggirare il rapporto, a scapito dell'informazione raccolta.

Le fonti referenziate sono due. Ciascuna conserva il proprio ruolo e i propri limiti. La nota diventa più solida senza pretendere che due testimonianze bastino a misurare la portata del fenomeno.

Il white paper rivela un'altra idea

Aggiungo poi un white paper dedicato all'uso dei dati di intervento.

Il documento difende un'altra esigenza. Per capire i guasti ricorrenti, confrontare gli impianti e preparare certi audit,

un'azienda ha bisogno di rapporti abbastanza strutturati. Se ogni tecnico scrive quello che vuole in un campo libero — o non compila nulla —, la compilazione diventa forse più facile, ma i dati perdono gran parte della loro utilità collettiva.

Quella **fonte** non rende meno vere le testimonianze precedenti. I tecnici descritti continuano a rimandare o ad aggirare. Il PDF rivela un'altra realtà: eliminare ogni vincolo avrebbe comunque un costo.

L'IA potrebbe essere tentata di fondere quegli elementi nella prima nota e di riscriverla come un compromesso generale. Sarebbe un errore. Il white paper non ridimensiona quanto osservato nelle interviste; presenta un'affermazione diversa. Produce dunque una seconda **nota atomica**:

Rapporti strutturati sono necessari per confrontare gli interventi e sfruttare i dati di intervento.

Il Contesto contiene ormai due idee distinte:

- il vincolo di compilazione può peggiorare l'informazione davvero raccolta;
- l'assenza di struttura può peggiorare l'uso successivo di quell'informazione.

Le due note possono essere vere nello stesso tempo. Tirano però la decisione di prodotto in direzioni opposte.

L'opposizione deve restare visibile fra le note

Il ruolo del Contesto non è modificare la prima nota finché la contraddizione sparisce. Deve conservare entrambe le proposizioni e rendere esplicita la loro relazione.

Una **nota tematica** può allora esprimere ciò che si comprende leggendole insieme:

2

La qualità di un rapporto dipende da una tensione fra lo sforzo richiesto al momento della compilazione e la struttura necessaria al suo utilizzo.

Questa terza nota non è una media diplomatica. Non sostituisce nessuna delle due note atomiche. Mostra il problema nuovo che compare quando le si legge insieme. La contraddizione raggiunge anche lo stato del lavoro: come ottenere un'informazione strutturata senza provocare il rinvio o l'aggiramento della compilazione? Nessuna delle tre fonti permette ancora di rispondere.

Ecco che cosa si vede sullo schermo

Il Contesto usato per questo esempio esiste sotto forma di una piccola cartella di file. La sua struttura non è il concetto in sé: è il mio modo attuale di dargli una forma concreta.

```
contexte-qualite-comptes-rendus/  
├─ 00-frame/  
│   ├── mission.md  
│   └── directives.md  
├─ 01-state/  
│   ├── state.md  
│   └── contradictions.md  
├─ 03-sources/  
│   ├── SRC00001-entretien-exploitation.md  
│   ├── SRC00002-entretien-exploitation.md  
│   └── SRC00003-livre-blanc-donnees-intervention.md  
├─ 04-notes/  
│   ├── atomic/  
│   │   ├── AT000001-la-contrainte-de-saisie.md  
│   │   └── AT000002-des-comptes-rendus-structures.md  
│   └── thematic/  
│       └── THM00001-la-qualite-depend-de-la-saisie.md  
└─ 05-outputs/  
    └── questions-prochain-entretien.md
```

Adesso puoi collegare le parole usate dall'inizio del libro a oggetti visibili.

- La **missione** spiega perché la cartella esista.
- Le **fonti** conservano ciò che è arrivato prima dell'interpretazione.
- Le **note atomiche** fissano idee distinte.
- La **nota tematica** rende intelligibile la loro tensione.
- Lo **stato** dice che cosa quella tensione lasci aperto.
- Il **deliverable** prepara l'azione successiva.

Niente di tutto ciò richiede un software misterioso. VS Code (ma potrebbe essere l'applicazione Codex o Claude Cowork) è qui la finestra attraverso cui accedo al Contesto. Il Contesto sta

invece nei file, nella funzione di ciascuno e nelle relazioni conservate fra di essi.

Il riscontro si fonda sul lavoro svolto

Qualche giorno dopo riprendo il lavoro e propongo all'IA:

«Forse dovremmo rendere tutti i campi facoltativi.»

2

In una conversazione isolata, l'IA potrebbe difendere quell'idea, proporre un'interfaccia più leggera o elencarne i vantaggi. Nel Contesto incontra le due note atomiche e la tensione che le lega.

Può allora rispondere:

Questa proposta ridurrebbe probabilmente una parte dell'attrito descritto nelle interviste. Si scontra però con il bisogno di dati strutturati rilevato nel white paper. Le fonti mostrano due vincoli reali; non permettono ancora di scegliere fra semplificazione, compilazione a tappe, dettatura vocale o adattamento dei campi al tipo di intervento.

Quel riscontro non viene da una conoscenza generale del product management. Si fonda sul lavoro in corso. L'IA sa quale domanda stiamo cercando di capire, quali idee siano sostenute da quali fonti e quale contraddizione resti aperta. Non decide al posto mio. Impedisce semplicemente alla mia nuova intuizione di cancellare una delle realtà già stabilite.

Il primo deliverable prepara la fonte successiva

Lo stato del Contesto non dice: «progettare un nuovo modulo».

Separa ciò che sembra assodato da ciò che resta incerto.

Ciò che sembra assodato *Dei vincoli di compilazione possono provocare rinvio e aggiramento. Una struttura sufficiente resta necessaria per sfruttare i rapporti.*

Ciò che resta aperto *Non sappiamo ancora quale attrito pesi di più e quali informazioni debbano assolutamente restare strutturate.*

Prossima azione *Preparare un'intervista che distingua le cause del problema invece di presentare subito una soluzione.*

Il primo deliverable è dunque modesto: una guida di domande per le nuove interviste. Deve aiutare a capire quando viene compilato il rapporto, quali vincoli incontrano i tecnici sul campo, l'utilità reale dei vari campi e come vengono aggirati i requisiti.

Il deliverable non risolve la contraddizione. Prepara le informazioni che forse permetteranno di risolverla più avanti.

Un oggetto di lavoro, non una macchina che fonde tutto

Questo esempio mostra anche un errore che l'IA può commettere: voler far entrare una fonte nuova in una nota esistente perché gli argomenti si somigliano.

Una **nota atomica** può crescere quando più fonti riguardano la stessa proposizione. È quello che succede con le due

interviste. Ma il white paper non è una nuova prova di quella proposizione. Sostiene un'idea diversa, che merita una nota propria.

La distinzione conta. Se ogni contraddizione riscrive la nota precedente, *l'insieme delle idee* finisce per smussare le tensioni. La conoscenza sembra coerente, ma non corrisponde più a quello che le fonti hanno davvero portato.

Il Contesto non garantisce che l'IA scelga sempre il confine giusto. L'essere umano deve poter rifiutare una fusione, chiedere una nota nuova e verificare la relazione creata fra di esse. La struttura rende quella correzione possibile e duratura; non la decide in automatico.

Adesso hai visto l'oggetto:

- **fonti** che restano intatte,
- **note atomiche** che conservano ciascuna il proprio centro,
- una **nota tematica** che accosta senza fondere,
- una **contraddizione** che resta aperta e un **deliverable** che prepara il seguito.

I capitoli seguenti potranno tornare agli usi del Contesto, e poi aprirne più da vicino l'architettura a partire da questi elementi diventati visibili.

DA RICORDARE — UNA CONTRADDIZIONE SI CONSERVA

Due fonti possono essere vere insieme e tirare la decisione in direzioni opposte. Una nota atomica cresce quando una fonte rafforza la stessa proposizione; una proposizione diversa merita la propria nota. Fondere per far sparire la tensione produce una conoscenza coerente che non corrisponde più alle fonti.

Parte 3

Tre usi concreti

Le due parti precedenti hanno detto che cosa cambia il **Contesto**. Questa parte lo mostra al lavoro, su tre usi che pratico davvero.

Rispondere a un questionario del cliente, scrivere un articolo a partire da un pensiero ancora informe, far maturare un tema di prodotto in mezzo a una giornata frammentata. Tre durate, tre tipi di materiale, tre destinatari, e ogni volta lo stesso movimento: ritrovare quello che è già stato capito, trasformarlo e poi produrre senza esaurire quello che è servito.

Rispondere a un questionario del cliente senza perdere il lavoro fatto

Il primo questionario del cliente che ho trattato con questo sistema aveva quarantatré domande.

Bisognava rispondere sull'architettura del prodotto, sulla sua sicurezza, sui backup, sulla continuità del servizio e sulla protezione dei dati. Una parte delle informazioni stava nella nostra documentazione. Altre erano sepolte in file di configurazione. Per alcune bisognava rivolgersi al team che gestiva il servizio.

Abbiamo dovuto mettere insieme sette fonti tecniche e fare un andirivieni con il team. Quel lavoro si aggiungeva alla stesura vera e propria: capire che cosa chiedesse il potenziale cliente, ritrovare l'informazione, verificare che fosse ancora vera, decidere che cosa potevamo comunicare e riportare poi la risposta nel formato atteso.

Non era una situazione eccezionale. Ogni nuovo potenziale cliente mandava il proprio documento. Una volta era un foglio di calcolo, un'altra un file Word o un PDF. I riferimenti cambiavano, le frasi anche, ma tornavano gli stessi argomenti. Eppure ogni volta avevamo l'impressione di ricominciare. Rispondere richiedeva dalle tre alle sei ore e impegnava più persone, senza nessuna garanzia che il potenziale cliente

diventasse cliente. Poi il documento partiva e il lavoro restava chiuso nel file spedito.

Il mio primo obiettivo era dunque semplice: quando fosse arrivato il questionario successivo, non volevo ripartire da zero.

Una risposta credibile, ma falsa

La soluzione più ovvia era conservare i vecchi questionari e riutilizzarne le risposte.

Abbiamo scoperto presto perché non bastasse.

Alcuni documenti ricevuti contenevano già delle risposte.

Potevano venire da un modello, da un altro fornitore o da uno scambio precedente. In uno di essi una risposta diceva che la certificazione ISO 9001 era del 2017. La frase sembrava perfettamente utilizzabile. Era precisa, già scritta e messa di fronte alla domanda giusta.

La data corretta era il 2018.

L'errore riguardava un solo anno, ma faceva emergere un problema molto più grande. Una vecchia risposta assomiglia a una conoscenza. Ha una forma affermativa e rassicurante.

Quando un'IA la incontra in un documento, può riprenderla senza fatica. Eppure la sua presenza non dice chi l'abbia scritta, né su che cosa poggi, né se descriva davvero il nostro prodotto.

Ho capito allora che non dovevo costruire una biblioteca di risposte da copiare. Bisognava costruire una memoria di ciò che sapevamo davvero rispondere.

Il questionario ricevuto conservava un valore: mostrava che cosa il cliente volesse sapere. Ma da solo non poteva provare la risposta. Quella distinzione è diventata una regola del *Contesto*: il documento del cliente porta la domanda; la risposta deve restare legata a un'informazione che abbiamo verificato.

Conservare i fatti invece dei moduli

Abbiamo cominciato a estrarre dai questionari ciò che poteva servire al successivo.

3

Le domande non erano più ordinate soltanto per cliente o per numero. Le raggruppavamo per argomento: backup, autenticazione, cifratura, subfornitori, hosting, continuità del servizio. Una domanda chiamata «14.6.l» da un potenziale cliente poteva così raggiungere una domanda chiamata «CL-072» da un altro, se entrambe riguardavano lo stesso argomento.

Per ognuno di quegli argomenti il *Contesto* conservava i fatti convalidati e la loro origine. La risposta poteva appoggiarsi a una documentazione ufficiale, a una configurazione verificata, a un test o alla conferma del team competente. Conservavamo anche la data della convalida e i limiti di ciò che potevamo dichiarare.

Quel lavoro non produceva sempre una risposta completa. Faceva emergere anche i buchi.

Quando mancava un'informazione, il sistema doveva dirlo. Preparava una domanda precisa per il team interessato e indicava quali righe del questionario quella risposta avrebbe

permesso di completare. Un'informazione assente smetteva così di nascondersi dietro una frase credibile. Diventava un'azione identificabile.

Separavamo infine ciò che serviva a verificare una risposta da ciò che poteva essere mandato al cliente. Il dettaglio di una configurazione poteva essere indispensabile a chi rileggeva all'interno e non avere posto nel documento pubblico. Il Contesto preparava dunque la formulazione destinata al potenziale cliente e conservava accanto le prove e le riserve necessarie alla sua approvazione.

A poco a poco l'obiettivo è cambiato. Non si trattava più solo di finire il questionario presente. **Ogni risposta verificata doveva ridurre il lavoro necessario per quelli successivi.**

Il quarto questionario

Il quarto questionario aveva centoquindici domande, molte più del primo.

Questa volta non partivamo più da un insieme di file sparsi. Il Contesto conosceva già gli argomenti ricorrenti e sapeva dove ritrovare i fatti corrispondenti. Novantasei risposte complete sono nate direttamente dalla memoria accumulata e dalle fonti del prodotto, senza nuova raccolta.

Il sistema ha anche classificato il resto del documento. Alcune risposte erano solide, ma chiedevano un'ultima verifica. Altre restavano incerte. Qualcuna non poteva essere data con le informazioni disponibili.

Soprattutto, i buchi avevano cambiato natura. Non passavamo più ore a cercare a caso chi sapesse che cosa. I punti ancora aperti corrispondevano a decisioni commerciali — per esempio la durata di un impegno — o a conferme tecniche puntuali. Il lavoro rimasto era visibile, assegnabile e limitato. Quel risultato mi ha permesso di vedere che cosa significhi davvero rispondere «automaticamente» a un questionario. Non si tratta di chiedere a un'IA di riempire tutte le caselle con sicurezza. Si tratta di preparare subito ciò che è preparabile, di mostrare quanta fiducia meriti ogni risposta e di isolare con chiarezza ciò che richiede ancora un giudizio umano.

Dieci minuti non vogliono dire cento per cento di certezza

Nella mia pratica, la preparazione di un questionario è passata da tre-sei ore a circa dieci minuti per produrre l'80-90 % del documento.

Quel numero potrebbe dare un'impressione ingannevole. Non significa che l'80-90 % delle risposte diventi all'istante certo e pronto a impegnare l'azienda. Significa che il documento è in buona parte preparato: le risposte note sono al loro posto, le fonti ritrovate, i punti fragili segnalati e le informazioni mancanti trasformate in richieste precise.

La rilettura umana resta indispensabile. Colpisce però nei punti giusti. Invece di passare il tempo a frugare vecchi file e a disturbare un'altra volta le stesse persone, chi rilegge può

verificare gli impegni, decidere che cosa verrà divulgato e concentrarsi sulle risposte nuove o incerte.

Il guadagno più importante non è dunque soltanto il passaggio da diverse ore a qualche minuto. È il cambio di natura del lavoro.

Il primo questionario aveva richiesto un'indagine. Quelli successivi hanno beneficiato di tutto quello che l'indagine aveva permesso di imparare. Un errore di data è diventato una regola di prudenza. Una conferma ottenuta da un team è diventata un fatto riutilizzabile. Una domanda senza risposta è diventata una mancanza nota invece che una dimenticanza.

Il potenziale cliente riceve sempre il suo documento.

L'azienda, invece, non perde più quello che ha imparato preparandolo.

DA RICORDARE — UNA RISPOSTA VALE PER LA SUA PROVA

Una vecchia risposta assomiglia a una conoscenza: è affermativa, ben formata, e tace sulla propria origine e sulla propria data.

Conservare i fatti verificati e la loro provenienza, non i moduli già compilati. Una mancanza dichiarata vale più di una frase credibile: diventa una domanda rivolta a qualcuno.

Scrivere un articolo a partire da un pensiero ancora informe

Gli articoli che pubblico sul mio blog cominciano di rado davanti al computer, di fronte a una pagina bianca preparata con cura.

3

Un'idea arriva dopo una difficoltà incontrata, uno strumento provato, una discussione o un cambio di parere. A volte capita camminando, sui mezzi pubblici o sul divano. Sento che merita un testo, ma non conosco ancora né la sua angolatura esatta né l'ordine in cui andrà raccontata.

In quel momento prendo spesso il telefono.

Non lo uso soltanto per registrare una nota vocale o dettare un testo che ricopierò più tardi. **Apro una conversazione con l'IA.** Mi ascolta, mi risponde e mi incalza mentre rifletto.

Quella conversazione finisce nello stesso ambiente di lavoro che ritroverò sul computer. Posso dunque cominciare lontano dalla scrivania senza creare una nota isolata che poi andrebbe trasferita e reinterpretata.

Scaricare un pensiero che si sta formando

Creo un `Contesto` con un obiettivo semplice: scrivere l'articolo. Poi gli do tutto quello che ho in testa.

A volte per due ore parlo di un primo punto, poi di un altro, e torno al primo. Racconto un'esperienza, salto a una

conclusione, aggiungo un dettaglio, apro una pista nuova e lascio per un po' quella precedente. Posso ripetere quattro volte la stessa idea perché mi torna in forme diverse. Posso anche affermare una cosa e dire il contrario un'ora dopo, perché il mio pensiero si è spostato durante la conversazione. Quel disordine non è un incidente da togliere di mezzo prima di cominciare. È spesso la forma reale di un'idea finché sta ancora nascendo.

Parlare con l'IA mi permette di andare avanti senza imporre troppo presto una struttura. Una risposta mi fa mettere a fuoco un ricordo. Una domanda rivela che ho appena saltato un passaggio. Una riformulazione mi mostra che quello che credevo ovvio non lo è ancora. Correggo, sfumo, e riparto in un'altra direzione.

Il Contesto conserva quelle parole come fonte di sessione. Non trattiene soltanto le frasi più pulite. Conserva anche i ritorni, le esitazioni e le contraddizioni che permetteranno di vedere come il pensiero si è spostato.

L'IA accosta poi le quattro formulazioni di una stessa idea invece di trattarle come quattro idee diverse. Segnala due affermazioni incompatibili, ritrova la sfumatura arrivata un'ora più tardi e mette un esempio accanto al principio che illumina. **Non fa sparire il disordine: lo rende lavorabile.**

Questa prima tappa mi libera da una scelta prematura. Non devo ancora decidere quale frase aprirà l'articolo, se una certa idea meriti una sezione o in che ordine raccontare i

fatti. Quello che cerco per prima cosa è svuotare ciò che ho in testa, senza lasciar scappare le idee che non fanno che attraversarla.

Il Contesto sa però perché quel materiale è lì. Non riceve una conversazione senza destinazione. Riceve il materiale ancora disordinato di un articolo da costruire.

Usare l'IA come compagna di maturazione

Una volta depositato quel primo materiale, chiedo all'IA che cosa manchi.

Individua un'affermazione troppo frettolosa, un presupposto mai spiegato, un passaggio che sta in piedi solo se il lettore conosce già la mia esperienza. Chiede un esempio. Accosta due idee formulate in momenti diversi. A volte propone una pista che sembra credibile e che rifiuto perché non corrisponde a quello che ho vissuto.

Rispondo, preciso, correggo. Quella conversazione non produce ancora l'articolo. Ne addensa il materiale.

Il ruolo dell'IA è importante, ma il suo posto resta chiaro. Dispone di tutto quello che ho versato e vi cerca tensioni più in fretta di me. Ignora invece che cosa io voglia davvero sostenere finché non l'ho formulato. Lei apre la domanda; io decido se quella domanda meriti di spostare il testo.

Questo dialogo accorcia il tempo necessario a imbattersi in un punto debole. Da solo avrei scoperto due giorni dopo che mancava un esempio o che la mia conclusione poggiava su un

passaggio mai spiegato. L'IA provoca quella scoperta mentre l'idea è ancora facile da rifare.

Aggiungere fonti quando l'esperienza non basta più

I miei articoli partono spesso da quello che ho vissuto. Il materiale esterno conserva però il suo ruolo:

- un riferimento dà un nome a quello che ho osservato;
- una documentazione verifica un funzionamento;
- un altro autore sostiene una posizione vicina, o mostra un limite che non avevo incontrato;
- una ricerca ulteriore rivela che la mia esperienza copre soltanto un caso particolare.

Quei documenti entrano nel `Contesto` come **fonti documentali**.

Il loro ruolo è diverso da quello delle mie prime parole. La **fonte di sessione** conserva quello che affermo e quello che capisco della mia esperienza. Le **fonti documentali** portano un materiale che posso confrontare con quell'esperienza. L'articolo può allora distinguere quello che ho osservato, quello che una fonte stabilisce e quello che resta un'interpretazione. Quella distinzione protegge il testo da una tentazione frequente: presentare un'esperienza personale come una regola generale.

Trasformare le fonti in idee riutilizzabili

A questo punto il `Contesto` non passa direttamente dalle fonti alla scaletta dell'articolo. Trasforma prima il materiale grezzo in **note atomiche**.

Ogni nota fissa un'idea autonoma nata dalla conversazione o da un documento:

- un'affermazione che voglio sostenere;
- un esempio che le dà una forma concreta;
- una sfumatura che ne limita la portata;
- un'obiezione a cui l'articolo dovrà rispondere;
- un fatto esterno che conferma o smentisce la mia esperienza.

3

La nota conserva la propria origine. Posso così capire l'idea senza rileggere tutta la conversazione, e tornare poi alla fonte se devo verificare una formulazione o ritrovarne il contesto.

Questa trasformazione cambia il modo di preparare l'articolo.

- La conversazione seguiva il movimento del mio pensiero; le note raggruppano ormai ciò che parla della stessa idea, anche quando quegli elementi sono comparsi a un'ora di distanza o in due fonti diverse.
- Una nota centrale può far emergere l'angolatura dell'articolo.
- Altre le portano esempi, prove, obiezioni o conseguenze.

Quelle note non sono ancora né paragrafi né una scaletta prefabbricata. **Restano neutre rispetto al deliverable: la stessa idea potrà alimentare questo articolo, una presentazione o un altro lavoro.**

Passare dalla scaletta prima di scrivere

Quando il materiale diventa sufficiente, chiedo una scaletta.

Questa tappa mi impedisce di passare direttamente da una conversazione ricca a un testo che ne seguirebbe semplicemente l'ordine. Una discussione avanza per ritorni, correzioni e associazioni. Un articolo deve condurre un lettore che non conosce né la discussione né la sua storia.

La scaletta fissa quella progressione. Dice che cosa vada capito per primo, dove mettere l'esempio principale, quale obiezione trattare e che cosa la conclusione possa davvero affermare.

L'IA propone una struttura a partire dal materiale disponibile. La sposto, la stringo, rifiuto le parti che danno troppo peso a una pista secondaria. Due o tre scambi bastano spesso per ottenere una direzione chiara.

La stesura viene dopo.

A quel punto l'IA non crea il testo a partire da un'istruzione vaga. Lavora con un corpus di idee e di temi estratti dalla mia esperienza raccontata, con le risposte alle domande, con le fonti e con una scaletta approvata. Raggruppa le ripetizioni, rimette a posto gli esempi e trasforma un parlato tortuoso in prosa leggibile.

Rileggo il risultato cercando tre scarti precisi:

- una sfumatura sparita;
- un passaggio diventato troppo perentorio;
- una formulazione scorrevole che tradisce l'idea.

Li correggo, e poi decido la versione pubblicata.

Che cosa porta il Contesto a un lavoro breve

Un articolo si può scrivere in uno o due giorni. Non sembra aver bisogno della stessa memoria di un prodotto seguito per anni o di un libro di decine di capitoli.

È proprio questo caso leggero a mostrare che la durata non è l'unica cosa in gioco.

Il Contesto tiene insieme le prime parole, le fonti aggiunte, le domande poste, la scaletta scelta e il testo completo. Se il lavoro si interrompe, ne ritrovo lo stato. Se una frase diventa discutibile, posso tornare alla sua origine. Se un'idea comparsa durante la scrittura va oltre l'articolo, può essere fissata a parte e alimentare un lavoro futuro.

Non tutto quel materiale merita per forza di essere conservato a tempo indeterminato. Un piccolo Contesto può raggiungere il suo obiettivo, produrre il suo articolo e poi essere archiviato. Il suo valore sta nella continuità che ha dato al lavoro mentre si formava e nella traccia che lascia se più avanti bisogna capire come il testo abbia preso quella direzione.

La pubblicazione stessa può far partire altri compiti: preparare un riassunto, adattare il testo a più lingue o produrre quello che serve alla messa online. Quelle operazioni prolungano lo stesso risultato, sotto regole che il Contesto conosce già.

Il movimento completo resta semplice: un'esperienza diventa parola, quella parola incontra domande e fonti, il materiale diventa una scaletta, e poi la scaletta diventa un testo.

L'IA accelera ogni trasformazione, e il *Contesto* tiene il filo che porta dall'una all'altra.

Niente di questo movimento appartiene in proprio alla scrittura lunga. Comincia appena un'idea merita più di una nota vocale dimenticata in un telefono.

DA RICORDARE — IL DISORDINE È MATERIALE

Un pensiero che nasce si ripete, si contraddice e torna indietro: è la sua forma normale, non un difetto da correggere prima di cominciare. Il *Contesto* conserva quelle parole come fonte di sessione, e poi le rende lavorabili. La scaletta si inserisce fra la conversazione e il testo, perché un lettore non segue l'ordine di una discussione.

Far maturare un tema di prodotto in mezzo al caos

I due capitoli precedenti hanno seguito un solo argomento per volta, dall'inizio alla fine. Questo capitolo descrive com'è davvero una giornata tipo di un product manager: più argomenti, richieste e tempi avanzano insieme.

Sono l'unico product manager della mia azienda. Vuol dire che gran parte delle informazioni legate al prodotto finisce per passare davanti a me.

Nella stessa giornata un cliente dice una frase che mi sembra importante.

- Sta in poche parole.
- Non so ancora se riveli un problema condiviso da altri utenti o soltanto una difficoltà particolare, ma non voglio perderla.

Poco dopo recupero un white paper sull'evoluzione di una questione di mestiere.

- Questa volta il materiale non sta più in una frase, ma in un PDF di diverse decine di pagine.
- Non so ancora che cosa mi sarà utile né se quel documento confermerà quello che osservo sul campo.

Poi gli sviluppatori mi parlano di una funzionalità in costruzione.

- Bisogna rispondere a una domanda, chiarire un comportamento o tornare su un'ipotesi presa qualche settimana prima.

Nel frattempo il CPO vuole fare il punto su due scommesse legate a un'altra parte del prodotto.

Una conversazione con il supporto rivela infine che manca una documentazione funzionale.

- Una parte di quella documentazione esiste ancora solo nella mia testa.

A fine giornata sono passato su tre o quattro argomenti diversi (o più). Ho ricevuto una citazione di poche parole, un documento intero, una domanda su uno sviluppo in corso, una richiesta di decisione e un bisogno di documentazione.

Non è una giornata particolarmente brutta: è una giornata ordinaria.

Ingressi che non hanno né la stessa dimensione né la stessa destinazione

Il caos del product management non viene soltanto dalla quantità di informazioni. Viene dalla loro differenza di natura. Ognuno di quei cinque ingressi guarda a un momento diverso della vita del prodotto:

- la frase di un cliente è un segnale debole, di cui non so ancora se ne annunci altri;
- un PDF dà una visione generale di un mestiere;

- una discussione con gli sviluppatori riguarda ciò che stiamo costruendo adesso;
- una richiesta del supporto riguarda ciò che esiste già e va spiegato;
- una domanda del CPO obbliga a confrontare più futuri possibili.

Quegli elementi non sono fatti per entrare in un unico documento.

3

- Non parlano tutti dello stesso argomento e non si collocano nello stesso momento della vita del prodotto.

Se li raccolgo semplicemente in un elenco di note, conservo le informazioni, ma rimando il lavoro di comprensione al giorno in cui ne avrò bisogno.

I Contesti mi permettono di tenere più fili in parallelo.

- La citazione raggiunge l'argomento a cui sembra appartenere, anche se il suo significato resta incerto.
- Il PDF viene conservato con la sua origine e può essere analizzato per estrarne le poche idee (**note atomiche**) davvero utili.
- La conversazione con gli sviluppatori raggiunge il lavoro sulla funzionalità in corso.
- La mancanza segnalata dal supporto diventa un lavoro da svolgere invece di restare un'osservazione sentita fra due riunioni.

Non tutto viene subito trasformato in conoscenza:

- la frase del cliente resterà forse ambigua,

- il white paper poco pertinente,
- l'intuizione smentita la settimana dopo.

Il Contesto non dà intelligenza a ciò che entra. Gli dà un posto, un'origine e una possibilità di essere ritrovato al momento giusto.

Il Contesto contiene il caos senza pretendere di abolirlo.

Mentre le informazioni entrano, le richieste in uscita continuano

La giornata non si riduce a ricevere e archiviare materiale: bisogna anche produrre.

Il supporto mi chiede per esempio una documentazione funzionale su una funzionalità a cui stiamo ancora lavorando.

- Bisogna spiegarne il funzionamento, chiarire i casi importanti e dare a chi segue i clienti quanto serve per rispondere come si deve.

Lo stesso giorno posso voler finire un articolo per il blog dell'azienda.

- Quell'articolo non parla per forza della stessa funzionalità.
- Può però riguardare lo stesso tema di mestiere: il lavoro a cui risponde il prodotto, una difficoltà ricorrente degli utenti o un cambiamento nelle loro pratiche.

I due documenti ricorrono allora a una parte della stessa conoscenza senza raccontare la stessa cosa.

- La documentazione destinata al supporto spiega il prodotto per come funziona.

- ▶ Deve essere precisa, operativa e adatta alle domande degli utenti.
- L'articolo sviluppa un'esperienza, un problema di mestiere o una riflessione.
 - ▶ Può usare alcune delle stesse idee, ma le organizza per un altro lettore e con un altro scopo.

Non ho bisogno di ricostruire due volte tutto quello che so dell'argomento.

3

- Il **Contesto** conserva un materiale intermedio (le **note atomiche** e le **note tematiche**) che non è ancora né la documentazione né l'articolo.
- A partire da quel materiale, l'IA può preparare più montaggi: uno centrato sull'uso del prodotto, l'altro sulla questione di mestiere più ampia.

Rileggo ovviamente entrambi i risultati.

- Una documentazione per il supporto impegna il modo in cui il prodotto viene capito.
- Un articolo impegna il mio punto di vista.

Ma dedico il mio tempo a verificare la sostanza e l'orientamento, invece di ritrovare gli stessi elementi e riscriverli da capo.

Questo movimento può avvenire nella stessa giornata. Mentre arrivano informazioni nuove, una conoscenza già ordinata permette di produrre i documenti di cui l'azienda o io stesso abbiamo bisogno.

Una piccola frase può diventare un tema di prodotto

La citazione sentita al mattino non fa scattare per forza una decisione.

- Presa da sola, può non provare nulla.
- Un altro cliente esprimerà forse una difficoltà simile qualche settimana dopo.
- Un'intervista porterà una sfumatura.
- Il white paper recuperato quel giorno darà forse un nome all'evoluzione osservata.
- Un dato del supporto mostrerà che il problema torna più spesso di quanto pensassimo.

Quegli elementi arrivano in momenti diversi e in forme diverse.

- Il **Contesto** permette di accostarli senza cancellarne l'origine.
- Può rilevare
 - che un'osservazione nuova assomiglia a un segnale già conservato,
 - far emergere una contraddizione,
 - chiedere che cosa permetterebbe di decidere fra due interpretazioni.

A poco a poco, una frase isolata può diventare un tema che vale la pena lavorare. A quel punto ho già una parte della sua storia: i primi segnali, i documenti che la illuminano, ciò che li conferma, ciò che li smentisce e le domande ancora aperte. Il **Contesto** aiuta quel materiale a maturare.

- Non decide che il problema sia importante e non sceglie la soluzione al posto mio.

La scommessa di prodotto resta mia responsabilità.

- In compenso posso prendere quella decisione con qualcosa di più di un ricordo vago o di una serie di documenti sparsi.

Lo stesso sistema accompagna poi il lavoro:

- aggiungere interviste,
- confrontare un'ipotesi con i fatti,
- preparare scenari,
- produrre un'argomentazione per il CPO,
- scrivere specifiche,
- ...

Un piccolo segnale colto in mezzo a una giornata può così continuare a vivere abbastanza a lungo da diventare una decisione vera.

Un'idea abbandonata può continuare a maturare

Non tutte le esplorazioni arrivano alla messa in produzione.

Ho lavorato su usi dell'intelligenza artificiale che sembravano interessanti ma che la tecnologia disponibile non permetteva ancora di realizzare come si deve.

- L'idea non era per forza sbagliata.
- Lo era il momento.

In un funzionamento classico, l'esplorazione avrebbe prodotto qualche documento e poi sarebbe sparita in una cartella dopo la decisione di abbandonarla.

- Uno o due anni dopo sarebbe stato necessario ritrovare quei file, ricordare le ipotesi di partenza e capire perché ci eravamo fermati.

Con un *Contesto*, l'esplorazione può essere archiviata insieme al suo percorso: i segnali iniziali, gli scenari immaginati, i tentativi, i limiti incontrati e la ragione dello stop.

- Quando la tecnologia cambia, posso riprendere quel lavoro senza trasformare un vecchio risultato negativo in un'idea definitivamente sbagliata.

Ritrovo quello che volevamo fare, ma anche perché non funzionava. Posso allora verificare se l'ostacolo esista ancora, conservare ciò che resta valido e riprendere la riflessione da lì.

La maturazione di un tema di prodotto non è dunque sempre un percorso continuo.

- Può cominciare da una frase, arricchirsi per mesi, interrompersi e poi riprendere quando il prodotto, il mercato o la tecnologia è cambiato.

Far emergere ordine senza rallentare la giornata

La mia giornata resta frammentata.

- I clienti non coordinano le loro osservazioni con le richieste del supporto.
- Gli sviluppatori non scelgono il momento in cui ho appena finito di leggere un PDF.
- Le decisioni da preparare non si fermano mentre scrivo una documentazione o un articolo.

Il Contesto non trasforma quella realtà in una successione perfetta di compiti.

Permette a ogni frammento di conservare una continuità.

- Una piccola frase e un lungo documento possono entrare senza ricevere lo stesso trattamento.
- Più argomenti possono avanzare senza confondersi.
- Una conoscenza già lavorata può servire a più deliverable senza restare chiusa nel primo.
- Un'idea interrotta può aspettare senza perdere tutta la sua storia.

È così che lo uso nel product management: non per rendere la mia giornata meno mossa, ma per evitare che il suo movimento disperda quello che imparo.

Il caos continua a entrare.

Non riparte più sotto forma di perdita.

Questi tre usi hanno un punto in comune che non ho ancora nominato: ciascuno finisce con qualcosa che parte — una risposta spedita, un articolo pubblicato, una decisione presa.

La parte seguente guarda che cosa resta una volta che il deliverable è partito.

DA RICORDARE — PIÙ FILI IN PARALLELO

Una frase di un cliente e un PDF di cinquanta pagine chiedono un trattamento e una destinazione diversi. Il *Contesto* dà a ogni frammento un posto, un'origine e una possibilità di essere ritrovato al momento giusto. Un'esplorazione abbandonata conserva le sue ragioni di stop: torna riprendibile quando la tecnologia o il mercato cambia.

Parte 4

Che cosa si lasciano dietro i deliverable

Ognuno dei tre usi precedenti finisce con un deliverable consegnato. Questa parte guarda che cosa resti una volta che è partito.

Fa da filo una stessa domanda, posta due volte a sei mesi di distanza. La prima volta la risposta sta in una cella di un foglio di calcolo. La seconda, quella risposta esiste ancora ma non basta più: manca il ragionamento che l'aveva prodotta, e le relazioni fra i documenti che lo sostenevano.

4.1

La risposta sta in una cella, il ragionamento no

Torniamo a uno dei questionari della parte precedente.

La domanda arriva nella cella D47 di un foglio di calcolo:

| *I vostri backup sono cifrati a riposo?*

La risposta finale starà in tre righe. Per scriverle, però, devo mettere insieme più elementi:

- la nostra documentazione descrive come funzionano i backup in generale;
- un file di configurazione indica l'impostazione davvero attiva;
- il team Ops conferma come quell'impostazione si applichi in produzione;
- l'ufficio legale chiarisce che cosa possiamo comunicare a un potenziale cliente.

Passo dal foglio di calcolo alla documentazione, dalla documentazione alla configurazione, e poi da uno scambio con il team Ops a una decisione legale. Torno infine nella cella D47 e ci scrivo la risposta.

Il documento parte.

Tre righe alla fine di una catena

La cella contiene il risultato del lavoro. Non mostra il cammino che l'ha prodotto.

Non dice:

- quale documentazione ho consultato;
- quale impostazione ho verificato;
- chi ne ha confermato l'applicazione;
- quale formulazione è stata autorizzata;
- quali riserve sono state tolte dal testo mandato al potenziale cliente.

Quegli elementi però esistono. Hanno determinato ognuna delle tre righe e, mentre rispondo, la loro relazione è chiara nella mia testa.

Una volta spedito il documento, quell'unità sparisce. Il foglio di calcolo conserva la risposta, ma non il ragionamento che permetteva di fidarsene.

Ogni strumento ha fatto il suo lavoro

Il problema non viene da un cattivo ordine.

- La documentazione sta proprio nello spazio documentale.
- La configurazione resta nel prodotto.
- La risposta del team resta nel suo scambio.
- La decisione legale ha il proprio supporto.
- La cella D47 è al suo posto nel documento del potenziale cliente.

Ogni strumento ha conservato l'oggetto che gli era stato affidato. Nessuno aveva la funzione di conservare l'insieme del ragionamento.

Per ricostruirlo bisogna sapere quali elementi cercare, dove cercarli e quale ruolo ciascuno abbia svolto. Quella conoscenza resta in capo a chi ha preparato la risposta.

Il deliverable appiattisce ciò che l'ha prodotto

Un deliverable deve presentare un risultato adatto al suo destinatario.

- Il potenziale cliente aspetta una risposta chiara, non il diario completo dell'indagine svolta per ottenerla.

Quella semplificazione è necessaria. Diventa costosa quando il deliverable resta l'unica traccia del lavoro.

Il risultato finale è sempre più corto del cammino che ci porta: la risposta spedita, l'articolo pubblicato o la decisione di prodotto. Se resta solo il risultato, il lavoro successivo deve rifare una parte di quel cammino.

Dare un indirizzo al ragionamento

Il Contesto non sostituisce né il foglio di calcolo, né la documentazione, né la configurazione, né gli scambi con i team.

- Dà un indirizzo comune al lavoro che li collega.

La domanda D47, gli elementi consultati, le convalide ottenute e la risposta prodotta appartengono allo stesso argomento.

Possono conservare la loro forma e la loro collocazione d'origine restando al tempo stesso collegati nel *Contesto*.

La cella continua a svolgere la sua funzione: rispondere al potenziale cliente. Il *Contesto* ne svolge un'altra: conservare ciò che permetterà di capire, di verificare e di riprendere quella risposta.

Sei mesi dopo, quella differenza diventa decisiva.

DA RICORDARE — IL DELIVERABLE NON CONSERVA IL RAGIONAMENTO

Una risposta può stare in tre righe e dipendere comunque da più fonti, convalide e riserve. Gli strumenti conservano ciascuno il proprio oggetto; a portare le relazioni fra di essi è ancora l'essere umano. Il *Contesto* dà un indirizzo al lavoro che collega quegli elementi senza sostituire i loro strumenti d'origine.

Sei mesi dopo, la risposta non basta più

Sei mesi dopo la cella D47, un nuovo potenziale cliente fa la stessa domanda sulla cifratura dei backup.

- Ritrovo la vecchia risposta in dieci secondi.
- È chiara, ben scritta e pronta da ricopiare.

Eppure resto lì davanti senza poterla usare.

Bastano tre domande a bloccarmi:

- da dove veniva quella risposta: dalla documentazione, dall'impostazione osservata o da quello che il team Ops mi aveva spiegato quel giorno?
- l'ufficio legale aveva approvato quella formulazione esatta o una versione più prudente?
- l'impostazione è ancora la stessa oggi?

La risposta è sopravvissuta; ciò che permetteva di fidarsene è sparito.

Ritrovare non è ancora riprendere

La vecchia cella mi restituisce un testo. Non mi restituisce:

- il suo livello di prova,
- i suoi limiti,
- né le verifiche necessarie prima di usarla di nuovo.

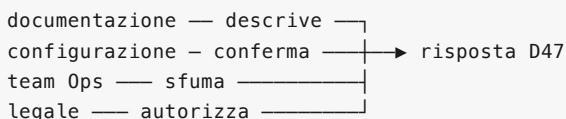
Senza un'altra memoria devo rifare tutta l'indagine del capitolo precedente, in ordine inverso. Non riparto proprio da

zero, visto che una risposta ce l'ho già. Ma riparto senza sapere quale parte del lavoro resti valida.

È l'errore della certificazione datata 2017 invece che 2018, visto questa volta dal lato delle nostre risposte: la frase è netta, viene da noi, e la sua prova è diventata invisibile.

Ritrovare il ruolo di ogni elemento

Nel Contesto la vecchia risposta non resta sola. Conserva le relazioni di cui aveva bisogno il ragionamento che l'ha prodotta:



4

I verbi cambiano ciò che posso fare di ogni elemento.

- La documentazione describe il funzionamento generale; da sola non prova l'impostazione attiva.
- La configurazione conferma quell'impostazione in un dato momento; non decide che cosa possa essere comunicato.
- Il team Ops aggiunge una sfumatura che viene dall'operatività quotidiana.
- Il legale autorizza una formulazione destinata al potenziale cliente.

Non ritrovo più soltanto quattro oggetti vicini a una risposta.

Ritrovo la funzione che ciascuno svolgeva.

Quella precisione impedisce a una fonte di provare più di quanto stabilisca davvero. Mostra anche quale verifica vada rifatta quando è passato del tempo.

Verificare ciò che può essere cambiato

Il Contesto non trasforma la vecchia risposta in una verità permanente: ne prepara la verifica.

- La documentazione può essere ancora valida mentre la configurazione è cambiata.
- La configurazione può essere rimasta identica mentre la formulazione autorizzata va rivista.
- Una sfumatura portata dal team Ops può essere sparita dopo un'evoluzione del prodotto.

Le relazioni rendono visibili quelle differenze. Posso controllare gli elementi che possono essere cambiati senza rifare a caso tutta l'indagine.

La risposta finale sarà forse identica a quella di sei mesi fa. La differenza sta nel fatto che so perché posso ancora usarla.

Conservare le relazioni che servono al lavoro

Non si tratta di collegare tutti i documenti fra loro.

- Due file possono condividere una parola senza partecipare allo stesso ragionamento.

Una relazione merita di essere conservata quando cambia il modo di capire, di verificare, di decidere o di produrre.

- Nel caso di D47 bastano quattro relazioni: descrivere, confermare, sfumare e autorizzare.

Permettono di rispondere a domande concrete:

- su che cosa poggia questa risposta?
- quale parte va verificata oggi?
- chi aveva approvato quello che potevamo comunicare?
- che cosa renderebbe caduca questa risposta?

Il *Contesto* non cerca dunque di fabbricare una mappa totale di tutto quello che so. Conserva il tessuto necessario a un lavoro preciso.

Aprire la macchina

Il Libro I ha mostrato come il *Contesto* sia nato, che cosa cambi nel rapporto con l'IA e come accompagni diversi usi. Il caso D47 fa emergere quello che resta da capire.

- Come conservare la missione del lavoro senza ripeterla a ogni seduta?
- Dove mettere le fonti, le idee, le decisioni e i deliverable perché conservino ruoli distinti?
- Come caricare ciò che serve adesso senza far entrare tutto il materiale nella conversazione?

Il Libro II aprirà quella macchina. Seguirà il percorso completo delle informazioni in un *Contesto*: dal loro arrivo alla trasformazione, a ciò che producono e alla ripresa del lavoro.

DA RICORDARE — RITROVARE CIÒ CHE PERMETTE DI FIDARSI

Ritrovare una risposta non basta: bisogna ritrovare ciò che permette ancora di usarla. Una relazione nominata indica il ruolo preciso di una fonte, di un'approvazione o di una riserva. Il Contesto prepara la verifica mostrando che cosa sostenga la risposta e che cosa possa essere cambiato.

LIBRO II

Costruire e far vivere il Contesto

- 5 Sotto il cofano di un Contesto
- 6 Il Contesto nel suo ecosistema

Parte 5

Sotto il cofano di un Contesto

Finora il Contesto è stato descritto attraverso quello che permette di fare. Questa parte lo apre.

Presenta anzitutto il suo albero completo, poi segue una seduta di lavoro dall'inizio alla fine: aprire e ritrovare il punto in cui si è, far entrare del materiale, trasformarlo in idee, produrre un deliverable, decidere, e infine salvare perché la seduta successiva riparta senza ricostruire nulla. L'esempio che attraversa questi sei capitoli è del tutto inventato: capire perché alcuni tecnici di intervento consegnano rapporti di intervento incompleti, prima di decidere se il prodotto debba cambiare.

La mappa completa

Nella parte 2 ho lasciato il Contesto dedicato alla qualità dei rapporti di intervento:

- tre fonti,
- due note atomiche,
- una nota tematica,
- una guida per il prossimo colloquio.

Allora guardavo che cosa quel Contesto permettesse di capire. Adesso apro la cartella vera e propria.

Ecco l'albero completo della versione che uso in questo libro:

```

contexte-qualite-comptes-rendus/
├─ README.md           navigazione generale, generato
├─ offline.md          la cassetta delle lettere fuori seduta
├─
├─ 00-frame/           ciò che dà una direzione duratura
│   ├── README.md      navigazione della zona, generato
│   ├── mission.md     lo scopo, la domanda, i limiti
│   ├── directives.md  le regole proprie della missione
│   └─ loading.md      ciò che verrà letto all'apertura
├─
├─ 01-state/           il presente del lavoro
│   ├── README.md      navigazione della zona, generato
│   ├── state.md        punti fermi, dubbi, debiti
│   ├── synthesis.md    ciò che è diventato duraturo
│   ├── todo.md         le azioni da compiere
│   ├── decisions.md    le scelte che strutturano il Contesto
│   ├── contradictions.md ciò che resta in tensione
│   ├── handoff.md      come riprendere il lavoro
│   ├── checkpoints.md  la serie compatta degli stati
│   └─ health.md        il risultato dell'ultimo controllo
├─
├─ 02-history/         la traccia fattuale e datata
│   ├── README.md      il ruolo di questa traccia, generato
│   └─ YYYY-MM-DD.md   gli eventi della giornata

```

└─ 03-sources/	ciò che è stato ricevuto o detto
└─ README.md	l'elenco delle fonti, generato
└─ ingestion-log.md	che cosa è stato letto, che cosa ha dato
└─ documents/	
└─ README.md	l'elenco dei documenti, generato
└─ raw/	i file originali
└─ fulltext/	il loro testo integrale utilizzabile
└─ SRC00001-....md	identità, provenienza, stato
└─ sessions/	
└─ README.md	l'elenco delle sedute, generato
└─ SESSION-....md	le parole dette durante il lavoro
└─ 04-notes/	la conoscenza tratta dalle fonti
└─ README.md	la mappa delle note, generato
└─ atomic/	un'idea per nota – AT000001-....md
└─ thematic/	un'angolatura che collega – THM00001-....md
└─ glossary/	un termine fissato – GLO00001-....md
└─ decision-records/	un arbitrato – DRE00001-....md
└─ 05-outputs/	ciò che si produce per un uso
└─ README.md	l'elenco dei risultati, generato
└─ _draft/	i deliverable in corso
└─ questions-entretien.md	una guida per proseguire
└─ note-de-cadrage.md	una sintesi per decidere
└─ presentation-equipe.md	gli stessi punti, riformulati

I numeri impongono un ordine di lettura, non una gerarchia di valore:

- il quadro e lo stato orientano il lavoro;
- la storia permette di controllarne lo svolgimento;
- le fonti conservano il materiale, le note lo trasformano in conoscenza;
- i deliverable gli danno una forma per un destinatario.

Sei zone, sei funzioni

00-frame/ risponde alla domanda «perché sono qui, e sotto quali regole?». Questa zona cambia di rado, e non a caso:

modificare la missione trasformerebbe il mio modo di giudicare tutte le fonti, tutti i compiti e tutti i risultati.

```

contexte-qualite-comptes-rendus/
├─ 00-frame/                ciò che dà una direzione duratura
│   ├── README.md          navigazione della zona, generato
│   ├── mission.md         lo scopo, la domanda, i limiti
│   ├── directives.md      le regole proprie della missione
│   └─ loading.md          ciò che verrà letto all'apertura

```

01-state/ risponde a «a che punto sono adesso?». Questa zona si muove con il lavoro e mi permette di distinguere un punto fermo da un'ipotesi, un'azione futura da una decisione già presa, una difficoltà passeggera da una contraddizione che merita di restare visibile.

5

```

contexte-qualite-comptes-rendus/
├─ 01-state/                il presente del lavoro
│   ├── README.md          navigazione della zona, generato
│   ├── state.md           punti fermi, dubbi, debiti
│   ├── synthesis.md       ciò che è diventato duraturo
│   ├── todo.md            le azioni da compiere
│   ├── decisions.md       le scelte che strutturano il Contesto
│   ├── contradictions.md  ciò che resta in tensione
│   ├── handoff.md         come riprendere il lavoro
│   ├── checkpoints.md     la serie compatta degli stati
│   └─ health.md           il risultato dell'ultimo controllo

```

02-history/ risponde a «che cosa è successo davvero?». Questa zona non abbellisce il racconto della seduta: ne conserva i fatti utili, una fonte aggiunta, una pista abbandonata, una decisione presa, una nota creata. La consulto di rado, ma diventa decisiva il giorno in cui lo stato presente mi sembra incoerente.

```

contexte-qualite-comptes-rendus/
├── 02-history/                la traccia fattuale e datata
│   ├── README.md             il ruolo di questa traccia, generato
│   └── YYYY-MM-DD.md         gli eventi della giornata

```

03-sources/ risponde a «quale materiale ho ricevuto?». Un documento esterno vi entra congelato, la mia parola vi entra seduta dopo seduta. In entrambi i casi la fonte resta separata da ciò che il Contesto ne capirà.

```

contexte-qualite-comptes-rendus/
├── 03-sources/                ciò che è stato ricevuto o detto
│   ├── README.md             l'elenco delle fonti, generato
│   ├── ingestion-log.md      che cosa è stato letto, che cosa ha dato
│   ├── documents/
│   │   ├── README.md         l'elenco dei documenti, generato
│   │   ├── raw/              i file originali
│   │   ├── fulltext/         il loro testo integrale utilizzabile
│   │   └── SRC00001-....md    identità, provenienza, stato
│   └── sessions/
│       ├── README.md         l'elenco delle sedute, generato
│       └── SESSION-....md     le parole dette durante il lavoro

```

5

La scheda SRC00001-....md non contiene né il documento né il suo riassunto. Risponde a tutto quello che ho bisogno di sapere sul suo conto:

- da dove viene e perché l'ho aggiunto;
- dove ritrovare l'originale e il suo testo integrale;
- se l'estrazione è riuscita;
- quanta fiducia accordargli;
- se la sua lettura ha già dato delle note.

Rende dunque visibile lo stato della fonte senza toccare il materiale stesso.

04-notes/ risponde a «che cosa ho imparato?».

- Una nota atomica fissa un'idea.
- Una nota tematica collega più idee.
- Il glossario mette a fuoco una parola.
- Un decision record conserva le ragioni di un arbitrato.

```

contexte-qualite-comptes-rendus/
├─ 04-notes/                la conoscenza tratta dalle fonti
|   └─ README.md           la mappa delle note, generato
|   └─ atomic/             un'idea per nota - AT000001-...md
|   └─ thematic/           un'angolatura che collega - THM00001-...md
|   └─ glossary/           un termine fissato - GL000001-...md
|   └─ decision-records/    un arbitrato - DRE00001-...md

```

05-outputs/ risponde a «che cosa farò di questa conoscenza?».

Le bozze e i risultati vivono lì, nella forma che il loro pubblico si aspetta:

5

- una guida al colloquio;
- un articolo;
- una specifica;
- una presentazione;
- una decisione, fra altre forme possibili.

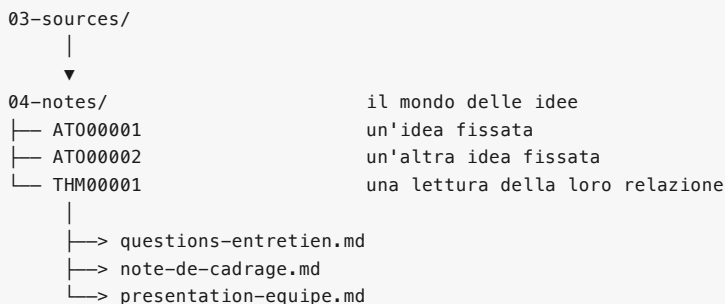
```

contexte-qualite-comptes-rendus/
├─ 05-outputs/              ciò che si produce per un uso
|   └─ README.md            l'elenco dei risultati, generato
|   └─ _draft/              i deliverable in corso
|   └─ questions-entretien.md una guida per proseguire
|   └─ note-de-cadrage.md    una sintesi per decidere
└─ presentation-equipe.md    gli stessi punti, riformulati

```

Il mondo delle idee: mattoni per più costruzioni

La separazione fra 04-notes/ e 05-outputs/ porta una delle forze principali del Contesto. Le note e il glossario vi formano un *mondo delle idee*: un insieme di mattoni di conoscenza, riutilizzabili all'infinito, che non sono più materiale grezzo e non sono ancora un risultato destinato a qualcuno.



Nel mio esempio, la fatica di compilare e il bisogno di dati strutturati sono stati fissati prima che sapessi in quale forma sarebbero stati presentati.

- la guida al colloquio ne ricava delle domande,
- la nota di inquadramento ne ricava la posta in gioco,
- la presentazione ne ricava un racconto destinato al team.

Questi tre deliverable non possiedono la conoscenza che usano.

- Ne scelgono e ricompongono una parte per un bisogno, un pubblico e un formato particolari.
- Un deliverable non esaurisce dunque le note di cui è fatto: i mattoni restano disponibili per un'altra costruzione.

Senza questo mondo intermedio, ogni nuovo risultato mi costringerebbe a riaprire le fonti, a rifare i collegamenti e a riformulare le stesse cose.

Il Contesto non conserva soltanto quello che è già stato prodotto: costruisce una capacità duratura di produrre di nuovo.

Non tutti i file cambiano allo stesso modo

L'albero diventa davvero leggibile solo quando so come cambia ognuno dei suoi elementi.

Certi file vengono riscritti per mostrare la situazione migliore disponibile adesso:

- `state.md`;
- `handoff.md`;
- le bozze di `05-outputs/_draft/`.

Altri non fanno che allungarsi, per conservare il passato:

- le fonti di sessione;
- la storia datata;
- la sintesi;
- la serie dei checkpoint.

I documenti ricevuti, invece, restano congelati, e una nota sostituita da un'altra resta consultabile. I file `README.md`, infine, vengono generati a partire da ciò che esiste: correggerli a mano creerebbe una seconda versione della realtà, che finirebbe per divergere.

Questa varietà spiega perché un solo grande file non mi bastava più. Avrei dovuto chiedere allo stesso testo:

- di mostrare il presente,
- di conservare il passato,
- di accogliere del grezzo,
- di fissare delle idee,
- di portare un risultato finito.

Uno scheletro invece di una ricetta da ricopiare

Questo scheletro non è una documentazione posata accanto al Contesto: è una cartella vera, che copio.

- Ogni file vi arriva con le sezioni che deve contenere e qualche indicazione, così che due Contesti creati a mesi di distanza condividano gli stessi punti di riferimento.

La struttura resta comunque adattabile. Un Contesto può ricevere:

- una regola particolare,
- un deliverable proprio del suo mestiere,
- una fonte di un tipo mai visto.

Lo zoccolo comune conserva soprattutto le tre frontiere di cui il mio lavoro ha bisogno:

- lo stabile e il presente,
- il grezzo e l'interpretato,
- la conoscenza e la sua messa in forma.

Parlare al Contesto

Finora ho mostrato cartelle e file. Se ne potrebbe dedurre che io debba aprire ciascuno di essi, sapere dove scrivere e spostare da solo ogni informazione. Non è così che lavoro.

Parlo all'IA:

Apri il Contesto dedicato alla qualità dei rapporti di intervento.

Aggiungi questo documento come fonte.

Trasforma questa fonte in note.

Salva il lavoro.

Dietro ognuna di queste richieste c'è una procedura che ho scritto una volta e che l'IA può riprendere a ogni seduta.

- In Claude Code e Codex, questo tipo di procedura riutilizzabile ha un nome: uno **skill**.

Lo skill di apertura indica per esempio:

- che cosa bisogna assorbire prima di cominciare;
- quali file leggere sistematicamente;
- che cosa deve restare su disco finché il compito non lo richiede;
- come segnalare un problema;
- quale informazione restituirmi perché io possa agire subito.

Non ho bisogno di ridare queste tappe ogni volta.

- Chiedo semplicemente di aprire il Contesto.

La conversazione porta la mia intenzione del momento.

- Lo skill porta il modo affidabile di eseguire i gesti che ritornano.

Il mio sistema ha così qualche verbo stabile:

- creare un `Contesto`;
- aprirlo e ritrovare il lavoro in corso;
- aggiungere una fonte o conservare una seduta;
- trasformare le fonti in note;
- salvare lo stato del lavoro;
- controllarne la coerenza;
- archiviarlo quando ha raggiunto il suo scopo.

Non tutti i miei scambi con l'IA passano da uno skill. Continuo a riflettere, a interrogare, a correggere e a decidere in lingua corrente.

Gli skill inquadrano soltanto le operazioni che richiedono di rispettare sempre le stesse tappe.

- Impediscono, per esempio, che una fonte venga trasformata senza conservarne l'origine o che una seduta finisca senza preparare la ripresa.

La procedura resta stabile, ma il giudizio non diventa automatico. Due modelli possono formulare una nota in modo diverso. Devono comunque consultare gli stessi elementi, rispettare le stesse frontiere ed eseguire gli stessi controlli. Condividere gli skill non dà dunque a qualcuno il mio `Contesto`. Gli dà lo stesso modo di creare il proprio e di tenerlo vivo. Non dirò altro sulla loro meccanica.

Gli skill non sono né un'invenzione né una particolarità del `Contesto`. Claude Code e Codex li propongono proprio per descrivere procedure riutilizzabili, e la loro documentazione spiega meglio di quanto potrei fare qui come crearli e usarli. Quello che appartiene al `Contesto` è la scelta delle competenze che gli si danno: creare, aprire, alimentare, trasformare, salvare, controllare e archiviare. È anche l'ordine in cui si susseguono e le frontiere che ciascuna deve rispettare. I file conservano quello che il `Contesto` sa. Gli skill organizzano i gesti attraverso cui esso cambia. La mappa è aperta. La visita può ora cominciare come comincia una seduta vera: aprendo il `Contesto`.

5

DA RICORDARE — LO SCHELETRO DI UN `CONTESTO`

Un `Contesto` nuovo arriva già arredato: sei zone, ognuna delle quali risponde a una domanda diversa. I numeri impongono un ordine di lettura, non una gerarchia di valore. Ciò che rende leggibile l'albero non è la sua forma, ma il fatto di sapere come cambia ogni file. Claude Code e Codex forniscono il meccanismo degli skill; il `Contesto` definisce i gesti ricorrenti, il loro ordine e i loro limiti.

Aprire: ritrovare la missione e il presente

Sono passate due settimane dall'ultima seduta sui rapporti di intervento — quei documenti in cui un tecnico annota che cosa ha riscontrato e che cosa ha fatto dopo un intervento su un'apparecchiatura.

- I colloqui e le note sono ancora lì, al loro posto.

Ma se li riapro a caso, passerò mezz'ora a ricostruire quello che già sapevo, e lo ricostruirò male.

- L'apertura comincia dunque da un nucleo molto più piccolo: il quadro, lo stato e la ripresa.

Non devo chiedere tutto questo. Chiamo lo skill di apertura, che svolge la sua procedura.

Ritrovo anzitutto perché sono qui

Il primo file che l'IA legge (tramite il suo skill di apertura del contesto) è `00-frame/mission.md`. Il suo modello pone cinque domande:

```
# Missione
```

```
## Lo scopo
```

```
Che cosa deve permettere questo `Contesto`?
```

```
## La domanda centrale
```

```
Quale domanda guida la selezione e l'esplorazione?
```

```
## Che cosa rientra nel tema
```

```
Quale materiale può entrare?
```

```
## Che cosa è fuori tema
```

```
Che cosa si rifiuta, anche se sembra interessante?
```

```
## Il deliverable atteso
```

```
Che cosa si vuole produrre, se già lo si sa?
```

Nel mio, l'inizio è compilato così:

```
## Lo scopo
```

```
Capire che cosa peggiora la qualità dei rapporti di  
intervento.
```

```
## La domanda centrale
```

```
Come raccogliere un'informazione utilizzabile senza rendere  
scoraggiante la compilazione?
```

```
## Che cosa è fuori tema
```

- Scegliere un'interfaccia prima di aver messo a confronto più fonti.
- Generalizzare a partire da un solo colloquio.

```
## Il deliverable atteso
```

```
Preparare le decisioni di prodotto e i prossimi colloqui.
```

Una missione che dicesse soltanto «migliorare i rapporti di intervento» non mi orienterebbe granché. La domanda centrale, invece, rende già certi documenti più pertinenti di altri, e il fuori tema impedisce a un'idea seducente di

trasformare in silenzio la mia esplorazione in un progetto di soluzione.

- Il deliverable, dal canto suo, può restare sconosciuto.
- Scrivo questa incertezza invece di inventare un esito prematuro: il Contesto sa allora che sta ancora esplorando.

Le regole che avevo stabilito mi aspettano

Due missioni vicine possono richiedere precauzioni diverse.

`directives.md` conserva queste regole insieme al lavoro:

- riservatezza,
- livello di prova,
- lessico,
- approvazione o forma attesa
- ...

Il modello è volutamente semplice: una regola, una data, una formulazione all'imperativo.

Direttive

- (2026-09-11) Anonimizzare ogni colloquio prima di inserirlo: sostituire il nome della persona intervistata con la sua funzione, ma conservare il nome dell'azienda e il paese.
- (2026-09-11) Distinguere quello che le fonti mostrano da quello che ne deduciamo.
- (2026-09-11) Non proporre nessuna soluzione prima di aver documentato le pratiche reali.

Una direttiva vale solo se può guidare un'azione futura.

- «Essere prudenti» mi lascia da rifare tutta l'interpretazione;

- «distinguere l'osservazione dalla deduzione» mi dà un controllo preciso nel momento in cui scrivo una nota o un deliverable.

Ma una direttiva non è soltanto un'istruzione che ritrovo aprendo un file. È una regola permanente che il `Contesto` deve applicare ogni volta che un'azione rientra nel suo campo. Aggiungo per esempio la trascrizione di un nuovo colloquio. Chiedo semplicemente al `Contesto` di inserirla (tramite lo skill apposito), senza pensare a ricordare che contiene il nome della persona intervistata. La direttiva sull'anonimizzazione interviene comunque durante l'inserimento: prima di registrare la trascrizione come fonte, lo skill sostituisce quel nome con la funzione dell'intervistato, conservando insieme il nome dell'azienda e il paese.

Non devo ricordarmi della regola nel momento esatto in cui diventa necessaria. Il `Contesto` se ne ricorda per me.

Le direttive non smettono dunque di agire una volta finita l'apertura. Restano applicabili per tutta la seduta e accompagnano i diversi gesti del `Contesto`:

- quando si aggiunge una fonte, la regola sull'anonimizzazione protegge l'identità dell'intervistato;
- quando si crea una nota, la regola sul livello di prova separa quello che la fonte mostra da quello che ne deduciamo;
- quando si prepara un deliverable, l'ultima direttiva impedisce di presentare troppo presto una soluzione come acquisita.

Lo skill che esegue un'azione deve cominciare individuando le direttive che la riguardano, poi verificare che il suo risultato le rispetti.

```
richiesta
↓
direttive applicabili
↓
azione dello skill
↓
controllo del risultato
```

Le direttive circondano così l'azione: la vincolano prima dell'esecuzione e servono a controllarne il risultato dopo. Non agiscono da sole e non fanno tutto. Sono gli skill a renderle efficaci al momento giusto.

La missione indica dove andare. Le direttive accompagnano ogni gesto del Contesto, perché io non debba ricordarmi, a ogni tappa, delle precauzioni già decise.

Non tutto merita di essere riaperto

La mia cartella contiene decine di fonti e di note. Caricarle tutte a ogni apertura annegherebbe le priorità nella massa.

loading.md sceglie dunque un nucleo stabile:

```
## Sempre

- `00-frame/mission.md`
- `00-frame/directives.md`
- `01-state/state.md`
- `01-state/todo.md`
- `01-state/handoff.md`

## Su richiesta

- le fonti necessarie a una verifica
- le note legate al tema in lavorazione
- la bozza attiva
```

Tutto resta accessibile, ma non tutto viene caricato nel contesto dell'IA. `loading.md` indica gli elementi indispensabili per capire la missione, interpretare una richiesta e agire senza perdere le regole già stabilite.

5

- Il resto rimane su disco e viene consultato solo in caso di bisogno.
- Una vecchia fonte può essere indispensabile per controllare un'affermazione senza occupare spazio nel contesto attivo per tutte le sedute.

Il contenuto di `loading.md` dipende dal Contesto.

- Un libro caricherà la sua scaletta a ogni apertura, un'esplorazione di prodotto la sua nota tematica centrale.
- Il principio resta lo stesso:
 - cominciare con abbastanza informazioni per agire,
 - e andare a cercare il resto quando il compito lo giustifica.

A che punto ero davvero

La missione resta stabile; `01-state/state.md`, invece, è fatto per cambiare. Sta in tre blocchi:

```
# Stato

## Quello che sappiamo
- La compilazione rinviata comporta una perdita di dettagli
  nel primo colloquio.
- Certi campi obbligatori vengono aggirati nel secondo.

## Quello che resta incerto
- Non sappiamo quali campi vengano davvero riutilizzati
  dopo la compilazione.

## I debiti
- Il white paper è stato aggiunto ma il suo grado di
  attendibilità è ancora da valutare.
```

Lo stato non racconta tutta la storia e non riassume tutte le note: espone la migliore comprensione che ho del lavoro stesso.

`state.md` si rivolge insieme all'IA e all'essere umano.

Per l'IA costituisce una memoria di lavoro permanente.

- Caricato a ogni apertura, le indica i punti fermi a cui può ricorrere, le incertezze che deve preservare e i debiti che non deve dimenticare.
- Senza di esso dovrebbe ricostruire quello stato a partire dalle fonti e dalle note a ogni nuova seduta.

Per me svolge il ruolo di un cruscotto.

- In poche righe posso verificare che cosa il `Contesto` consideri stabilito, che cosa consideri ancora incerto e quali lavori restino incompleti.
- Posso allora correggere quello stato se non corrisponde alla mia comprensione.

Qualche decina di righe basta così all'IA per riprendere il lavoro senza confondere i punti fermi con le ipotesi, e a me per controllare la comprensione su cui si appoggia.

`todo.md` ne ricava le azioni:

```
## In corso
- T-001 - Preparare un colloquio sull'uso reale dei campi
  obbligatori.

## Da fare
- T-002 - Verificare quali dati servono agli audit.

## Fatto
- T-003 - Confrontare i primi due colloqui. - concluso il
  2026-09-10
```

Non apro necessariamente `todo.md` per modificarlo da solo. Posso semplicemente dire:

Aggiungi un compito per verificare quali dati servono agli audit.

Il `Contesto` crea una riga, le assegna un identificativo e la mette nella sezione giusta.

Un compito può anche comparire durante il lavoro.

- Se l'analisi di una fonte fa emergere una verifica necessaria, l'IA mi propone di aggiungerla invece di lasciarla sparire nella conversazione.
- Non trasforma in silenzio ogni idea in un nuovo obbligo: quando un compito cambia le priorità o comporta lavoro in più, la scelta resta mia.

Quando il lavoro è finito, posso chiedere:

Chiudi il compito T-002 (oppure una descrizione approssimativa del compito, e l'IA ritroverà il riferimento giusto).

Il Contesto non cancella la riga. La sposta in Fatto, aggiunge la data di chiusura e, quando è utile, indica il risultato o l'elemento che permette di considerarlo concluso.

- Un compito concluso non sparisce.
- La sua presenza mi evita di rifare lo stesso lavoro e mi spiega perché una pista ha smesso di essere attiva.

todo.md non è dunque un elenco scritto una volta per tutte. È la coda delle azioni del Contesto: l'IA lo consulta per sapere che cosa resta da fare, lo aggiorna quando il lavoro avanza e conserva i compiti conclusi per evitare di ricominciarli.

Da dove riparto

handoff.md è l'ultimo file del nucleo. Viene riscritto a ogni salvataggio per rispondere a una situazione molto concreta: qualcuno arriva adesso, una persona o un'IA, senza niente

della conversazione precedente. Che cosa deve sapere per ripartire senza sbagliare?

Ripresa

A che punto siamo

Tre fonti hanno fatto emergere una tensione fra facilità di compilazione e sfruttamento dei dati.

Che cosa sapere prima di continuare

I due colloqui descrivono situazioni particolari, mentre il white paper sostiene un punto di vista generale.

Questo insieme fa emergere una tensione, ma non permette ancora di misurarne l'ampiezza né di concludere che esista in tutte le organizzazioni.

La prossima azione

Preparare le domande che permetteranno di individuare i campi realmente usati.

Che cosa è rimasto aperto

Quanta struttura serva agli audit resta sconosciuto.

5

La prossima azione è unica e immediatamente eseguibile. Un elenco di dieci possibilità mi restituirebbe quel lavoro di scelta delle priorità che questo file esiste proprio per risparmiarmi.

Il comando di apertura assorbe dunque quello che è stato depositato durante il sonno del Contesto, legge questo nucleo, poi mi annuncia quell'azione.

- Ritrova il suo scopo prima di ritrovare i suoi dettagli, e le mie due settimane di assenza non costano più di un minuto di lettura.

-
- La seduta può ora ricevere materiale nuovo senza perdere la propria direzione.
-

DA RICORDARE — IL NUCLEO DI APERTURA

Riaprire un *Contesto* non consiste nel rileggerne il contenuto, ma nel caricare un piccolo nucleo: la missione, le direttive, lo stato e la ripresa. La missione dice dove andare, le direttive fissano le condizioni del percorso. Una ripresa utile indica una sola azione, immediatamente eseguibile.

Alimentare: far entrare il materiale

La seduta comincia, e tre cose arrivano insieme.

- Un responsabile delle operazioni mi manda il resoconto di un colloquio,
- un'idea mi viene durante lo scambio,
- una nota che avevo lasciato il giorno prima aspetta in una Inbox (gestore di note, e-mail...).

Questi tre materiali non entrano nel Contesto allo stesso modo, e ognuno ha il suo skill.

Un documento arriva come oggetto finito

Un documento esterno viene conservato con il file d'origine, il suo testo utilizzabile e una scheda che permette di seguirlo.

```
03-sources/documents/  
├─ SRC00001-entretien-exploitation.md  
├─ raw/  
│   └─ SRC00001-entretien-exploitation.pdf  
└─ fulltext/  
    └─ SRC00001-entretien-exploitation.md
```

La scheda SRC00001-entretien-exploitation.md può prendere questa forma:

```
---
added: 2026-09-11
kind: pdf
intention: colloquio
raw: raw/SRC00001-entretien-exploitation.pdf
fulltext: fulltext/SRC00001-entretien-exploitation.md
extraction: ok
reliability: medium
---
```

Colloquio con la responsabile delle operazioni

Perché è qui

Capire che cosa impedisce ai tecnici di produrre rapporti di intervento utilizzabili.

Interpretazione – non analizzata

La responsabile descrive un'opposizione fra la rapidità della compilazione sul campo e il bisogno di dati strutturati per l'analisi.

Copertura

- Compilazione da mobile
- Campi obbligatori
- Riutilizzo dei dati

Che cosa ha prodotto

- Da completare dopo l'analisi

Questa scheda non riassume per forza il documento. Risponde anzitutto a domande di gestione:

- da dove viene,
- perché l'ho aggiunto,
- posso leggerne il contenuto,
- quanta fiducia accordargli?

La sezione Interpretazione – non analizzata mette in quarantena una prima lettura. Evita che un'impressione formulata al momento dell'importazione venga subito trattata come conoscenza stabilita.

Quando una fonte arriva sotto forma di PDF, di pagina web o di un altro formato difficile da sfruttare direttamente, l'estrazione ne produce una versione testuale leggibile dall'essere umano e dall'IA.

- Quest'operazione non produce ancora nessuna conoscenza: rende soltanto accessibile il materiale senza modificare il documento originale.

La trasformazione viene dopo.

- Il testo estratto viene analizzato per far emergere note atomiche, note tematiche, termini di glossario o arbitrati.
- Queste note alimentano il *mondo delle idee*, dove il materiale delle fonti diventa un insieme di mattoni di conoscenza riutilizzabili.

Il grado di attendibilità non giudica le fonti aggiunte.

- Indica semplicemente come usare quel materiale: testimonianza isolata, dato verificato, documento contrattuale o ipotesi da confermare.

Non tutto il materiale utile esiste già in un documento. Una parte si trova nella mia esperienza, nei miei ricordi, nelle mie osservazioni o nelle distinzioni che formulo lavorando.

Quando le esprimo durante una seduta di scambio con l'IA, anche queste entrano nel `Contesto`.

- Non diventano automaticamente fatti stabiliti
- Conservano la loro origine e potranno essere messe a confronto con altre fonti.

Aggiungere del materiale consiste dunque tanto nell'inserire un documento quanto nel conservare quello che la persona a cui appartiene il `Contesto` porta nella conversazione.

La conoscenza della persona entra attraverso la seduta

Una sessione non è un documento consegnato una volta per tutte. Si costruisce nel corso dello scambio. Viene registrata in `03-sources/sessions/SESSION-2026-09-12.md`, che può prendere questa forma:

```
# Sessione del 12 settembre 2026

## 09:41 – IA

I campi obbligatori migliorano davvero la qualità, oppure
spostano soltanto la fatica sul tecnico?

## 09:42 – Autore

Nei colloqui, i campi obbligatori garantiscono soprattutto
che un valore sia stato inserito.
Non ne garantiscono né la precisione né l'utilità.
Quando il tecnico non capisce a che cosa servirà poi un
campo, lo compila il più in fretta possibile per andare
avanti.
Il vincolo migliora dunque la completezza apparente, ma può
spostare la fatica senza migliorare l'informazione
davvero utilizzabile.

...
```

Non tutto quello che dice l'IA ha bisogno di essere conservato. Una domanda, un'obiezione o una riformulazione che cambia il lavoro, invece, se lo merita.

- La mia parola e quella dell'IA restano separate.
- Posso così ritrovare l'origine di un'idea e distinguere la mia intenzione da una proposta del sistema.

Una sessione può essere disordinata.

- Il suo ruolo è catturare il movimento reale del pensiero, non presentare una sintesi perfetta.

La cassetta delle lettere parla al Contesto quando è chiuso

Un'idea mi viene fuori seduta, in treno o nel mezzo di una riunione e senza accesso a Internet.

- Entra allora nella cassetta delle lettere.
- Questa cassetta è il file `offline.md`, posto alla radice del Contesto.
- Mi basta scrivere in questo file Markdown.

Può prendere questa forma:

Chiedere se i tecnici aggirano i campi obbligatori con valori generici.

Alla prossima apertura del contesto, quella nota viene riletta, avvicinata al lavoro in corso e poi assorbita al posto giusto. Può diventare un compito, una domanda da colloquio, una nota, oppure sparire se non è più pertinente.

Aggiungere non è ancora capire

Una volta entrato questo materiale, il mio albero contiene più fonti e la traccia della seduta in corso.

```
03-sources/  
├── documents/  
│   ├── SRC00001-entretien-exploitation.md  
│   ├── SRC00002-entretien-exploitation.md  
│   └── SRC00003-livre-blanc-donnees-intervention.md  
└── sessions/  
    └── SESSION-2026-09-11.md
```

Questi file provano che il materiale è disponibile. Non provano che sia stato capito.

Il file `03-sources/ingestion-log.md` distingue le due operazioni. Per ogni fonte indica se l'analisi è completa, parziale o da rivedere, con un'impronta del contenuto e qualche nota di lavorazione.

Questa distinzione diventa essenziale quando un documento viene sostituito, quando un'estrazione cambia o quando una seduta si interrompe. Il Contesto può allora riprendere l'analisi senza far finta di sapere tutto.

Il materiale è entrato. Mi resta da trasformarlo in idee abbastanza precise da poter essere collegate, discusse e riutilizzate.

DA RICORDARE — LA FONTE

Un documento, una parola di seduta e una nota fuori seduta entrano da tre porte diverse e conservano ciascuna la propria forma d'origine. Una fonte conserva il materiale e la prova, mai l'interpretazione che se ne trarrà. Aggiungere non è capire: il registro di ingestione distingue ciò che è presente da ciò che è assimilato.

Trasformare: passare dalle fonti alle idee

Ho adesso tre fonti davanti a me.

- Due colloqui descrivono l'attrito provocato dai moduli,
- Un white paper sostiene la necessità di dati strutturati.

La tentazione sarebbe decidere subito e scrivere che i campi obbligatori danneggiano la qualità. Sarebbe andare troppo in fretta: il `Contesto` deve preservare questi apporti senza fonderli in un'unica conclusione.

Chiamo dunque lo skill che crea e aggiorna le note: legge quello che non è ancora stato trattato e ne ricava delle idee.

Estraggo un'idea che sta in piedi da sola

Una fonte contiene di rado un'idea già pronta per essere riutilizzata.

- Un colloquio mescola fatti, esempi, opinioni, esitazioni e dettagli propri della situazione raccontata.
- Trasformarlo non consiste nel riassumere tutto, ma nel ricavarne le idee.

Il percorso sta in tre tappe:

Testimonianza tecnici	«Quando il modulo è troppo lungo, certi
	mettono un valore generico per poter
finire.»	
↓	
Osservazione	Valori generici vengono a volte usati per
	accelerare la compilazione.
↓	
Idea fissata impoverire	Una compilazione troppo gravosa può
	i rapporti di intervento.

Un'idea forte non è una formula spettacolare né una verità definitiva.

- Ha un baricentro chiaro e saldo,
- Si può capire fuori dal testo che l'ha fatta nascere
- Resta abbastanza delimitata da poter essere confermata, sfumata o contraddetta.
- Conserva inoltre la propria origine e i propri limiti, per non trasformare una testimonianza particolare in regola generale.

L'IA legge la fonte alla luce della missione e delle direttive, poi la mette a confronto con le note già presenti. Crea una nota nuova quando l'idea è nuova; altrimenti arricchisce quella esistente con la sfumatura o il limite in più.

Fa questo lavoro senza chiedermi di approvare ogni formulazione. Posso poi rileggere e correggere le note.

- Quando durante la seduta compare un'ambiguità, mi interroga;

- Se più tardi scopre una contraddizione che non riesce a risolvere, la conserva invece di decidere al posto mio.

```
04-notes/  
└─ atomic/  
    └─ AT000001-la-contrainte-de-saisie-peut-appauvrir-les-comptes-rendus.md
```

```
---
id: AT000001
status: active
sources: [SRC00001, SRC00002]
---

# Una compilazione troppo gravosa può impoverire i rapporti
  di intervento

## Idea principale

Quando il modulo rallenta l'intervento, i tecnici possono
  inserire valori generici o ridurre la loro descrizione al
  minimo atteso.

## Perché è importante

Un dato formalmente strutturato può sembrare pulito e
  descrivere comunque peggio la realtà sul campo.

## Sfumature e limiti

I due colloqui non permettono di misurare la frequenza del
  fenomeno.

## Origine / prova

- SRC00001: esempi di valori generici usati per finire più
  in fretta.
- SRC00002: calo di precisione osservato dopo l'aggiunta di
  campi obbligatori.

## Collegamenti

- Contrasto con AT000002.
```

Il titolo porta un'affermazione, non soltanto un tema. La nota resta comprensibile senza che io debba riaprire i documenti da cui proviene, e conserva insieme i collegamenti che mi permettono di verificarne l'origine.

Una nota atomica non è fissata al primo getto.

- Può guadagnare una sfumatura, una nuova prova o una relazione senza perdere la propria identità.

Il white paper produce una seconda nota: i dati strutturati sono necessari per confrontare, aggregare e automatizzare.

- Questa idea resta separata dalla prima perché afferma qualcos'altro.

Mi rifiuto di decidere troppo presto

Le due idee sembrano opporsi, e non ho ancora abbastanza materiale per arbitrarle. Apro dunque una contraddizione.

C-001 – Strutturare senza peggiorare la compilazione

- Da un lato: i campi imposti possono impoverire le descrizioni sul campo.
- Dall'altro: lo sfruttamento su larga scala richiede dati confrontabili.
- La posta in gioco: stabilire quali informazioni debbano essere strutturate al momento della compilazione e quali possano esserlo dopo.
- Aperta il: 2026-09-11

5

La contraddizione mi evita due errori simmetrici: scegliere troppo presto una parte, oppure smussare il disaccordo in una formulazione vaga che non dà fastidio a nessuno.

- Resta aperta finché una decisione, una nuova fonte o una riformulazione non la risolve.

Non ogni incertezza merita questo trattamento.

- Lo riservo alle tensioni che pesano davvero sul seguito del ragionamento o sulla qualità di una decisione.

Collego quello che queste idee dicono insieme

Durante l'ingestione, l'IA non crea soltanto note atomiche.

- Esamina anche le relazioni fra le idee estratte e le note già presenti.
- Quando più d'una formano un insieme che non si può ridurre a una sola affermazione, crea nella stessa operazione una nota tematica e pone i collegamenti che le uniscono.

Non devo lanciare un secondo comando.

- La creazione delle note atomiche, il loro avvicinamento e l'eventuale comparsa di una nota tematica fanno parte di una stessa ingestione.

```
---
id: THM00001
status: active
sources: [AT000001, AT000002]
---
```

La qualità dipende dalla compilazione e dallo sfruttamento

Angolatura

La qualità di un rapporto di intervento non può essere valutata solo dal numero di campi compilati.

Sintesi

Il sistema deve ottenere abbastanza struttura per sfruttare i dati senza rendere la compilazione artificiale o costosa.

Note collegate

- AT000001 – La gravosità della compilazione può impoverire i rapporti di intervento.
- AT000002 – Dati strutturati sono necessari allo sfruttamento.

Tensioni

- C-001 – Strutturare senza peggiorare la compilazione.

La nota tematica non giustappone soltanto dei riassunti. Propone una lettura delle loro relazioni e mostra che cosa resta aperto.

Mi accorgo che una parola non ha lo stesso senso per tutti

Una parte del disaccordo viene a volte dal lessico. Il glossario fissa allora il senso scelto in questo Contesto.

Dato strutturato

Informazione registrata secondo una forma esplicita che ne permette il confronto o il trattamento automatico.

Da non confondere con: campo che l'utente è obbligato a compilare al momento dell'intervento.

Esempi: tipo di guasto, durata del fermo, apparecchiatura interessata.

Questa definizione locale non ha la pretesa di diventare universale.

- Impedisce semplicemente a due note, o a due persone, di usare la stessa parola per parlare di cose diverse.

Mattoni neutri di fronte al deliverable

Una nota appartiene al *mondo delle idee* finché cerca di fissare una conoscenza invece di produrre un effetto su un pubblico.

- Non deve dunque diventare troppo presto l'aggancio di un articolo, una slide, un argomento commerciale o un pezzo di specifica.

Questa neutralità non significa che la nota resti vaga.

- Al contrario: il suo titolo afferma qualcosa di preciso, il suo contenuto ne espone le sfumature e i suoi collegamenti permettono di controllarne le fondamenta.
- Quello che resta neutro è la sua forma: non è organizzata per servire un solo risultato.

AT000001 potrà così diventare una domanda sulle scappatoie nella guida al colloquio, un rischio nella nota di inquadramento o un esempio nella presentazione al team.

AT000002 potrà sostenere una domanda sugli usi del dato, giustificare un requisito di prodotto o spiegare al team perché il testo libero non sempre basta.

La nota tematica non fonde questi mattoni in un testo definitivo.

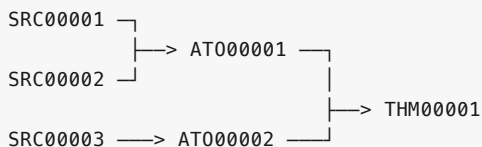
- Indica che cosa permettono di capire insieme e conserva la tensione che andrà trattata in modo diverso a seconda del deliverable.

Questo strato intermedio evita di ricominciare l'analisi per ogni produzione.

- Le fonti restano le prove,
- le note diventano conoscenze a cui ricorrere,
- ogni deliverable compone con esse secondo il proprio bisogno.

Posso risalire da ogni idea alla sua prova

A questo punto, le relazioni si rappresentano così:



I collegamenti nominati mi permettono di distinguere una fonte che conferma un'idea da una che la sfuma o la contraddice.

- Un livello di fiducia rende visibile quello che poggia su più osservazioni e quello che resta ipotetico.

La copertura dell'ingestione mostra quali fonti abbiano davvero contribuito alle note. Mi impedisce di considerare assimilato un documento semplicemente presente nella cartella.

Queste note non sono scritte per un deliverable particolare.

- Formano una conoscenza comune che potrà alimentare più risultati senza essere ricopiata né rinchiusa in uno di essi.

****Il *mondo delle idee* è proprio questo capitale riutilizzabile: un insieme di mattoni abbastanza saldi da essere ritrovati e abbastanza indipendenti da essere ricombinati.****

5

DA RICORDARE — LA NOTA ATOMICA

Un'idea fissata porta un'affermazione e non un tema, e si capisce senza riaprire il documento da cui proviene. Una contraddizione si conserva finché pesa sul ragionamento, invece di essere decisa troppo presto o smussata. Ogni nota conserva il collegamento con le proprie fonti, il che permette di risalire dall'idea alla sua prova.

Produrre e decidere

La mia nota tematica ha fatto emergere una domanda precisa: quali informazioni vanno inserite alla fine dell'intervento, e quali possono essere completate dopo?

Adesso tre persone aspettano qualcosa da me.

- Un tecnico che devo intervistare,
- un comitato che deve decidere,
- un team a cui devo spiegarlo.

Non condurrò tre indagini: ricorrerò tre volte alle stesse note. È l'unico momento della seduta in cui nessuno skill mi aiuta: produrre un deliverable è un lavoro di scrittura, non un gesto che si ripete.

Ricorro tre volte allo stesso fondo

Il passaggio dalle note ai deliverable non è una copia. Ogni risultato sceglie i mattoni che gli servono, li ordina e li riformula in funzione del proprio destinatario.

Mattone comune	Guida al colloquio	Nota di inquadramento	Presentazione al team
AT000001 — Una compilazione troppo gravosa può	Chiedere quali campi vengono aggirati e come	Descrivere il rischio di perdita di qualità	Mostrare un esempio di attrito sul campo

Mattone comune	Guida al colloquio	Nota di inquadramento	Presentazione al team
impoverire i rapporti di intervento			
AT000002 — Dati strutturati sono necessari allo sfruttamento	Chiedere quali dati vengono davvero riutilizzati	Stabilire il bisogno di confronto e di automazione	Spiegare il valore atteso per il team
THM00001 — La qualità dipende dalla compilazione e dallo sfruttamento	Costruire il colloquio intorno alle due esigenze	Formulare la posta in gioco e i criteri di scelta	Costruire il filo del racconto
C-001 — Strutturare senza peggiorare la compilazione	Trasformare la tensione in domande da verificare	Conservarla come problema da risolvere	Renderla visibile senza annunciare una soluzione prematura

Lo stesso fondo produce così tre assemblaggi, senza che nessun deliverable riprenda tutto né sostituisca le note che lo alimentano.

Questa separazione mi permette soprattutto di correggere la conoscenza una volta sola.

- Se una nuova fonte sfuma AT000001, i miei prossimi deliverable ripartiranno da quel mattone arricchito, invece di riprodurre una vecchia formulazione sparsa in più documenti.

Comincio dal più urgente: il prossimo colloquio

Il mio primo bisogno è preparare il colloquio con un tecnico.

Il deliverable comincia nello spazio delle bozze.

```
05-outputs/  
└─ _draft/  
    └─ questions-prochain-entretien.md
```

5

```
# Domande per il prossimo colloquio
```

1. Quali campi la rallentano di più durante un intervento?
2. Che cosa inserisce quando l'informazione richiesta non è disponibile?
3. Quali informazioni potrebbe aggiungere dopo l'intervento?
4. Quali dati devono essere esatti subito per servire allo sfruttamento?

Le note cercano di conservare un'idea riutilizzabile. Il deliverable, invece, sceglie un destinatario, una forma e un'azione attesa.

Una volta riletto e approvato, il file lascia `_draft/` e passa direttamente in `05-outputs/`. Lo spostamento indica un grado di maturità, non una differenza di genere documentario.

Finora non ho fatto che produrre: ho dato forma a quello che già sapevo, per tre pubblici. Il seguito di questo capitolo appartiene a un gesto del tutto diverso. A un certo punto bisogna smettere di restituire e decidere — e quello che decido deve poter essere ritrovato fra sei mesi.

Il decision record conserva la scelta che plasma il risultato

«Intervistare un tecnico prima di redigere una specifica» non è un arbitrato su quello che sarà consegnato.

- Se quel colloquio è semplicemente la prossima azione, appartiene a `01-state/todo.md`.
- Se tutti i lavori di esplorazione devono rispettare quella regola, essa diventa una direttiva in `00-frame/directives.md`.

In nessuno dei due casi giustifica un decision record.

Il decision record compare quando il lavoro fa emergere più direzioni possibili e bisogna scegliere quella che strutturerà il prodotto, il testo, il metodo o un altro risultato del `Contesto`.

Conserva allora quello che si è deciso di progettare o di consegnare, le alternative scartate, le ragioni della scelta e le conseguenze accettate.

Nel mio esempio, l'arbitrato non riguarda dunque l'organizzazione dei colloqui. Riguarda una decisione funzionale:

- imporre tutte le informazioni alla fine dell'intervento,
- conservare soltanto del testo libero,
- oppure permettere che il rapporto venga completato in un secondo momento.

Un decision record utile non delega il cuore della scelta alla persona che la metterà in pratica. Tre persone che lo leggono devono capire lo stesso perimetro, gli stessi comportamenti attesi e gli stessi limiti. I dettagli tecnici che non strutturano nulla possono restare aperti, ma nessuno di essi deve poter modificare in silenzio la decisione.

```
04-notes/  
└─ decision-records/  
    └─ DRE00001-limiter-la-cloture-a-quatre-informations.md
```

Ecco che cosa viene fuori una volta scritto. Il modello è lungo perché conserva tutto il ragionamento che rende la decisione comprensibile e rivedibile — chi conosce già i decision record può saltare direttamente al commento che segue.

```
---  
uuid: 0d5efbe2-1eac-4ad7-94c1-29f68adcf001  
id: DRE00001  
created: 2026-09-11  
updated: 2026-09-11  
status: accepted  
decided_by: team di prodotto  
decided_on: 2026-09-11  
supersedes: null  
superseded_by: null  
sources: [AT000001, AT000002, THM00001]  
---
```

Limitare la chiusura di un intervento a quattro informazioni obbligatorie

Decisione

Perché un intervento sia considerato chiuso devono essere presenti esattamente quattro informazioni:

1. L'apparecchiatura interessata, già indicata sulla scheda di intervento oppure inserita al momento della chiusura.
2. La data e l'ora reali di fine dell'intervento.
3. Lo stato dell'apparecchiatura alla fine dell'intervento: rimessa in servizio, lasciata in funzionamento ridotto o lasciata fuori servizio.
4. Un riassunto non vuoto dei lavori svolti o della ragione per cui l'apparecchiatura non ha potuto essere rimessa in servizio.

Nessun'altra informazione condiziona la chiusura.

Dopo la chiusura, il rapporto può essere completato con la natura del guasto, la sua causa individuata, la natura dei lavori svolti, i pezzi sostituiti e la durata di indisponibilità dell'apparecchiatura, senza rimettere in discussione la chiusura dell'intervento.

Il tecnico può inserire queste informazioni aggiuntive, ma il responsabile delle operazioni deve completarle o approvarle entro ventiquattro ore.

Un rapporto ancora incompleto resta consultabile.

Viene segnalato come da completare, e dopo ventiquattro ore il responsabile delle operazioni viene avvisato.

Contesto

I tecnici devono poter chiudere il proprio lavoro sul campo senza percorrere un modulo di analisi completo.

Il reparto operativo ha comunque bisogno di informazioni confrontabili per analizzare i guasti, le loro cause, i lavori svolti, i pezzi sostituiti e le durate di indisponibilità.

Ipotesi

- La scheda di intervento identifica di norma l'apparecchiatura prima dell'inizio dei lavori.
- Il tecnico conosce lo stato dell'apparecchiatura e può descrivere brevemente i lavori svolti al momento della chiusura.
- Il responsabile delle operazioni dispone del tempo e delle informazioni necessarie per approvare il rapporto completo entro ventiquattro ore.

Alternative

- Esigere le nove informazioni al momento della chiusura.
- Conservare soltanto un riassunto libero ed estrarne automaticamente le informazioni sugli interventi in seguito.
- Rendere tutte le informazioni facoltative e controllare la qualità solo a posteriori.

Ragioni

Le quattro informazioni immediate bastano a individuare l'intervento, a datarne la fine, a conoscere lo stato dell'apparecchiatura e a capire i lavori svolti.

Le cinque informazioni di analisi restano necessarie, ma non devono impedire al tecnico di finire il lavoro sul campo.

Il secondo momento attribuisce esplicitamente al responsabile delle operazioni la responsabilità di completare o approvare il rapporto.

Base

- AT000001 descrive le scappatoie legate a una compilazione troppo vincolata.
- AT000002 stabilisce il bisogno di dati confrontabili.
- THM00001 collega le due esigenze.

Conseguenze

Positive

- La chiusura richiede solo quattro informazioni note subito.
- Le informazioni necessarie alle analisi degli interventi restano disponibili dopo che il rapporto è stato completato.
- La responsabilità dei dati mancanti è visibile.

Compromessi

- Gli indicatori degli interventi possono restare incompleti nelle ventiquattro ore successive a un intervento.
- Il responsabile delle operazioni riceve un carico di approvazione in più.

Rischi

- Rapporti incompleti possono accumularsi.
- Informazioni aggiunte in seguito possono rispecchiare meno fedelmente la situazione riscontrata sul campo.

Criteri di conformità

- Un intervento non può essere chiuso se manca una delle quattro informazioni immediate.
- Un intervento può essere chiuso senza nessuna delle cinque informazioni aggiuntive.
- Le informazioni aggiuntive possono essere modificate senza cambiare lo stato di chiusura dell'intervento.
- Il rapporto viene presentato come «da completare» oppure «completo e approvato» in tutti gli usi che se ne fanno.
- Il responsabile delle operazioni viene avvisato ventiquattro ore dopo la chiusura se il rapporto non è completo e approvato.

Decisioni collegate

- Risolve C-001 – Strutturare senza peggiorare la compilazione.

Fonti

- SRC00001
- SRC00002
- SRC00003

Lo stato `accepted` indica qui che l'arbitrato è approvato e abbastanza preciso da guidare la realizzazione.

- Finché le informazioni attese, le responsabilità o i criteri di conformità restano indeterminati, il decision record deve conservare lo stato `proposed`.
- Una volta accettato, viene fissato come traccia storica e non cambia più in silenzio.

Se una nuova decisione lo sostituisce, il vecchio decision record resta consultabile e i due file sono collegati da `supersedes` e `superseded_by`.

Questa continuità conserva la cronologia completa degli arbitrati: quello che era stato deciso, su quali informazioni, e poi quale elemento ha portato a scegliere diversamente. Sei mesi dopo, dunque, non ritrovo soltanto la regola attuale. Posso capire perché una prima direzione era sembrata giusta, che cosa l'ha resa insufficiente e come si è costruita la decisione successiva.

Senza questa cronologia, una successione di scelte finisce per somigliare a un'incoerenza. Con essa, torna a essere la storia comprensibile di un prodotto che cambia.

- Le alternative mostrano che cosa è stato scartato.
- I compromessi rendono visibile che cosa è stato sacrificato.

- La base collega la decisione alle conoscenze e alle prove che la sostengono.

Produrre non esaurisce quello che so

La guida al colloquio, la nota tematica e il decision record trattano lo stesso problema senza sovrapporsi, perché svolgono tre funzioni diverse.

- La nota conserva quello che il **Contesto** crede di sapere.
- La decisione conserva una scelta e le sue ragioni
- La guida trasforma quella conoscenza in azione presso un destinatario preciso.

Un altro deliverable potrà riprendere la stessa conoscenza comune sotto una forma diversa. **Produrre non svuota**

dunque il **Contesto della sua sostanza: rivela che cosa quella sostanza permette di fare.**

Mi resta da registrare lo stato raggiunto, perché la prossima seduta riparta senza dover ricostruire il cammino percorso.

5

DA RICORDARE — IL DECISION RECORD

Le stesse note alimentano più deliverable, e nessun deliverable consuma la conoscenza che usa. Un decision record conserva un arbitrato che plasma quello che sarà progettato o consegnato, non un'azione da svolgere. Conserva quello che è stato scartato e quello che è stato sacrificato, perché la scelta resti rivedibile.

Salvare e riprendere

Il decision record è appena stato accettato e fra tre minuti comincia una riunione. Non ho finito né lo studio né tantomeno una tappa che assomigli a una fine.

È il caso ordinario: un altro compito reclama la mia attenzione, la macchina deve riavviarsi, cambio computer, oppure la finestra di contesto dell'IA diventa troppo carica.

- Salvare non risponde dunque a un bisogno del lavoro stesso, del tipo «lo studio è finito».
- È una garanzia tecnica: lo stato del `Contesto` deve sopravvivere alla sessione in cui è stato prodotto.

Devo poter interrompermi adesso, riaprire più tardi e ritrovare il lavoro esattamente dove l'ho lasciato.

Se ne occupa uno skill, e deve stare in pochi secondi: un salvataggio che costa caro non verrà lanciato nel momento in cui si va di fretta, cioè nel momento in cui serve di più.

L'IA fissa quello che si sa in questo istante

Il file `01-state/state.md` riassume quello che è abbastanza stabile da orientare il seguito.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ state.md
```

Al momento del salvataggio può prendere questa forma:

```
# Stato attuale
```

```
## Conoscenza fissata
```

- Una compilazione troppo gravosa può impoverire i rapporti di intervento.
- Lo sfruttamento su larga scala richiede tuttavia dati strutturati.

```
## Decisione presa
```

- La chiusura richiede l'apparecchiatura interessata, la data e l'ora di fine, lo stato dell'apparecchiatura e un riassunto dei lavori svolti.
- La natura e la causa del guasto, la natura dei lavori, i pezzi sostituiti e la durata di indisponibilità vengono completati dopo la chiusura e approvati entro ventiquattro ore dal responsabile delle operazioni.

```
## Incertezza principale
```

- Misurare se questa separazione riduca davvero il tempo di chiusura senza peggiorare la qualità dei dati.

```
## Debito di conoscenza
```

- Nessun progetto pilota ha ancora misurato il tempo di chiusura né la quota di rapporti completi dopo ventiquattro ore.

- Un compito concluso può sparire dalla lista delle cose da fare.
- Una contraddizione non risolta resta visibile nello stato.

La sintesi contiene soltanto quello che il lavoro ha fissato a sufficienza. Lo stato viene riscritto per descrivere il presente, mentre la sintesi si arricchisce a poco a poco per conservare la comprensione acquisita.

Preparo una sola azione per il me futuro

Il file di ripresa non racconta tutta la storia: prepara il prossimo gesto utile.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ handoff.md
```

Il file 01-state/handoff.md può prendere questa forma:

```
# Ripresa  
  
## Stato attuale  
  
DRE00001 è accettato e C-001 è risolta.  
Le quattro informazioni di chiusura e le cinque  
informazioni aggiuntive sono fissate.  
  
## Punto di attenzione  
  
Non aggiungere un'informazione obbligatoria alla chiusura  
senza aprire un nuovo decision record che sostituisca  
DRE00001.  
  
## Prossima azione  
  
Avviare la realizzazione di DRE00001.  
  
## Aperto  
  
- Definire le diciture mostrate ai tecnici per i tre stati  
  possibili dell'apparecchiatura dopo l'intervento.  
- Preparare la misura del tempo di chiusura e della quota  
  di rapporti completi dopo ventiquattro ore.
```

Una ripresa utile indica una sola prossima azione. Un elenco di dieci possibilità costringerebbe la seduta seguente a rifare quel lavoro di scelta che ho appena fatto.

Conservo una traccia per confrontare le mie riprese

Il file `01-state/checkpoints.md` conserva una serie di fotografie compatte della situazione.

```
contexte-qualite-comptes-rendus/  
└─ 01-state/  
   └─ checkpoints.md
```

Il file può prendere questa forma:

```
## 2026-09-11  
  
- Fonti: 3  
- Note atomiche: 2  
- Note tematiche: 1  
- Contraddizioni aperte: 0  
- Decision record accettati: 1  
- Prossima azione: avviare la realizzazione di DRE00001.  
  
## 2026-09-04  
  
- Fonti: 1  
- Note atomiche: 1  
- Note tematiche: 0  
- Contraddizioni aperte: 0  
- Prossima azione: raccogliere il punto di vista di chi  
  gestisce gli impianti.
```

Queste fotografie mi mostrano l'evoluzione del Contesto senza obbligarmi a confrontare tutti i file. Mi fanno anche notare una stagnazione, o un debito che cresce senza che me ne accorga.

L'IA annota quello che è successo, non quello che ne pensa

La storia registra gli eventi importanti senza cercare di ricostruire ogni scambio. Nel mio esempio, gli eventi della giornata vengono aggiunti a 02-history/2026-09-11.md.

```
contexte-qualite-comptes-rendus/
└─ 02-history/
   └─ 2026-09-11.md
```

Il blocco aggiunto può prendere questa forma:

```
## 2026-09-11

- Aggiunta di SRC00003-livre-blanc-donnees-intervention.
- Creazione di AT000002 e THM00001.
- Apertura di C-001.
- Approvazione di `questions-prochain-entretien.md`.
- Accettazione di DRE00001 e risoluzione di C-001.
```

5

- Lo stato dice a che punto si è.
- La storia dice che cosa è successo.
- La ripresa dice che cosa fare dopo.

Lascio che la macchina verifichi quello che non vedrei

Al momento di salvare, qualche controllo meccanico riduce le dimenticanze. Il loro risultato viene scritto in 01-state/health.md.

```
contexte-qualite-comptes-rendus/
└─ 01-state/
   └─ health.md
```

- Esistono tutti i file richiamati?

- Gli identificativi sono unici?
- Ogni nota indica le proprie fonti?
- Una fonte aggiunta ma non analizzata viene segnalata?
- Le decisioni sostituite sono collegate a quelle che le sostituiscono?
- La prossima azione è esplicita?

Queste verifiche non sostituiscono il mio giudizio.

Impediscono semplicemente a un errore di struttura di farsi passare per un problema di pensiero.

Il ciclo completo, e quello che permette

Il movimento generale diventa ora visibile.

5

```
apertura
↓
lettura della cassetta delle lettere
↓
caricamento della missione, dello stato e della ripresa
↓
aggiunta di una fonte
↓
creazione o aggiornamento delle note
↓
produzione di una bozza
↓
controllo, approvazione o decisione
↓
aggiornamento dello stato, della sintesi e della ripresa
↓
checkpoint e storia
↓
controllo di salvataggio
↓
prossima apertura
```

Non aspetto la fine di questa catena per salvare.

- Lo faccio dopo l'aggiunta di una fonte, in mezzo alla creazione di una nota, prima di una decisione o durante la preparazione di un deliverable, quando vedo la mia finestra di contesto esplodere...

È proprio questo il ruolo del salvataggio: **rendere ognuno di questi stati riprendibile, indipendentemente dalla conversazione in corso.**

La mia riunione può dunque cominciare.

Il Contesto non è soltanto un albero di file: è un sistema per pensare nella durata, che fa entrare il materiale, preserva le tensioni, fissa delle idee, produce più risultati e prepara la ripresa.

La sua forma resta volutamente semplice e trasportabile — dei file leggibili bastano a rendere il lavoro osservabile, verificabile e trasmissibile.

Una volta capito questo motore, compare un'altra domanda: come dialoga il Contesto con gli strumenti, le basi di conoscenza e il prodotto reale che esistono fuori di esso?

DA RICORDARE — IL SALVATAGGIO

Salvare non segna la fine di una tappa: è la garanzia che lo stato sopravviva alla sessione che l'ha prodotto. Lo stato dice a che punto si è, la storia che cosa è successo, la ripresa che cosa fare dopo. Il ciclo può essere interrotto in qualsiasi punto, e ognuno di questi punti resta riprendibile.

Parte 6

Il Contesto nel suo ecosistema

Un Contesto tenuto bene fa venire voglia di metterci dentro tutta la conoscenza disponibile.

Questa parte esamina che cosa lascia fuori e perché. Resta piccolo per conservare la capacità di ragionare. Tutto ciò che tenesse soltanto per precauzione invecchierebbe senza preavviso mentre l'originale continuerebbe a cambiare altrove. Lavora dunque con dei vicini: strumenti la cui funzione è conservare, misurare o eseguire, e altri Contesti che indagano domande diverse dalla sua.

Interrogare le conoscenze strutturate

I rapporti d'intervento aspetteranno: stamattina lavoro su un altro argomento dello stesso prodotto, le persone che abbandonano a metà dell'iscrizione. Un'agenzia mi ha appena consegnato il suo studio e lo aggiungo al *Contesto*: diversi partecipanti esitano davanti alle regole della password. Questa osservazione mi ricorda un'idea già incontrata: chiedere uno sforzo importante prima che l'utente percepisca il valore del prodotto può indurlo ad abbandonare.

- Sono quasi certo di avere materiale sull'argomento.
- Semplicemente non so più dove si trovi né quali parole avessi usato.

Ho parlato di sforzo iniziale, di attrito, di attivazione o di onboarding? L'idea è forse

- stata tratta da un libro,
- sviluppata in una nota di product management,
- collegata a un altro percorso utente,
- ...

Potrei provare varie ricerche per parole chiave e aprire i risultati uno per uno. Invece chiedo al *Contesto*:

Che cosa abbiamo già imparato sugli sforzi chiesti prima che l'utente percepisca il valore del prodotto?

Il Contesto interroga una memoria più ampia

La risposta non è nel Contesto attivo. Può invece trovarsi nel mio LLM Wiki, una biblioteca comune in cui sono già strutturate le migliaia di note tratte da libri, articoli, norme e altri documenti.

Il LLM Wiki e il Contesto non hanno la stessa natura.

Il wiki è una base di conoscenze.

- Conserva note strutturate (nel mio caso ho riutilizzato il concetto di note atomiche) a partire da quello che altri hanno scritto: una definizione, un principio, il risultato di uno studio, un requisito di una norma.
- Ogni nota dice che cosa si sa, e da dove viene.
- Può contenerne decine di migliaia, accumulate in anni e su argomenti senza rapporto fra loro.

Nessun Contesto conterrà mai quella quantità, e non è il suo ruolo.

Ma il wiki si ferma lì.

- Non dice perché quella conoscenza conti nel lavoro che porto avanti oggi, né che cosa ne ho fatto.
- Non collega un'idea a una decisione, né una decisione a quello che mi è costata.

Il LLM Wiki non è un sistema di pensiero: conserva conoscenza su larga scala, ma non ragiona.

Il ragionamento vive nel Contesto. È lui a conservare il perché del come:

- quale sia la missione,

- quali direttive e decisioni siano acquisite,
- perché quest'idea sia stata scelta per questa missione,
- quali alternative ho scartato,
- che cosa ho accettato di perdere,
- quale contraddizione resti aperta.

Ci riesce solo perché resta piccolo — qualche decina di note che un essere umano può rileggere.

Ciascuno paga il prezzo della propria forza.

- **Il wiki guadagna la scala e perde il filo.**
- **Il Contesto tiene il filo e rinuncia alla scala.**

È proprio per questo che devono parlarsi invece di fondersi.

Descrivo l'idea che cerco

Non copio tutto il wiki nel Contesto dedicato all'iscrizione. E non eseguo nemmeno una ricerca RAG seguita da una ricerca in un grafo.

Pongo semplicemente la mia domanda al Contesto:

Che cosa abbiamo già imparato sugli sforzi chiesti prima che l'utente percepisca il valore del prodotto?

La formulo con parole mie.

- Non ho bisogno di conoscere né il titolo di una nota, né dove si trovi, né il lessico usato quando è stata scritta.

Il Contesto trasmette quest'idea al LLM Wiki tramite la sua API. Tutto il resto avviene dentro il wiki.

Il wiki cerca per senso, poi intorno al risultato

Il LLM Wiki comincia recuperando le note semanticamente vicine alla mia domanda.

- Una nota può parlare di attrito iniziale, di attivazione o di tempo prima del primo beneficio senza usare la parola «sforzo».

È questo che il suo RAG gli permette di fare: trovare conoscenze già strutturate vicine alla mia idea senza esigere lo stesso lessico.

Poi il wiki guarda intorno ai risultati migliori. Il suo grafo semantico segue le relazioni già stabilite fra le note e aggiunge le vicine che possono chiarire l'argomento:

- una definizione,
- un limite,
- una contraddizione,
- un'idea complementare.

Una di queste note vicine non avrebbe forse mai risposto direttamente alla mia domanda. Diventa preziosa lo stesso, perché mostra a quali condizioni l'idea valga, o in quale punto smetta di essere vera.

```
graph TD; A[un'idea formulata con parole mie] --> B[il wiki ritrova le note vicine per senso]; B --> C[aggiunge il loro vicinato utile]; C --> D[tornano alcuni riferimenti con le relative fonti]; D --> E[il Contesto tiene quello che serve alla sua missione];
```

Il RAG ritrova quello che non avrei saputo nominare. Il grafo fa emergere quello che non avrei pensato di cercare.

Per me queste due operazioni ne formano una sola. Chiedo riferimenti che rispondano a un problema; il LLM Wiki cerca l'idea e il suo vicinato, poi mi restituisce un elenco breve.

6

Il Contesto riceve riferimenti, non una risposta già fatta

Il wiki rimanda note esistenti con il loro contenuto, la loro origine e le loro relazioni. Non fabbrica una sintesi che mescolerebbe in silenzio più risultati.

Il Contesto può dunque leggere ogni risultato, confrontarlo con le osservazioni sull'iscrizione e tornare alla relativa fonte quando occorre verificare una prova.

- Certe note riguardano davvero uno sforzo imposto prima dell'accesso al prodotto.
- Altre parlano di uno sforzo chiesto più tardi, quando l'utente ne conosce già il valore.

La loro vicinanza alla mia domanda non basta a renderle equivalenti.

Il `Contesto` fa la cernita a partire dalla sua missione. Può:

- tenere un'idea che chiarisce l'abbandono durante l'iscrizione,
- conservare una contraddizione,
- scartare una nota troppo generale.

Una nota selezionata entra nel `Contesto` con la sua origine e le fonti su cui poggia. Il resto rimane nel wiki.

Dispongo così di una memoria molto più vasta senza trasformare ogni `Contesto` in una copia di tutta la mia biblioteca.

- Non ho bisogno di imparare come quella memoria sia indicizzata per servirmene: descrivo l'idea che cerco, poi verifico i riferimenti che mi riporta.

Questa catena dà accesso alla conoscenza già strutturata.

- Non dice ancora che cosa il prodotto faccia davvero oggi.

Per rispondere a quest'altra domanda, il `Contesto` deve lasciare la biblioteca e interrogare direttamente il codice, i test e il prodotto in funzione.

DA RICORDARE — DUE SISTEMI, UNA PORTA

Il LLM Wiki guadagna la scala e perde il filo; il Contesto tiene il filo e rinuncia alla scala. Descrivo un'idea con parole mie; il wiki ritrova le note vicine per senso e aggiunge il loro vicinato utile. Il Contesto riceve alcuni riferimenti accompagnati dalle relative fonti e tiene soltanto quello che serve alla sua missione.

Interrogare il prodotto reale

Sto rileggendo lo studio sugli abbandoni durante l'iscrizione quando arriva una domanda:

| *Un utente invitato deve creare una password come gli altri?*

La domanda sembra minuscola. La risposta è però attesa da una persona del supporto che deve rispondere a un cliente.

Potrei:

- interrompere la lettura,
- aprire un ambiente di prova,
- preparare due account,
- ripercorrere i due percorsi,
- confrontare quello che succede.

È quello che facevo prima.

- Mezz'ora dopo avrei probabilmente la risposta.
- Avrei anche perso il filo del mio lavoro.

Questa volta pongo la domanda al **Contesto**.

Il Contesto interroga il prodotto

Il **Contesto** lavora con le note uscite dallo studio sugli utenti e conserva quello studio come fonte delle loro affermazioni.

- Porta anche la missione in corso.

- Capisce perché la domanda conti, ma questi elementi non gli permettono di conoscere il comportamento attuale del prodotto.

Non mantengo più una specifica incaricata di descrivere durevolmente quel comportamento.

- Il prodotto cambia troppo spesso perché un documento scritto accanto resti la sua copia fedele.

L'unica verità aggiornata è quello che è stato rilasciato e gira in produzione: il codice sorgente dell'applicazione.

Nel mio sistema, alcuni skill esterni al `Contesto` sanno interrogare il codice sorgente e generare documentazione a partire dall'implementazione. Il `Contesto` chiama quello che risponde alle domande di prodotto e gli trasmette la mia richiesta.

6



Lo skill comincia individuando il percorso di invito nel codice. Poi segue quel percorso fino alla regola di creazione dell'account.

Il codice mostra il comportamento programmato. Un test conferma che un utente invitato può concludere l'iscrizione grazie al link ricevuto, senza creare subito una password. La configurazione indica che nessuna variante propria di un cliente è attiva per quel percorso.

Lo skill mi restituisce una risposta funzionale:

Un utente invitato può accedere al proprio account dal link di invito senza creare una password durante quella tappa. Se poi sceglie di crearne una, valgono le regole ordinarie. I test confermano il percorso di invito e la creazione volontaria della password. Il percorso di recupero di una password dimenticata non è stato esaminato.

Posso trasmettere le prime due frasi al supporto. Posso anche aprire i riferimenti forniti dallo skill se voglio controllarne l'analisi con uno sviluppatore.

Infine posso registrare questa risposta nel `Contesto`, con le sue prove, la data della verifica e il limite segnalato.

- Rispondere al supporto risolve un caso e si ferma lì.
- Registrarla nel `Contesto` fa entrare un fatto verificato sul prodotto nel *mondo delle idee* della missione: la prossima domanda sul percorso di invito non ripartirà da zero, e la nota uscita dallo studio guadagna quello che l'osservazione da sola non dava — il momento esatto in cui ogni percorso chiede la password.

Una risposta vale per le prove che l'accompagnano

Lo skill non si è limitato a produrre una risposta plausibile.

Per ogni affermazione indica che cosa ha consultato:

- la regola presente nel codice,
- il test che verifica il comportamento,
- la configurazione che determina la variante attiva.

Questa traccia protegge da una risposta convincente ma falsa.

Permette anche di vedere subito quello che manca.

Qui lo skill non ha verificato il recupero di una password dimenticata. Lo dice invece di estendere la propria conclusione a un percorso che non ha letto.

A seconda della domanda, altri elementi possono diventare necessari.

- Il codice contiene le regole possibili, ma un feature flag indica quale è attiva.
- La struttura del database mostra che cosa il prodotto può registrare, mentre i dati consentiti permettono di capire la situazione di un account preciso.
- Lo storico Git e le pull request aiutano a ritrovare quando una regola è cambiata e le spiegazioni conservate con quel cambiamento.
- ...

Lo skill raccoglie soltanto quello che gli serve per rispondere alla domanda posta. Non ingloba tutto il repository e non trasforma ogni file in fonte del `Contesto`.

La risposta resta leggibile dal supporto, dalle vendite o da un product manager. Le prove restano disponibili per chi debba verificarla o proseguire l'indagine.

Trovare la regola non decide di cambiarla

Torno al mio studio sull'iscrizione. Mi aveva già insegnato che i due percorsi non presentano la creazione della password nello stesso momento.

Lo skill aggiunge quello che lo studio non poteva vedere.

L'utente invitato non aggira la regola della password: la incontra più tardi, ed è allora esattamente la stessa.

L'attrito non è dunque sparito per lui, ha cambiato posto.

Questa precisione sposta la mia domanda. Non cerco più di sapere chi subisca la regola, ma in quale momento ciascuno la incontri: prima di aver percepito il valore del prodotto, oppure dopo.

Il codice mi ha insegnato che cosa fa il prodotto.

- Non mi dice se la regola sia buona, se protegga davvero gli account o se il suo costo per gli utenti sia accettabile.

Per decidere ho ancora bisogno degli altri elementi del

Contesto:

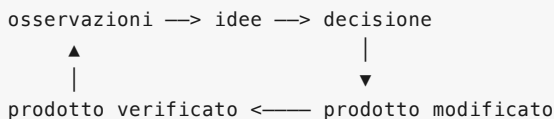
- le osservazioni dello studio e il loro ambito;
- gli obiettivi di sicurezza;
- le ragioni che avevano portato a scegliere quella regola;
- le conseguenze attese da un cambiamento.

Lo skill stabilisce un fatto sul prodotto. Il `Contesto` collega quel fatto alle note e alle decisioni già presenti, conservando

insieme le fonti che permettono di controllarne l'origine. **La scelta resta mia.**

Se decido di modificare la creazione della password, il decision record conserva la scelta, il suo ambito e le ragioni che la sostengono. Dopo la realizzazione, lo skill può tornare al prodotto e verificare che il nuovo comportamento corrisponda davvero alla decisione.

Il movimento funziona dunque nei due sensi. Il prodotto illumina la riflessione, poi la decisione trasforma il prodotto.



Conservare il perché, ricostruire il che cosa

6

L'APPROCCIO CODE-CENTRIC

Il codice non è più soltanto il risultato finale dello sviluppo. Diventa la fonte a partire dalla quale posso ricostruire ciò che descrive il comportamento attuale del prodotto. Il *Contesto* conserva quello che il codice non può spiegare da solo: le fonti, il ragionamento, le decisioni, le alternative scartate e le ragioni della scelta.

Questa separazione cambia il posto della documentazione. Conservo quello che spiega perché il prodotto abbia preso una direzione: i colloqui, i vincoli, le ipotesi e le decisioni.

Ricostruisco dal prodotto quello che descrive il suo comportamento attuale:

- una risposta al supporto,
- una documentazione funzionale,
- un registro dei cambiamenti,
- ...

Non mantengo più manualmente due versioni dello stesso comportamento, una nel codice e l'altra in un documento che comincia a invecchiare dal momento della pubblicazione (e anche ben prima).

Non tutto viene però dal codice.

- Uno studio sugli utenti non si trova lì.
- Un obbligo normativo conserva la propria fonte.
- Una decisione non si riduce al comportamento che ne risulta.

Il codice registra quello che è stato realizzato. Il Contesto conserva il perché e permette di discuterlo quando la situazione cambia.

Tornarci sei mesi dopo

La risposta data al supporto resta legata alla versione del prodotto esaminata oggi. Non descrive automaticamente le versioni successive.

- Sei mesi dopo, il percorso di invito può essere cambiato.
- Un feature flag può attivare un'altra variante, oppure può essere comparso un nuovo modo di accedere.

Il Contesto conserva la domanda, la risposta funzionale, la data della verifica, la versione esaminata, le prove citate e il limite segnalato. Ritrovo così quello che sapevamo quando abbiamo ragionato.

Per conoscere il comportamento presente, chiedo semplicemente allo skill di rifare la verifica.

- Riparte dal prodotto reale invece di ricopiare la vecchia risposta.

Posso allora distinguere due domande che spesso si confondono:

- perché abbiamo preso quella decisione all'epoca?
- che cosa fa il prodotto oggi?

Il Contesto risponde alla prima conservando il ragionamento.

Lo skill risponde alla seconda tornando alla fonte di verità.

Il prodotto non sostituisce dunque il Contesto, e il Contesto non pretende di contenere il prodotto.

- L'uno fornisce una realtà da verificare;
- l'altro conserva le conoscenze necessarie a darle senso e a farla evolvere.

Questa circolazione non si ferma agli strumenti esterni.

Quando più Contesti lavorano su aspetti diversi di uno stesso prodotto, anche le loro conoscenze devono poter circolare senza essere fuse in un'unica cartella.

DA RICORDARE — L'APPROCCIO CODE-CENTRIC

Nessuna specifica viene mantenuta come descrizione duratura del prodotto: ciò che descrive il suo comportamento si ricostruisce dal codice. Una risposta vale solo per le prove che l'accompagnano e per i limiti che segnala. Il codice registra ciò che è stato realizzato; il `Contesto` conserva il perché e permette di rimetterlo in discussione.

Il giorno in cui ho avuto tre Contesti

Un tecnico mi racconta di aver dovuto richiamare un cliente per un'informazione che mancava in un rapporto di intervento.

Annoto la frase. Poi mi fermo, perché non so dove metterla.

- Ho un `Contesto` sulla qualità dei rapporti di intervento — quello che mi ha occupato per tutta la seconda metà di questo libro.
- Ne ho un secondo sull'intervento stesso: che cosa contiene, quali regole lo governano, che cosa è già stato deciso in proposito.
- E ne ho un terzo, più ampio, aperto qualche settimana prima: come portare gli operatori a farsi carico degli interventi di primo livello.

La frase del tecnico riguarda tutti e tre:

- documenta un rapporto carente,
- dice qualcosa dell'intervento,
- chiarisce la questione degli operatori.

È il momento in cui il problema cambia natura.

- Finché avevo un solo `Contesto`, la domanda era: questo entra, oppure è fuori tema?
- Con tre, la domanda diventa: in quale?

I due modi di sbagliare

La prima tentazione è mettere tutto insieme.

- Un unico grande spazio, tutta la conoscenza nello stesso posto, niente più problemi di collocazione.

È esattamente quello da cui scappavo. Uno spazio unico che contiene tutto ricrea il problema dei sistemi di gestione documentale: un luogo in cui

- l'informazione è presente e la cernita ricomincia a ogni consultazione,
- gli scopi si mescolano,
- le fonti si accumulano,
- e nessuna nota ha più un proprietario chiaro.

Ho raccontato questo vicolo cieco all'inizio del libro; ricrearlo con un lessico nuovo non lo renderebbe migliore.

La seconda tentazione è l'opposta: che ogni *Contesto* resti in casa propria. Ciascuno è pulito, delimitato, coerente — e cieco.

Il risultato si vede presto. Il mio *Contesto* sui rapporti ha passato tre sedute a consolidare una regola che quello sull'intervento conteneva già. Non era un doppione visibile: era lavoro rifatto, senza che nessuno potesse accorgersene. Serve dunque una terza via, e comincia col sapere che cosa porta ciascun *Contesto*.

Una cifra davanti al nome

I miei *Contesti* si chiamano 01-saisie-compte-rendu, 02-intervention, 03-autonomie-operateurs, 04-exploitation-catch-all.

La cifra non è un rango, è una convenzione di denominazione. Dice che cosa porta il Contesto prima ancora che lo apra.

- **02- un'entità del prodotto.** Una parte del software che gira in produzione e che ha le proprie regole: 02-facturation in un ERP, 02-intervention nel mio. Il suo contenuto è stabilito. Questo Contesto resta attivo.
- **01- un'ipotesi su un'entità.** 01-saisie-compte-rendu: penso che la compilazione dovrebbe cambiare, non so ancora come. Niente di quello che contiene è acquisito, per questo creo un Contesto 01-, per distinguere bene quello che è stabile (in produzione) da un'ipotesi.
- **03- un problema.** 03-autonomie-operateurs: portare gli operatori a farsi carico degli interventi di primo livello. Non è una funzionalità, attraversa più entità e dà origine a dei 01-.
- **04- il contenitore generale.** Una frase sentita, un PDF, un'intuizione: materiale il cui argomento non ha ancora nessun Contesto. Senza di esso bisognerebbe crearne uno per ogni frammento di idea — ho cominciato così e in poche settimane mi sono ritrovato con più di cinquanta Contesti inconsistenti.

Un 01- finisce sempre per uscire dal lavoro attivo

Il giorno della messa in produzione, la nuova compilazione smette di essere un'ipotesi: diventa una parte dell'intervento. Quello che descrive ormai il prodotto viene proposto a 02-intervention. Il resto resta nel Contesto esplorativo, che viene

poi archiviato. Non ci sono più due posti attivi in cui cercare come funziona un rapporto di intervento.

Un'ipotesi che non approda a nulla viene archiviata insieme a quello che l'ha fermata. Non è un fallimento da cancellare: è quello che eviterà di rifare la stessa strada fra due anni.

Il 04-, invece, si svuota nell'altro senso. Il giorno in cui emerge un problema, creo il Contesto che gli corrisponde, poi torno a vedere che cosa il contenitore generale aveva conservato in proposito. Quello che non serve a niente viene cancellato. Tutto questo risolve una sola domanda — dove metto questo — separando quello che sto studiando da quello che è acquisito.

- Senza questa separazione, ogni ipotesi abbandonata lascia le sue tracce nella descrizione del prodotto.
- E non si osa più supporre niente in un posto in cui tutto dovrebbe essere vero.

L'errore che ho impiegato mesi a vedere

Ho cominciato aprendo 03-tarification. Mi sembrava ovvio: è un grosso argomento, riguarda più entità, dura nel tempo.

Sei mesi dopo conteneva:

- il listino prezzi,
- due confronti con la concorrenza,
- una discussione sugli sconti,
- un ticket di supporto su una fattura sbagliata,
- una nota sull'IVA belga,
- tre schermate di un concorrente,

- ...

Niente di tutto ciò era fuori tema: tutto parlava davvero di tariffazione.

È esattamente il problema. Per rifiutare del materiale bisogna poter dire «questo non mi aiuta». Ora, quel Contesto non cercava niente — **non aveva una domanda**, soltanto un'etichetta.

03-autonomie-opérateurs, invece, ne ha una: come portare gli operatori a farsi carico degli interventi di primo livello.

Un 03- non è dunque un grosso argomento. È una domanda. La tariffazione, da parte sua, non è sparita per questo: attraversa i miei Contesti — 02-facturation ne porta le regole, un 01- può dipenderne — senza comandarne nessuno.

La mia convenzione non ha dunque una casella per i domini, né per molte altre cose. Serve a sistemare quello che ha un posto evidente, e nient'altro. Il resto aspetta nel contenitore generale, e lo ritrovo descrivendo quello che cerco invece di provare a ricordare dove l'avevo messo.

Ora ne ho due, una soluzione che ho imparato per tentativi ed errori. Un unico contenitore generale mescolava quello che veniva dal mio lavoro di prodotto con quello che veniva dall'ambito dell'esercizio industriale — due lessici, due pubblici, due modi di chiamare gli stessi oggetti. È diventato illeggibile più in fretta di quanto si svuotasse.

- 04-exploitation-catch-all riceve ciò che appartiene all'ambito funzionale dei miei clienti: l'esercizio;

- 04-productops-catch-all riceve ciò che riguarda il mio lavoro e la sua organizzazione: product ops.

Ciascuno resta leggibile, quindi ciascuno si svuota.

La frase del tecnico, invece, non ha avuto bisogno del contenitore generale. È andata in 02-intervention, perché è lì che vive ciò che è vero riguardo al prodotto, indipendentemente dalla domanda che mi pongo in questo momento.

Resta da capire come i miei tre Contesti ne vengano a conoscenza.

DA RICORDARE — QUATTRO PREFISSI, UNA SOLA DOMANDA

La cifra davanti al nome dice che cosa porta il Contesto: 02- un'entità che gira, 01- un'ipotesi su quell'entità, 03- un problema che ne attraversa più d'una, 04- lo spazio di attesa in cui si deposita quello che non si sa ancora dove mettere. Un 01- esiste per proporre al suo 02- quello che descrive ormai il prodotto, e poi per essere archiviato. Se non approda a nulla, viene archiviato insieme a quello che l'ha fermato. Un dominio non è un Contesto: ne attraversa più d'uno senza comandarne nessuno. Un Contesto senza domanda non saprà mai che cosa sia fuori tema.

Leggere dal vicino senza trasferirsi da lui

Riprendo 01-saisie-compte-rendu, il Contesto dedicato alla qualità dei rapporti di intervento. Per andare avanti ho bisogno di conoscere le regole dell'intervento stesso: che cosa un tecnico ha il diritto di inserire, che cosa è obbligatorio e che cosa è già stato deciso.

Queste regole non vivono nel mio Contesto. Sono fissate in 02-intervention, il Contesto vicino che descrive quello che esiste davvero nel prodotto.

Potrei ricopiarle per averle sottomano. Sei mesi dopo avrei probabilmente due versioni della stessa regola e nessun modo semplice di sapere quale faccia fede.

Comincio dunque da una relazione più leggera: il mio Contesto dichiara che consulta 02-intervention.

Leggere lascia la conoscenza al suo proprietario

Questa relazione è scritta nel quadro stabile del mio Contesto, accanto alla sua missione e alle sue direttive. Apre un diritto preciso: consultare quello che 02-intervention sa già.

Il materiale resta presso il suo proprietario. Continua a evolvere lì, e la mia esplorazione vi si appoggia senza pretendere di possederlo.

Posso così verificare la regola attuale sui campi obbligatori senza duplicarla in ogni studio che riguardi un rapporto di

intervento. Posso anche tornare alle sue fonti e alle decisioni che l'hanno costruita.

Leggere non autorizza a modificare. Se una nota mi sembra incompleta o falsa, il mio *Contesto* può formulare una proposta, ma non corregge mai direttamente il vicino.

Questa prima relazione risponde a un bisogno comune: usare una conoscenza esterna e lasciarla vivere nel luogo responsabile di mantenerla.

Non risponde a tutti i casi.

Assorbire rende la conoscenza autonoma

A volte una semplice consultazione non basta.

Posso avere bisogno di fissare la regola esatta su cui poggia una decisione. Posso volerla riformulare per l'ambito specifico della mia esplorazione. Posso anche avere bisogno che quella conoscenza resti disponibile se il *Contesto* d'origine viene archiviato, spostato o riorganizzato.

In quel caso scelgo di assorbirla.

Assorbire fa entrare una conoscenza nel mio *Contesto*. Posso riprendere una nota così com'è oppure riformularla nel mio lessico. A seconda del bisogno copio anche le fonti che permettono di verificarla, oppure conservo dei riferimenti a esse.

La copia non diventa per questo un'idea senza passato.

Conserva il *Contesto* da cui viene, la data dell'assorbimento e quello che è stato riformulato.

La differenza sta dunque meno nel gesto tecnico che nella responsabilità che ne risulta:

- quando leggo, la conoscenza continua ad appartenere al vicino;
- quando assorbo, una versione autonoma entra nel mio Contesto e divento responsabile del suo uso.

Leggere crea una dipendenza. Assorbire crea una nuova responsabilità.

Il movimento può ripartire nell'altro senso

La mia esplorazione non si limita a consumare quello che 02-intervention sa già. Produce a sua volta osservazioni, ipotesi e a volte una conoscenza duratura.

Studiando la qualità dei rapporti, posso scoprire che una regola di compilazione appartiene in realtà all'intervento stesso. Quella conoscenza non dovrebbe restare chiusa in un Contesto esplorativo che forse finirà archiviato.

Non la scrivo direttamente in 02-intervention. Gliela propongo. Il senso della circolazione è appena cambiato:

```
01-saisie-compte-rendu — legge → 02-intervention
01-saisie-compte-rendu — propone → 02-intervention
                                   |
                                   └─ decide se assorbire o no
```

Chi produce una conoscenza non ne decide il posto definitivo. Il Contesto che la riceve la valuta secondo la propria missione. Può:

- riprendere la nota così com'è;
- riformularla nel proprio lessico;
- conservare una divergenza quando le due letture restano valide;
- rifiutare l'assorbimento quando ciò che è utile all'esplorazione non descrive l'intervento stesso.

Il rifiuto non è un fallimento della circolazione. È quello che impedisce a un risultato locale di trasformarsi silenziosamente in una verità generale.

Quando l'esplorazione diventa una realtà del prodotto

Qualche settimana dopo, la nuova compilazione dei rapporti va in produzione. 01-saisie-compte-rendu ha svolto il suo ruolo: quella che era solo un'ipotesi è diventata una realtà del prodotto.

Non per questo sposto tutto il suo contenuto in 02-intervention.

Passo in rassegna quello che l'esplorazione ha prodotto e separo due insiemi.

Il primo contiene quello che ormai descrive o spiega il prodotto:

- le note che stabiliscono le nuove regole di compilazione;
- il lessico fissato per nominarle;
- le decisioni fondamentali e le ragioni di quelle scelte;
- le contraddizioni che restano aperte dopo la messa in produzione;
- le fonti o i riferimenti necessari a verificare quella conoscenza.

Questi elementi vengono proposti a *02-intervention*. Il Contesto stabile non li accumula alla cieca: arricchisce le note che ha, riformula quello che va riformulato e crea una nota nuova solo quando un'idea è davvero nuova.

Nel nostro esempio, la regola che limita la chiusura a quattro dati passa a *02-intervention* con la decisione che la spiega.

Adesso descrive una realtà del prodotto.

Il secondo insieme racconta la storia del lavoro senza descrivere il prodotto attuale:

- le ipotesi abbandonate;
- i mockup e le alternative scartate;
- le bozze intermedie;
- i compiti propri dell'esplorazione;
- le discussioni che permettono di capire il cammino percorso.

6

Questi elementi restano in *01-saisie-compte-rendu*. Non sono diventati inutili: spiegano che cosa è stato provato, perché certe strade sono state scartate e che cosa non andrà ricominciato senza una ragione nuova.

Una volta finita questa cernita, il Contesto esplorativo viene archiviato per intero. Resta leggibile e può essere ripreso più tardi, ma non pretende più di descrivere il lavoro attivo.

Se l'esplorazione non fosse approdata a nulla, sarebbe stata archiviata ugualmente, con le ragioni del suo abbandono, senza imporre niente a *02-intervention*.

La messa in produzione non fonde dunque due `Contesti`. Fa scattare una revisione: quello che ormai descrive il prodotto passa al `Contesto stabile`; quello che racconta l'esplorazione resta nel `Contesto archiviato`.

Questa cernita protegge entrambe le parti:

- l'esplorazione resta libera di sbagliare;
- il `Contesto stabile` riceve soltanto quello che sceglie di riconoscere come proprio.

Le relazioni fanno parte del quadro

Leggere, assorbire e proporre non devono restare gesti impliciti.

Il quadro stabile del `Contesto` indica:

- quali altri `Contesti` può consultare;
- che cosa quella consultazione autorizza;
- secondo quali regole una conoscenza può essere assorbita;
- come se ne conservano l'origine e le eventuali riformulazioni.

Questa governance non richiede un'infrastruttura pesante. Dà semplicemente un posto esplicito alle relazioni perché la provenienza, le responsabilità e i confini restino visibili.

La rete non si costruisce dunque collegando tutto a tutto. Si costruisce relazione dopo relazione, ogni volta che un lavoro ha davvero bisogno di un vicino.

- Leggere dal vicino evita di duplicare quello che lui sa.
- Assorbire permette di rendere autonomo quello che deve vivere da me.

-
- Proporre permette infine al vicino di decidere se vuole integrare quello che la mia esplorazione ha prodotto.

Questa circolazione sembra lineare sulla carta. Ha però un prezzo: un riferimento può invecchiare senza avvisare, una copia può perdere la propria origine e due Contesti possono conservare letture diverse.

È quello che il capitolo seguente andrà a esaminare.

DA RICORDARE — TRE GESTI, TRE RESPONSABILITÀ

Leggere consulta una conoscenza che resta presso il suo proprietario; assorbire ne crea una versione autonoma, accompagnata dalle sue fonti, nel Contesto che la riceve.

Un'esplorazione propone quello che ha imparato, ma il proprietario del Contesto di destinazione decide che cosa entra da lui. La messa in produzione non fonde le cartelle: il Contesto stabile assorbe quello che ormai descrive il prodotto, mentre l'esplorazione conserva il resto prima di essere archiviata.

Quanto costa la circolazione

Sono passate sei settimane.

Il mio lavoro sui rapporti di intervento è pronto per la revisione, e qualcuno nota una discrepanza. La regola su cui mi appoggio — quella che dice quali campi sono obbligatori alla chiusura — non è più quella del prodotto.

È cambiata. Mentre lavoravo, un'altra esplorazione è arrivata a un esito, e il Contesto dell'intervento ne ha assorbito i risultati. Nessuno me l'ha detto. Non l'ho riletto, perché l'avevo letto sei settimane prima e niente ha segnalato che bisognasse tornarci.

Il capitolo precedente pone una regola che sembra semplice: si fa riferimento nel senso della lettura, si copia nel senso dell'assorbimento. L'avevo presa per una soluzione definitiva. Non elimina il rischio. Lo sposta, ed è meglio sapere dove ricade.

Una dipendenza che non si vede

Quello che è appena successo ha un nome: una dipendenza cieca.

Il legame è vivo, quindi quello che leggo può cambiare senza di me. Non è quello il problema — è spesso esattamente ciò che voglio. Una regola che evolve dal suo proprietario è il sistema che funziona.

Il problema è non sapere che è cambiata.

Quello che manca è semplice da dire: niente monitora quello che leggo. Il mio *Contesto* sa da chi dipende; non sa quando ciò da cui dipende sia cambiato. È metà del meccanismo, ed è proprio la parte costosa.

Una copia mi avrebbe protetto da questo. Il *Contesto* che riceve diventa autonomo: non dipende più né dall'esistenza né dall'evoluzione della sua fonte.

Ma in cambio perde qualcos'altro. Se niente dice da dove viene la copia, la provenienza sparisce — e un anno dopo ho una nota corretta di cui non so più su che cosa poggi. È esattamente la situazione che tutta la tracciabilità di questo libro cerca di impedire.

La conclusione non è dunque scegliere una parte. È registrare che cosa ogni azione deve conservare:

- un **riferimento** richiede di sapere quando la sua fonte cambia;
- una **copia** richiede di dire da dove viene — quale *Contesto*, a quale data, e che cosa è stato riformulato per strada.

EVOLUZIONE POSSIBILE

Un'altra risposta, ancora da venire, sarà tecnologica: i *Contesti* potranno avvisare i propri osservatori quando cambia il loro stato, soprattutto quando la modifica riguarda l'ambito monitorato da un altro *Contesto*. Il *Contesto* è un tema vivo in cui restano molte possibilità da esplorare.

I diritti che la lettura non dà

Quando leggo spesso un *Contesto*, finisce per sembrarmi un po' mio. Più volte ho avuto voglia di correggere direttamente una nota dal vicino — era incompleta, sapevo che cosa metterci, e mi sarebbero bastati trenta secondi.

La regola non cambia per questo: quello che ho da dirgli è una proposta, e la decisione appartiene a chi riceve.

Cancellare senza rompere la tracciabilità

La cancellazione richiede una regola a parte, e l'ho imparata svuotando un contenitore.

Quando una nota cambia *Contesto*, le fonti che l'hanno prodotta restano indietro. Possono ritrovarsi sole, senza più nessuna nota collegata a loro.

Il riflesso è fare pulizia. È corretto solo in parte.

Una fonte già letta e trasformata in note può essere cancellata: quello che portava vive altrove. Una fonte che nessuno ha ancora letto non può essere cancellata.

- Porta un materiale che nessuno ha estratto, e buttarla equivale a perdere quello che non si è nemmeno aperto.

Il criterio non è dunque né l'età del file né lo spazio che occupa.

- Bisogna sapere se la fonte è già stata letta e verificare che cosa la cita ancora.

C'è inoltre una condizione, e non si negozia. Prima di cancellare, verificare che nessuna nota, nessuna voce di glossario, nessuna bozza e nessuna riga del registro citi

ancora quella fonte. Senza questa verifica, la regola rompe la tracciabilità che pretende di proteggere.

Una divergenza non è un errore

Resta il caso in cui due *Contesti* non dicono la stessa cosa dello stesso argomento.

Il riflesso è vederci un'incoerenza, e quindi decidere in fretta per rimettere ordine. Spesso è sbagliato.

Una divergenza può essere una differenza di ambito: ciascuno può avere ragione entro i propri confini. Può essere un'ipotesi ancora aperta che l'uno sta studiando e l'altro ignora. Possono essere due letture che stanno in piedi entrambe, perché guardano l'argomento da due posti diversi.

Merita allora di essere conservata come tale, e spetta alla persona deciderlo. Quello che il sistema deve garantire è che si sappia che esiste.

Una divergenza dimenticata finisce sempre per farsi sentire nel momento peggiore: quando qualcuno si appoggia a una delle due versioni senza sapere che l'altra esiste. È quello che mi è successo con la regola di chiusura, salvo che non si trattava di due letture ma di una sola che era invecchiata.

Mantenere affidabile la rete ha dunque un costo, e preferisco nominarlo piuttosto che scoprirlo in revisione. Collegare due *Contesti* non elimina il lavoro di vigilanza: lo rende possibile, e ne affida la responsabilità alla persona.

È, in fondo, quello che questo libro racconta dall'inizio.

DA RICORDARE — IL PREZZO DELLA RETE

Un riferimento può invecchiare senza avvisare, una copia può perdere la propria provenienza: ogni gesto ha un debito, e bisogna scrivere quale. Leggere non autorizza a scrivere; quello che si ha da dire al vicino è una proposta, mai una correzione. Una fonte già trasformata in note può sparire; una fonte mai letta non può — e niente si cancella senza verificare chi la cita ancora. Due *Contesti* che si contraddicono non hanno per forza torto: la cosa grave è ignorarlo.

Prima di chiudere

Chiusura

Chiusura

Il libro si ferma qui, ma l'argomento non si ferma con lui.

L'epilogo dice a che punto è questo lavoro, che cosa ha soltanto sfiorato e dove ne uscirà il seguito. Le parentele collocano il *Contesto* fra gli approcci che l'hanno preceduto o che lo circondano, perché tu sappia che cosa prende in prestito e che cosa non riprende.

Epilogo

Grazie di essere arrivato fin qui.

Ho cercato di rendere questo libro il più leggibile possibile, e non ci sono riuscito del tutto. Restano passaggi più aridi di quanto avrei voluto, punti in cui la meccanica prende il sopravvento sul racconto. Tu hai tenuto duro lo stesso: è un regalo che non prendo alla leggera.

Questo libro ha solo sfiorato l'argomento

Nonostante il suo spessore, affronta il `Contesto` solo in superficie.

Diverse parti restano leggere sul piano concettuale, e lo so mentre le pubblico. Le note atomiche, per esempio: a seconda di come le realizzo, ottengo risultati molto diseguali. Quello che funziona in un `Contesto` richiede adattamenti appena si passa a un LLM Wiki, dove il materiale non ha né lo stesso volume né lo stesso uso.

Quelle zone meritano di essere rinforzate. Non lo sono state qui perché un libro deve pur finire, non perché l'argomento sia chiuso.

Un lavoro che resterà un progetto a lato

Il `Contesto` continuerà ad andare avanti, ma sempre come side project.

È un'attività portata avanti a margine del mio mestiere, e le ho già dedicato troppe serate. Ci tornerò — ci torno sempre — ma è ora di rallentare e di godersi la vita.

Che cosa vorrei esplorare dopo

La ricerca di precedenti l'ho fatta. Ha riportato a galla concetti molto vicini al mio, a volte in campi lontani dal prodotto e dalla gestione della conoscenza, spesso in altre lingue — il giapponese e il cinese in particolare, i cui lavori restano in gran parte invisibili dal mondo francofono e anglofono. Ho scritto un articolo su quel percorso; uscirà sul blog.

Quello che manca è il seguito: analizzare quei lavori.

I nomi sono individuati, gli articoli ci sono, e non li ho letti. Sono decenni di ricerca condotta da centinaia di persone su questioni che ho affrontato da solo, la sera. Studiarli a fondo rafforzerebbe questo quadro con molta più solidità che continuare a inventarlo.

Il seguito non sarà un volume II

Questo libro è servito a cristallizzare il concetto. Era il suo ruolo: raccogliere ciò che era sparso, renderlo trasmissibile, dargli una forma abbastanza stabile da poter essere discussa. Per un po' di tempo gli sviluppi usciranno su malorean.net, e non in un volume II o in una nuova versione di quest'opera. Un articolo si corregge, si completa, si contraddice. Un libro fissa — e quello che è descritto qui non è ancora abbastanza maturo per essere fissato due volte.

Se vuoi seguire il resto, è lì che si scriverà.

Parentele e vicinanze

Contesto non è comparso in un vuoto intellettuale.

La maggior parte dei meccanismi che lo compongono ha dei parenti, a volte molto antichi, nella gestione della conoscenza, nei metodi di presa di appunti, nell'ingegneria del software, nei sistemi di supporto alle decisioni, nelle scienze cognitive o ancora nella memoria organizzativa.

Alcune di queste influenze erano consapevoli fin dall'inizio.

Altre le ho scoperte solo dopo aver progettato il metodo, durante una ricerca di precedenti condotta a libro quasi finito.

Bisogna dunque distinguere due cose.

Un'**influenza** è un metodo o un'idea che conoscevo e di cui ho consapevolmente ripreso o adattato certi principi.

Una **parentela** o una **vicinanza** indica un sistema anteriore che risolve un problema simile, a volte con meccanismi sorprendentemente affini, senza per questo aver influenzato direttamente Contesto.

Questa distinzione è importante. Due idee possono essere vicine senza che una derivi dall'altra.

Zettelkasten: la conoscenza come rete di unità riutilizzabili

La parentela con lo Zettelkasten è probabilmente la più evidente.

Nel sistema di Niklas Luhmann, una nota non viene semplicemente riposta in una cartella corrispondente al progetto del momento. Entra in un sistema duraturo di note collegate fra loro e può essere riutilizzata in nuovi ragionamenti.

Contesto condivide fortemente questa idea: una conoscenza prodotta durante un lavoro non dovrebbe sparire insieme al suo deliverable.

Le note atomiche del Contesto appartengono chiaramente a questa famiglia.

La differenza principale sta nel loro statuto. Una nota atomica deve conservare la propria provenienza e si inserisce in un'architettura che distingue esplicitamente fonti, contraddizioni, decisioni, stato attuale, sintesi e deliverable. Lo Zettelkasten offre dunque una parentela molto forte per la **costruzione progressiva di un capitale di conoscenze**, ma non per l'insieme del sistema.

PARA: organizzare l'informazione in funzione dell'azione

PARA, di Tiago Forte, fa parte delle influenze consapevoli di Contesto.

PARA organizza l'informazione secondo quattro categorie: Projects, Areas, Resources e Archives. Una delle sue intuizioni importanti è che un'organizzazione dell'informazione deve servire l'azione, non una classificazione astratta del mondo. Contesto conserva qualcosa di quell'intuizione.

Un Contesto non esiste perché esiste un argomento. Esiste perché un essere umano persegue una missione.

Il confine non è dunque: «tutto quello che so sui database».

Potrebbe essere: «scegliere un'architettura dei dati per il prodotto X».

La differenza è che i Contesti stessi diventano unità di lavoro autonome, con il proprio stato, le proprie fonti, le proprie conoscenze, le proprie decisioni e il proprio ciclo di vita.

Domain-Driven Design: il confine prima del contenuto

Il termine stesso di `Contesto` richiama naturalmente il **Bounded Context** del Domain-Driven Design.

La parentela sta soprattutto in un'idea: un'informazione o un concetto ha senso solo all'interno di un certo confine.

La stessa parola può avere un significato diverso in due contesti diversi senza che sia necessario fonderli.

`Contesto` traspone quell'intuizione.

Ma invece di far dipendere il confine da un modello di dominio coerente, lo fa dipendere da una **missione umana**.

Due Contesti possono dunque conservare comprensioni diverse dello stesso argomento perché perseguono obiettivi diversi.

Git: divergenza, autonomia e reintegrazione

Git è un'influenza più direttamente tecnica.

Quando un'esplorazione diventa abbastanza importante, può essere preferibile farla evolvere in un `Contesto` separato invece di disturbare lo spazio stabile.

Questa logica assomiglia a un branch.

L'esplorazione può divergere, produrre le proprie fonti e conoscenze, e poi alcune conclusioni possono essere reintegrate nello spazio d'origine.

Ma, come con Git, integrare non significa per forza fondere tutto.

Quell'intuizione ha influenzato anche le relazioni fra Contesti: autonomia locale, divergenza possibile e integrazione selettiva.

Project Memory: conservare il perché del progetto

I lavori sulla **memoria di progetto** sono una parentela che non avevo individuato quando ho progettato *Contesto*.

Il problema posto è quasi esattamente quello che compare quando un lavoro dura diversi mesi: le organizzazioni conservano abbastanza facilmente i documenti finali, ma molto meno bene il contesto che spiega come e perché il progetto sia arrivato a quel risultato.

Perché questa decisione?

Che cosa abbiamo provato prima?

Quali vincoli c'erano in quel momento?

Che cosa dovrebbe capire una persona nuova prima di continuare?

Contesto risponde alle stesse domande con più memorie distinte: lo stato attuale, la storia, le decisioni, la sintesi e l'handoff.

La vicinanza funzionale è dunque molto forte.

Experience Factory: il deliverable non è il capitale

L'Experience Factory, sviluppata intorno ai lavori di Victor Basili e sperimentata in particolare alla NASA, porta una parentela ancora più importante.

Distingue l'organizzazione che realizza un progetto da quella che trasforma l'esperienza del progetto in conoscenza riutilizzabile per i progetti successivi.

Quest'idea si collega direttamente a una distinzione fondamentale di Contesto:

il deliverable non è il capitale.

Un progetto può produrre un ottimo risultato e non lasciarsi dietro quasi nulla.

Al contrario, la sua esperienza può essere analizzata, trasformata e conservata in modo da aumentare la capacità di risolvere i problemi successivi.

Contesto integra quella capitalizzazione direttamente nello spazio in cui il lavoro viene svolto.

Knowledge-Centered Service: produrre la conoscenza lavorando

Knowledge-Centered Service — KCS — è probabilmente uno dei vicini più interessanti di Contesto.

Il suo principio è che la documentazione non deve essere un'attività separata svolta dopo il lavoro.

Quando un problema viene risolto, la conoscenza esistente viene cercata e riutilizzata. Se è imperfetta, viene migliorata. Se non esiste, viene creata.

Il semplice fatto di lavorare diventa così un meccanismo di creazione e di manutenzione della conoscenza.

È esattamente una delle ambizioni di *Contesto*:

ogni uso del sistema dovrebbe poter aumentare il suo capitale informativo.

KCS è però pensato soprattutto per ambienti di servizio e di supporto, mentre *Contesto* cerca di fornire una struttura generica per molte missioni diverse.

HWMSE: esseri umani, macchine e conoscenze in uno stesso spazio

La parentela più sorprendente scoperta durante la ricerca viene probabilmente dalla Cina.

La **Hall for Workshop of Meta-Synthetic Engineering**, o HWMSE, è sviluppata intorno ai lavori di Qian Xuesen a partire dall'inizio degli anni Novanta.

È pensata per trattare problemi complessi combinando esplicitamente:

- esperti umani;
- sistemi macchina;
- sistemi di conoscenze.

Ancora più interessante: i lavori su HWMSE non parlano soltanto di usare conoscenza esistente. Parlano anche di

creazione di conoscenza e di un sistema di conoscenza proprio del problema studiato, che si arricchisce continuamente durante il suo uso.

La vicinanza con *Contesto* è dunque importante:

- un'attività orientata a un problema;
- una collaborazione essere umano-macchina;
- risorse e modelli mobilitati durante il lavoro;
- conoscenza prodotta dall'attività;
- un capitale che si accumula.

HWMSE non ha però la stessa grammatica operativa di *Contesto*. Non vi si ritrova, in questa forma, la separazione fra fonte, nota atomica, stato, sintesi, contraddizione, handoff e deliverable.

Ma fra i sistemi studiati è probabilmente uno dei vicini architetturali più prossimi.

ScholOnto e SWAN: un'affermazione non è la sua fonte

Un'altra parentela forte compare nei sistemi dedicati al discorso scientifico.

ScholOnto e, più tardi, SWAN distinguono i documenti scientifici dalle **affermazioni prodotte a partire da quei documenti**.

Una risorsa può fare da origine a un'affermazione.

Quell'affermazione conserva la propria provenienza.

Può essere sostenuta, discussa o smentita da altri elementi.

È estremamente vicino al principio che sta sotto le note atomiche del Contesto:

la fonte resta una fonte; la conoscenza costruita a partire da essa è un altro oggetto.

Questi sistemi mostrano anche che un'affermazione registrata in un sistema di conoscenze non va automaticamente considerata vera.

Ha una provenienza e uno statuto dentro un ragionamento. È una parentela particolarmente forte con l'architettura epistemica di Contesto.

Sistemi federati: leggere non è copiare

Infine, certe idee sulle relazioni fra Contesti hanno parenti sorprendentemente antichi nei sistemi di basi di dati federate. Quelle architetture devono permettere a più basi autonome di cooperare senza rinunciare alla propria autonomia.

Un sistema può esporre certe informazioni.

Un altro può consultarle a distanza o decidere di conservarne una copia locale.

E appena viene creata una copia compare il problema classico: che cosa succede quando l'originale cambia?

È esattamente ciò che in Contesto riassume la formula:

riferimento = dipendenza; copia = responsabilità.

I meccanismi informatici esistevano dunque da tempo.

Contesto li traspone in regole di governance della conoscenza fra spazi di lavoro condivisi da esseri umani e IA.

Una sintesi più che un'invenzione dal nulla

Queste vicinanze rendono difficile presentare Contesto come l'invenzione di meccanismi interamente nuovi.

- Non è ciò che mostra la storia.
- La provenienza esisteva.
- Le note atomiche esistevano.
- La memoria dei progetti esisteva.
- La capitalizzazione fra progetti esisteva.
- La creazione di conoscenza durante il lavoro esisteva.
- Gli ambienti che combinano esseri umani, macchine e sistemi di conoscenze esistevano.
- Le asserzioni con fonte e contestabili esistevano.
- Le architetture autonome capaci di condividere o copiare informazione esistevano.

Ciò che sembra caratterizzare Contesto è piuttosto il modo in cui quei meccanismi sono **selezionati, combinati e normalizzati in uno stesso metodo**.

Il metodo dà loro una grammatica comune:

- fonti;
- note atomiche;
- contraddizioni;
- decisioni;
- stato;
- storia;

- sintesi;
- azioni;
- handoff;
- deliverable;
- relazioni fra Contesti.

E soprattutto applica quell'assemblaggio a un problema molto attuale: mantenere una collaborazione duratura fra un essere umano e un'IA senza trasformare ogni conversazione in un episodio isolato e amnesico.

Contesto ha dunque molti parenti.

Ma, come spesso accade, conoscere i propri parenti serve soprattutto a capire meglio che cosa distingue il proprio lavoro.