L’AI scrive il codice.
Ma chi capisce il dominio?

Alberto Acerbis
Software Architect - Trainer

xedotnet.org

un ringraziamento a


Se non stai lavorando sul collo di bottiglia, stai solo creando spreco più velocemente.
Focalizza
Misura
Migliora
Più di questo è solo spreco
Focalizzati sul collo di bottiglia

ILLUSIONE DELLA PRODUTTIVITÀ


IL CODICE È GIUSTO...
MA È SBAGLIATO
L'AI GENERA CODICE PLAUSIBILE.
MA QUANDO IL DOMINIO È AMBIGUO,
RIEMPIE I VUOTI CON IPOTESI.
NESSUN PROBLEMA!
ECCO IL CODICE.
ASPETTA...
HA CAPITO DAVVERO
IL DOMINIO?
DOMINIO REALE
ORDINE
DI VENDITA
?
SPEDIZIONE
PRONTA
CODICE GENERATO DALL'AI
ORDER
(generico)
READY
(generico)
NOTIFY
CUSTOMER
DISALLINEAMENTO
SEMANTICO
RISCHIO: BUG DI DOMINIO
CODICE PLAUSIBILE
public void PrepareOrder(Order order)
{
order.Status = "Ready";
order.DatePrepared = DateTime.Now;
SendNotification(order.Customer);
Save(order);
}
DDD
CLEAN
ARCHITECTURE
AI
I ♥
CODE
VELOCE
SEMBRA
GIUSTO
FUNZIONA!
FUNZIONA.
MA NON È
CORRETTO.

3 AMBIGUITÀ IN INGRESSO → BUG IN USCITA

SE IL NOSTRO LINGUAGGIO È AMBIGUO,
L'AI NON CHIARISCE. DECIDE.

COSA SIGNIFICA ESATTAMENTE "ORDER READY"?

DOMINIO REALE

ORDINE DI VENDITA → ? → SPEDIZIONE PRONTA

INPUT UMANO

..Order is "ready"

AMBIGUO
(significati possibili)

- Pronto per la preparazione?
- Pronto per la spedizione?
- Pronto per la fatturazione?

IA (LLM)

NON CAPISCE.
PREVEDE.

RIEMPIE I VUOTI

USA IPOTESI
STATISTICHE

OUTPUT (CODICE)

```
if (order.Status == Ready)
{
  ShipOrder(order);
}
```

PLAUSIBILE.
MA POTREBBE ESSERE SBAGLIATO.

RISULTATO?

IL SISTEMA FUNZIONA.
MA IL DOMINIO VIENE TRADITO.

DDD
LINGUAGGIO UBIQUO
SPECIFICHE COME COSTITUZIONE

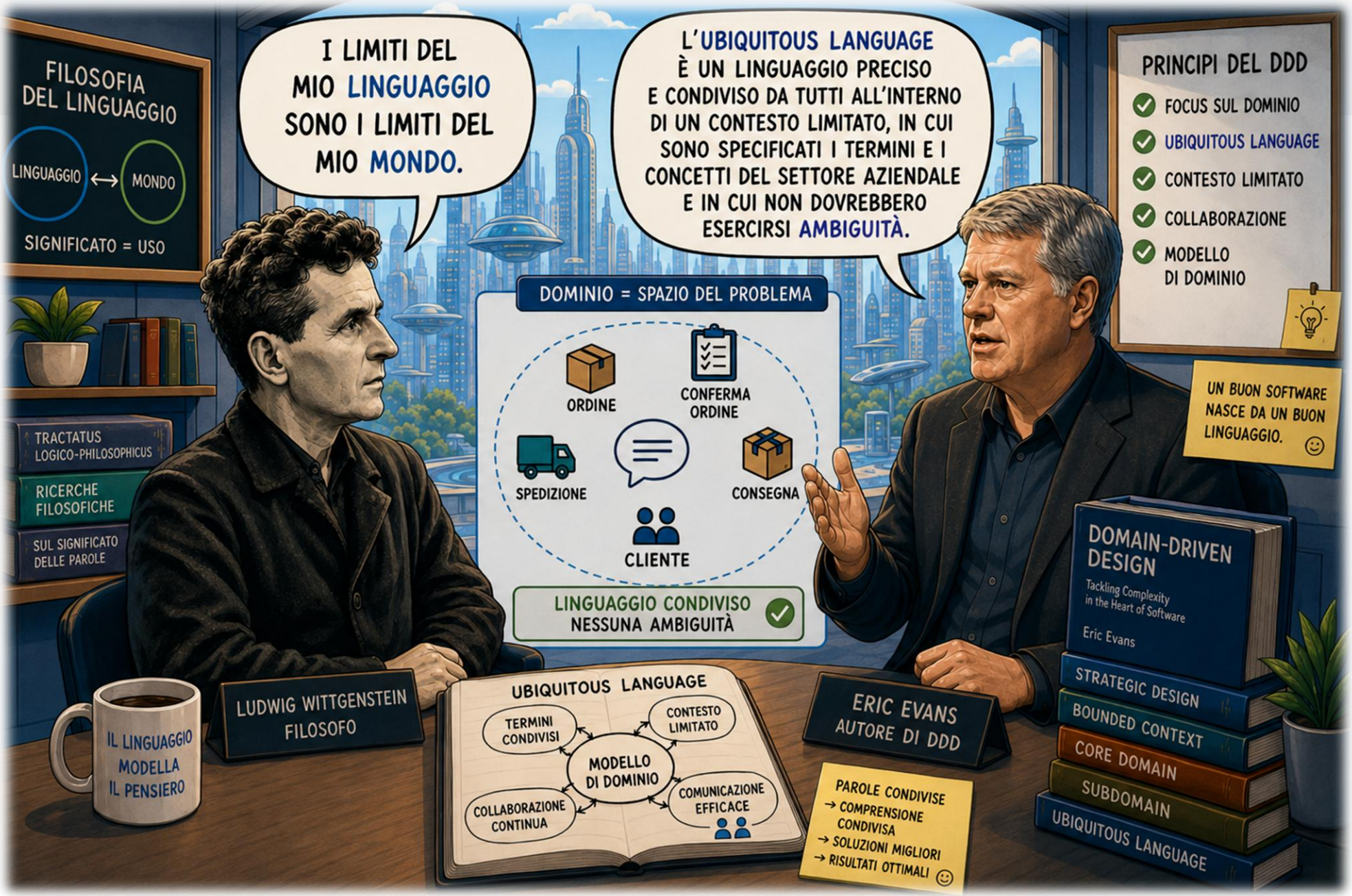
I ♥ DOMINIO

LA VELOCITÀ DELL'AI È STRAORDINARIA.
MA SENZA CHIAREZZA,
È SOLO VELOCITÀ NELLA DIREZIONE SBAGLIATA.

Prima di decidere...

- Perché stiamo facendo questa scelta?
- Quale problema stiamo realmente risolvendo?
- Quale trade-off nascosto stiamo accettando?

Le grandi architetture iniziano con grandi domande.


FILOSOFIA DEL LINGUAGGIO
LINGUAGGIO ↔ MONDO
SIGNIFICATO = USO
TRACTATUS LOGICO-PHILOSOPHICUS
RICERCHE FILOSOFICHE
SUL SIGNIFICATO DELLE PAROLE
I LIMITI DEL MIO LINGUAGGIO SONO I LIMITI DEL MIO MONDO.
L'UBIQUITOUS LANGUAGE È UN LINGUAGGIO PRECISO E CONDIVISO DA TUTTI ALL'INTERNO DI UN CONTESTO LIMITATO, IN CUI SONO SPECIFICATI I TERMINI E I CONCETTI DEL SETTORE AZIENDALE E IN CUI NON DOVREBBERO ESERCIRSI AMBIGUITÀ.
PRINCIPI DEL DDD
FOCUS SUL DOMINIO
UBIQUITOUS LANGUAGE
CONTESTO LIMITATO
COLLABORAZIONE
MODELLO DI DOMINIO
DOMINIO = SPAZIO DEL PROBLEMA
ORDINE
CONFERMA ORDINE
SPEDIZIONE
CONSEGNA
CLIENTE
LINGUAGGIO CONDIVISO
NESSUNA AMBIGUITÀ
UN BUON SOFTWARE NASCE DA UN BUON LINGUAGGIO.
IL LINGUAGGIO MODELLA IL PENSIERO
LUDWIG WITTGENSTEIN FILOSOFO
UBIQUITOUS LANGUAGE
TERMINI CONDIVISI
CONTESTO LIMITATO
MODELLO DI DOMINIO
COMUNICAZIONE EFFICACE
COLLABORAZIONE CONTINUA
ERIC EVANS AUTORE DI DDD
PAROLE CONDIVISE
→ COMPRENSIONE CONDIVISA
→ SOLUZIONI MIGLIORI
→ RISULTATI OTTIMALI
DOMAIN-DRIVEN DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
STRATEGIC DESIGN
BOUNDED CONTEXT
CORE DOMAIN
SUBDOMAIN
UBIQUITOUS LANGUAGE

BOUNDED CONTEXT

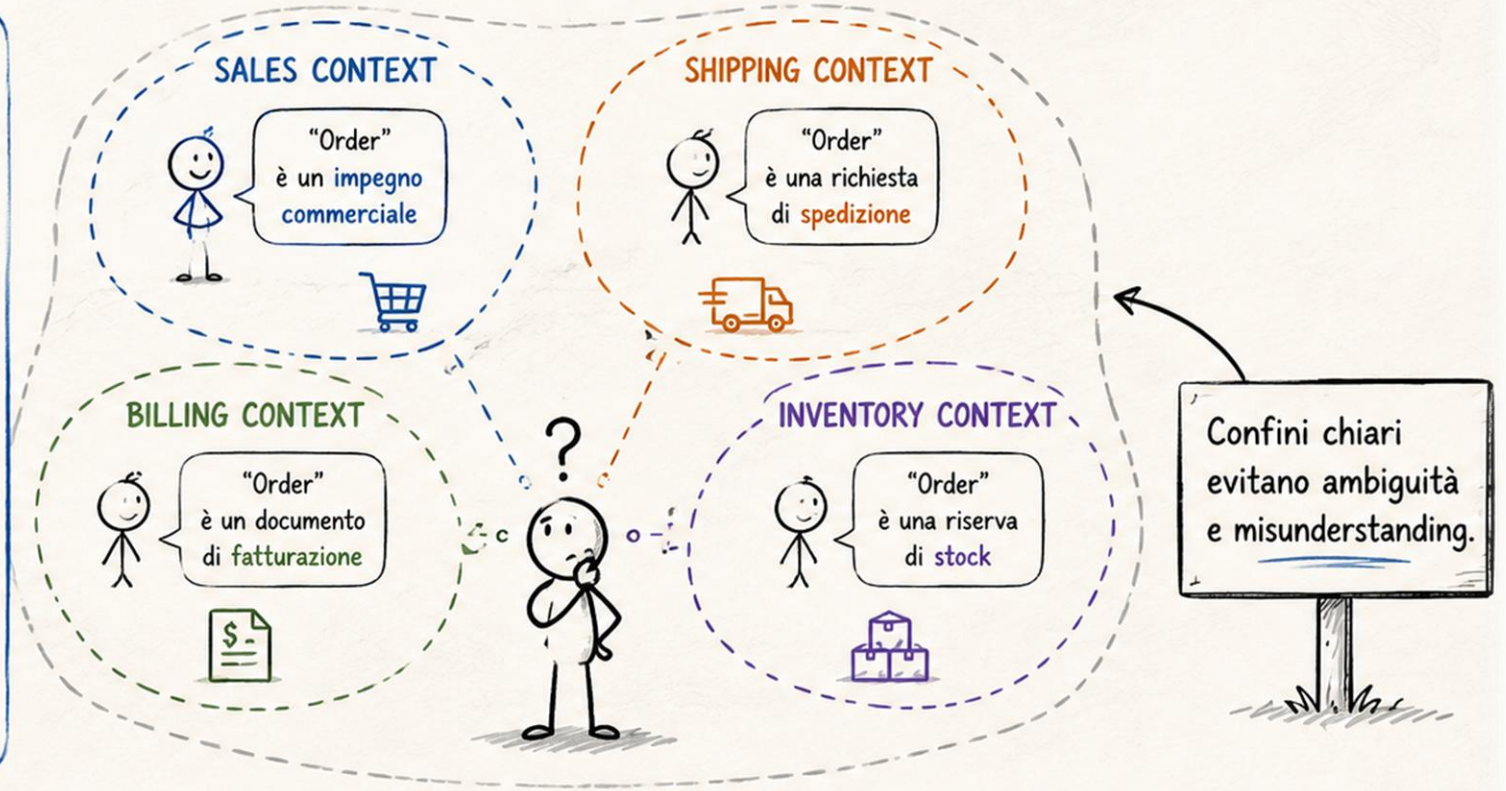
Un confine semantico che dà significato alle parole.

CONTEXT
L'insieme delle circostanze in cui una parola, o un'affermazione, si inserisce, che ne determina il significato.

BOUNDED CONTEXT
È un confine semantico.

SIGNIFICATO SPECIFICO
All'interno di questo contesto, ogni elemento ha un significato specifico.

SCOPO CHIARO
Ogni contesto risolve problemi specifici.



Stesso termine, significati diversi.
Contesti diversi, problemi diversi.

Bounded Context
è chiarezza.
È collaborazione.
È architettura.


FILOSOFIA
Il senso non è dato.
Si produce nei rapporti,
nei flussi, nei contesti.
— Gilles Deleuze
SIGNIFICATO
NEL
CONTESTO
DDD
DOMAIN-DRIVEN DESIGN
SALES
CONTEXT
INVENTORY
CONTEXT
SHIPPING
CONTEXT
SHARED KERNEL
AI / LLM
PROMPT
Contesto definito
Istruzioni chiare
Obiettivo specifico
...
LLM

STESSO PRINCIPIO, DUE MONDI

IL SIGNIFICATO ESISTE SOLO DENTRO UN CONTESTO LIMITATO

DDD
DOMAIN-DRIVEN DESIGN

LIMITO IL CONTESTO PER ELIMINARE AMBIGUITÀ
Riduce le interpretazioni possibili.

DEFINISCO UN LINGUAGGIO CONDIVISO
Tutti parlano la stessa lingua.

BOUNDED CONTEXT
Confini chiari per un sotto-dominio specifico.

MODELLO ESPLICITO
Concetti e relazioni definiti in modo esplicito.

LLM
LARGE LANGUAGE MODEL

LIMITO IL CONTESTO PER RIDURRE ALLUCINAZIONI
Meno ambiguità, meno invenzioni.

FORNISCO UN PROMPT PRECISO
Istruzioni chiare, obiettivo definito.

CONTEXT WINDOW / PROMPT SCOPE
Finestra di contesto limitata e mirata.

PATTERN STATISTICO
Il modello predice la prossima parola, non conosce la verità.

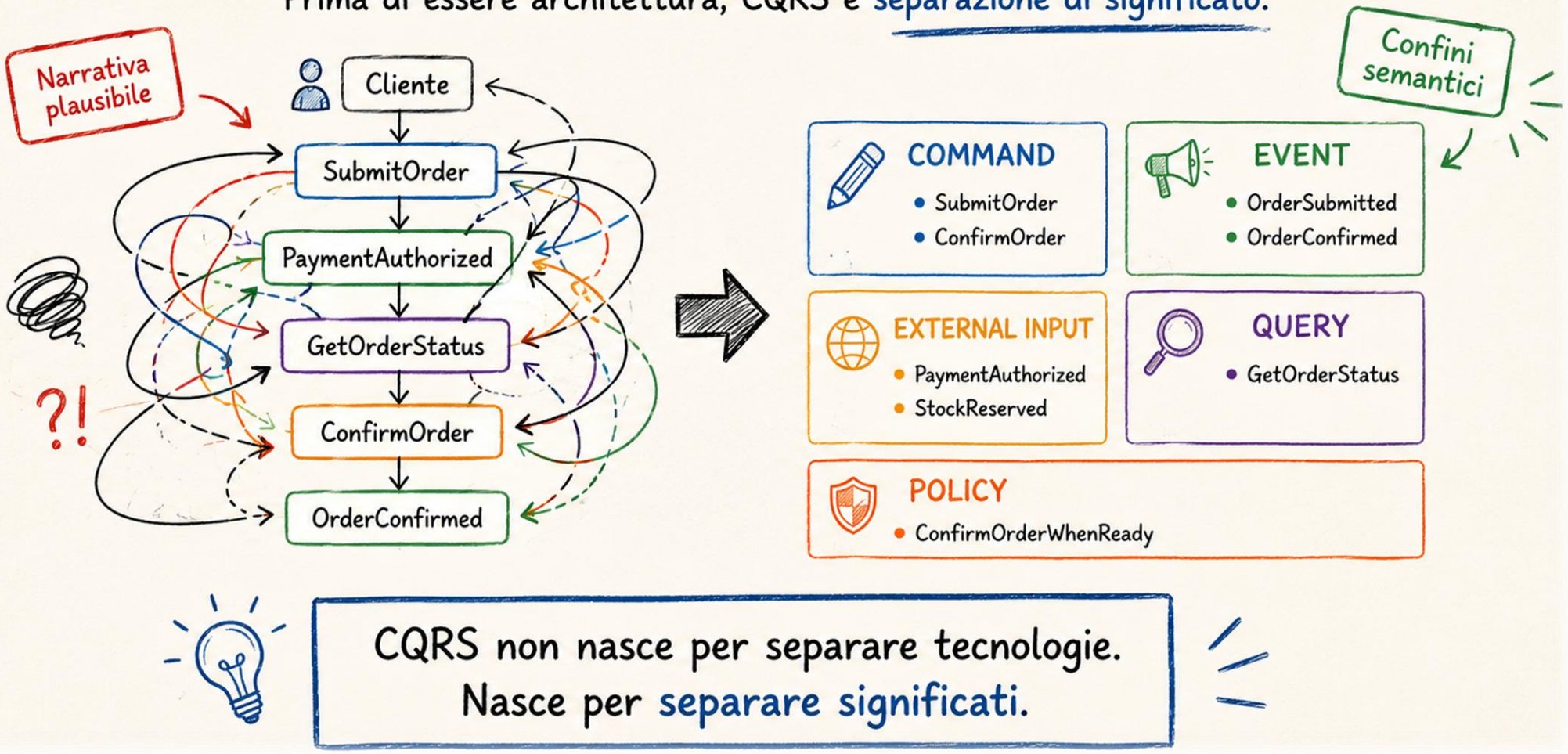
STESSO PRINCIPIO

LIMITARE IL CONTESTO PER DARE SIGNIFICATO

IDEA CHIAVE
Senza contesto limitato non esiste significato. Solo interpretazioni plausibili.

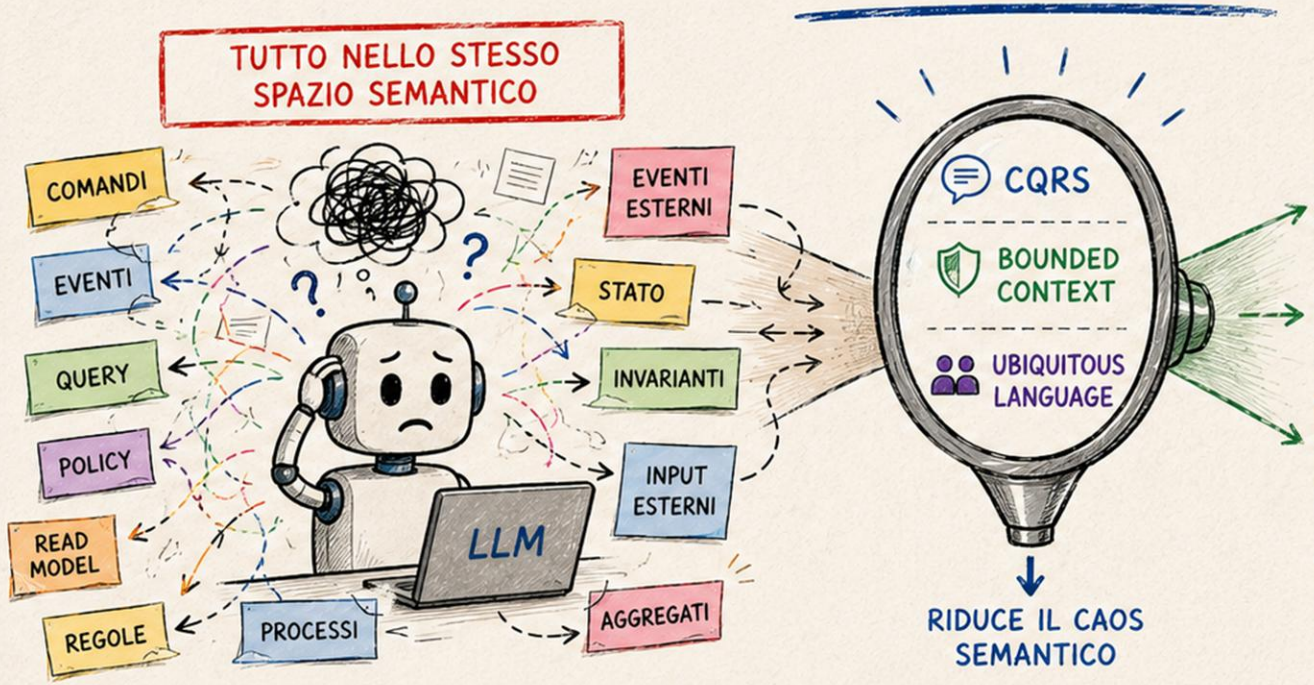
CQRS non separa il codice

Prima di essere architettura, CQRS è separazione di significato.



CQRS non nasce per separare tecnologie.
Nasce per separare significati.

RIDURRE IL CONTESTO
SIGNIFICA RIDURRE L'AMBIGUITÀ



RIDUCE IL CAOS SEMANTICO

CONTESTO LIMITATO = SIGNIFICATO STABILE

ORDERS
- SubmitOrder
- OrderSubmitted
- OrderConfirmed

Read Model

PAYMENTS
- AuthorizePayment
- PaymentAuthorized
- RefundIssued

Read Model

STOCK
- ReserveStock
- StockReserved
- StockReleased

Read Model

SHIPPING
- CreateShipment
- ShipmentCreated
- ShipmentDelivered

Read Model

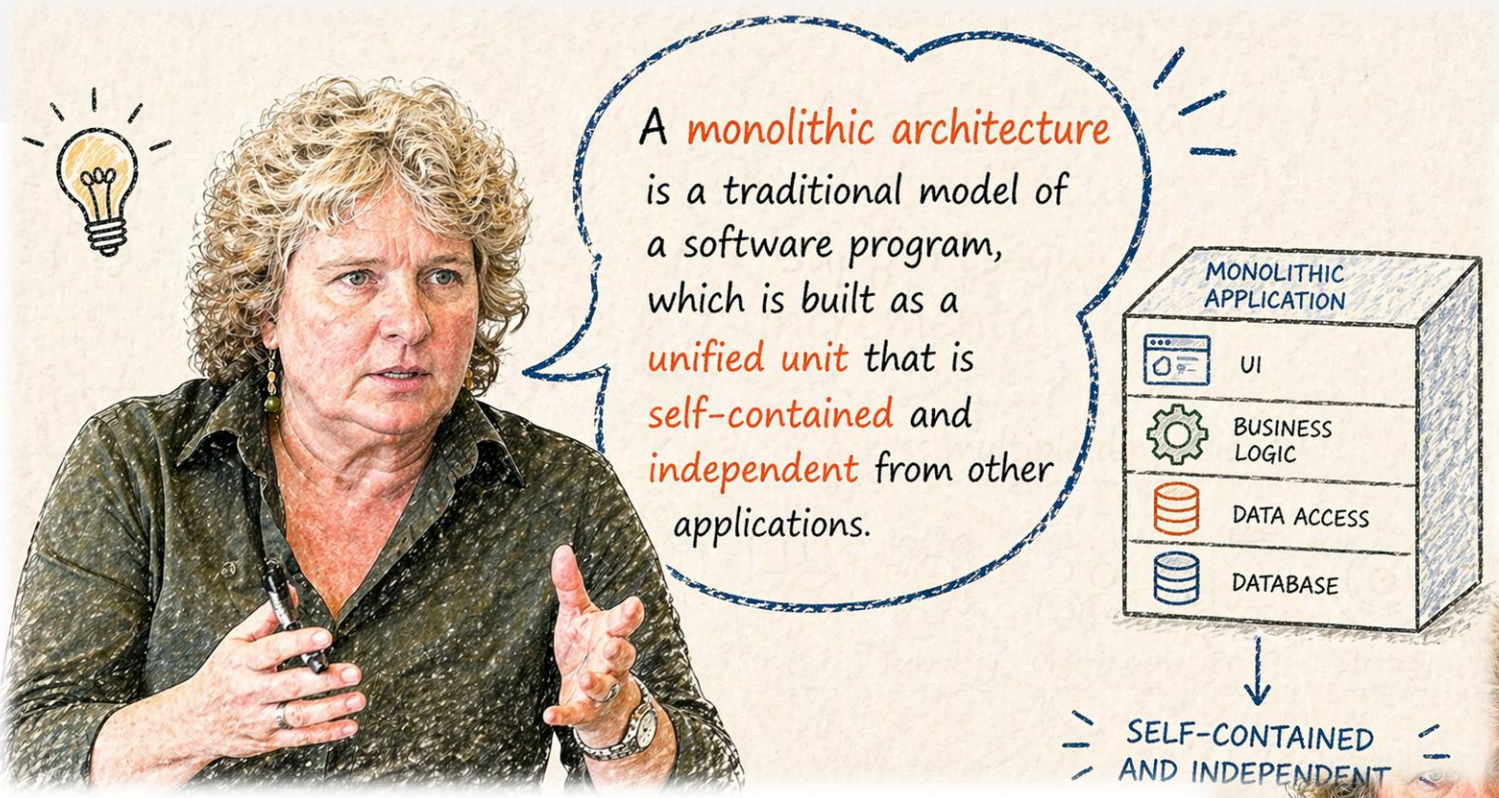
Un LLM non comprende il dominio.
Comprende pattern dentro un contesto.

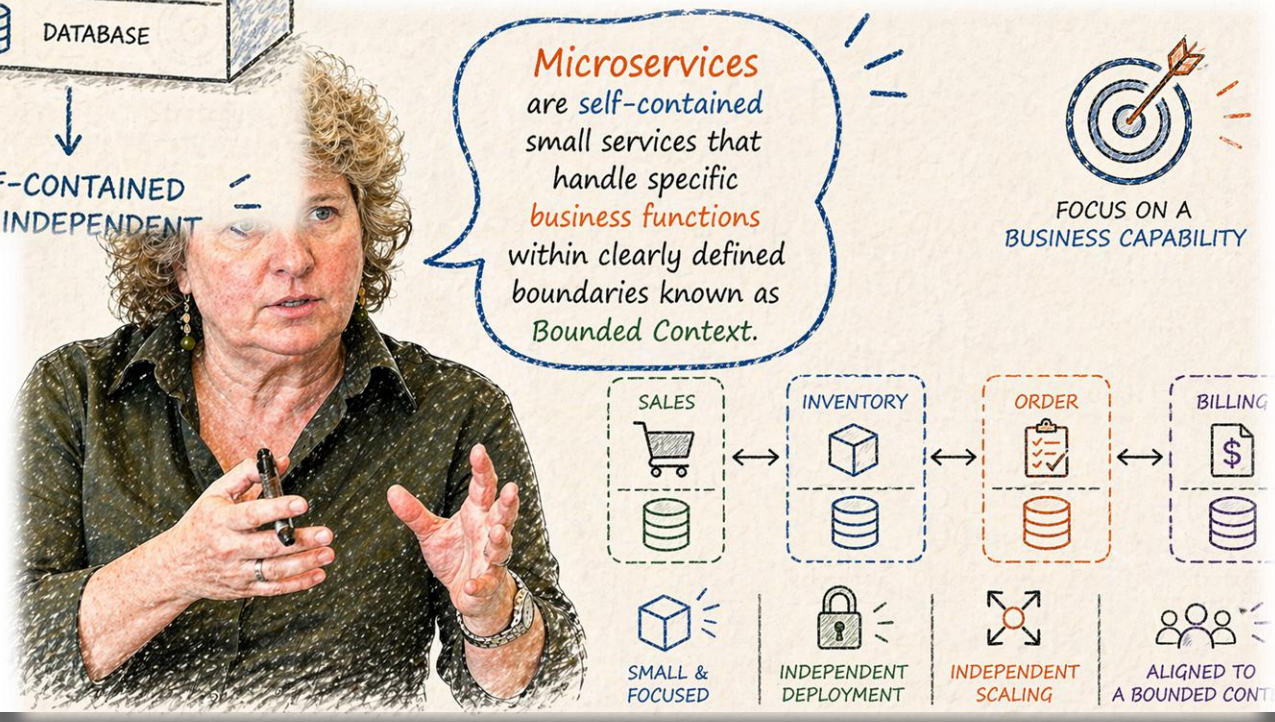
Più il contesto è ampio,
più il significato diventa probabilistico.

- CONTESTO RIDOTTO
- MENO AMBIGUITÀ
- MENO DRIFT SEMANTICO
- CODICE PIÙ STABILE
- MANUTENZIONE PIÙ SEMPLICE


I patterns non sono soluzioni. Sono strumenti per risolvere specifici problemi
Rebecca Wirfs Brock
YOW!
ENGINEERING

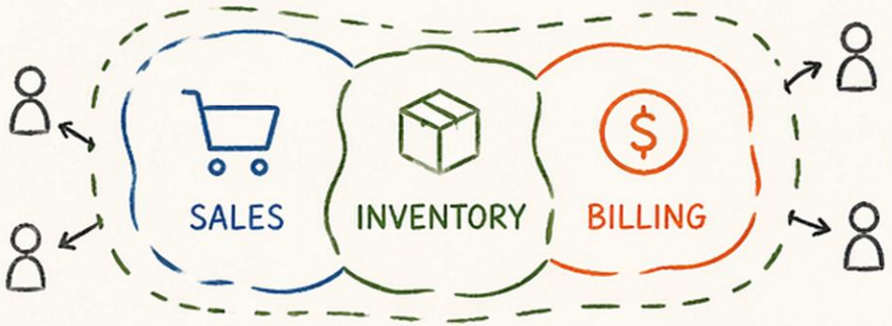

Una simulazione casuale di stressori genera architetture in grado di resistere a fattori imprevedibili
Barry O'Reilly


A monolithic architecture is a traditional model of a software program, which is built as a unified unit that is self-contained and independent from other applications.
MONOLITHIC APPLICATION
UI
BUSINESS LOGIC
DATA ACCESS
DATABASE
SELF-CONTAINED AND INDEPENDENT


Microservices are self-contained small services that handle specific business functions within clearly defined boundaries known as Bounded Context.
FOCUS ON A BUSINESS CAPABILITY
SALES
INVENTORY
ORDER
BILLING
SMALL & FOCUSED
INDEPENDENT DEPLOYMENT
INDEPENDENT SCALING
ALIGNED TO A BOUNDED CONT

BOUNDED CONTEXTS

Built around the
MODEL'S PURPOSE.



Ubiquitous Language,
business rules, model consistency.

≠

MICROSERVICES

Built around
DEPLOYMENT BOUNDARIES.

Technology, scaling,
team & infrastructure constraints.

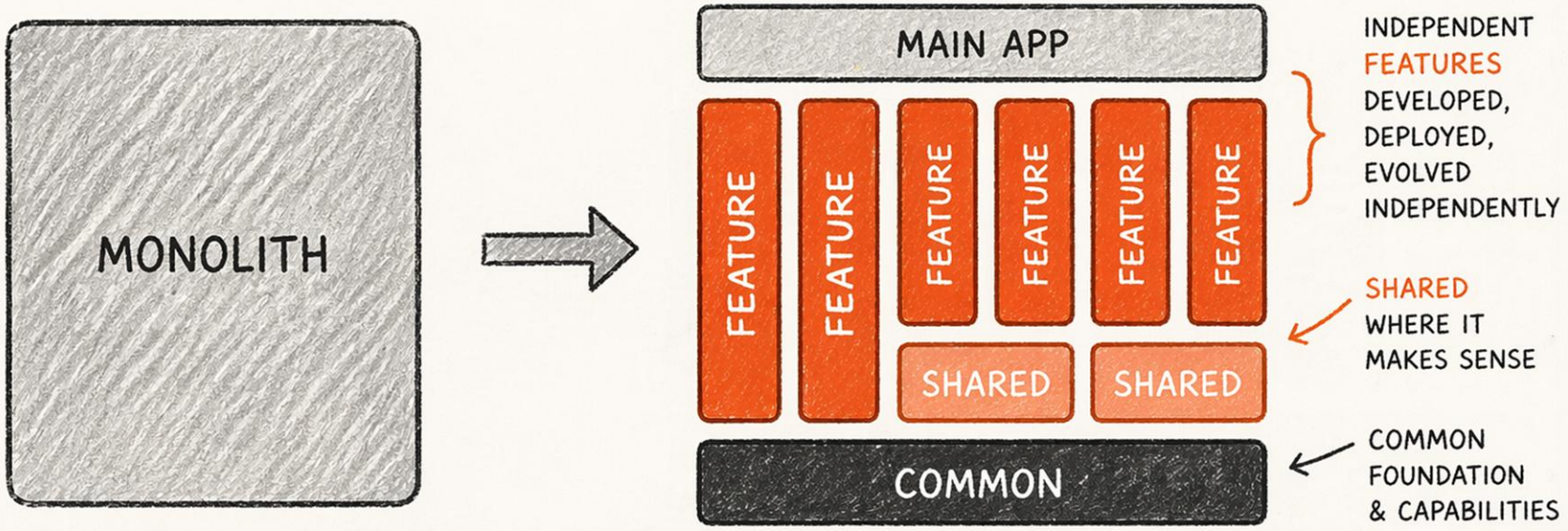
THE DRIVING FORCES ARE NOT THE SAME.

KEEP THE BOUNDARY
LOGICALLY CLEAN:
DEFINED BY THE DOMAIN.

ENFORCE PHYSICAL SEPARATION
WHEN IT'S WORTH IT:
COMPLEXITY HAS A COST.


Come è possibile
pianificare a lungo termine
in un contesto
in continua evoluzione?

MODULAR ARCHITECTURE AS COMPOSABLE ARCHITECTURE



INDEPENDENCE
TEAMS CAN WORK AUTONOMOUSLY

FASTER DELIVERY
RELEASE WHAT CHANGES

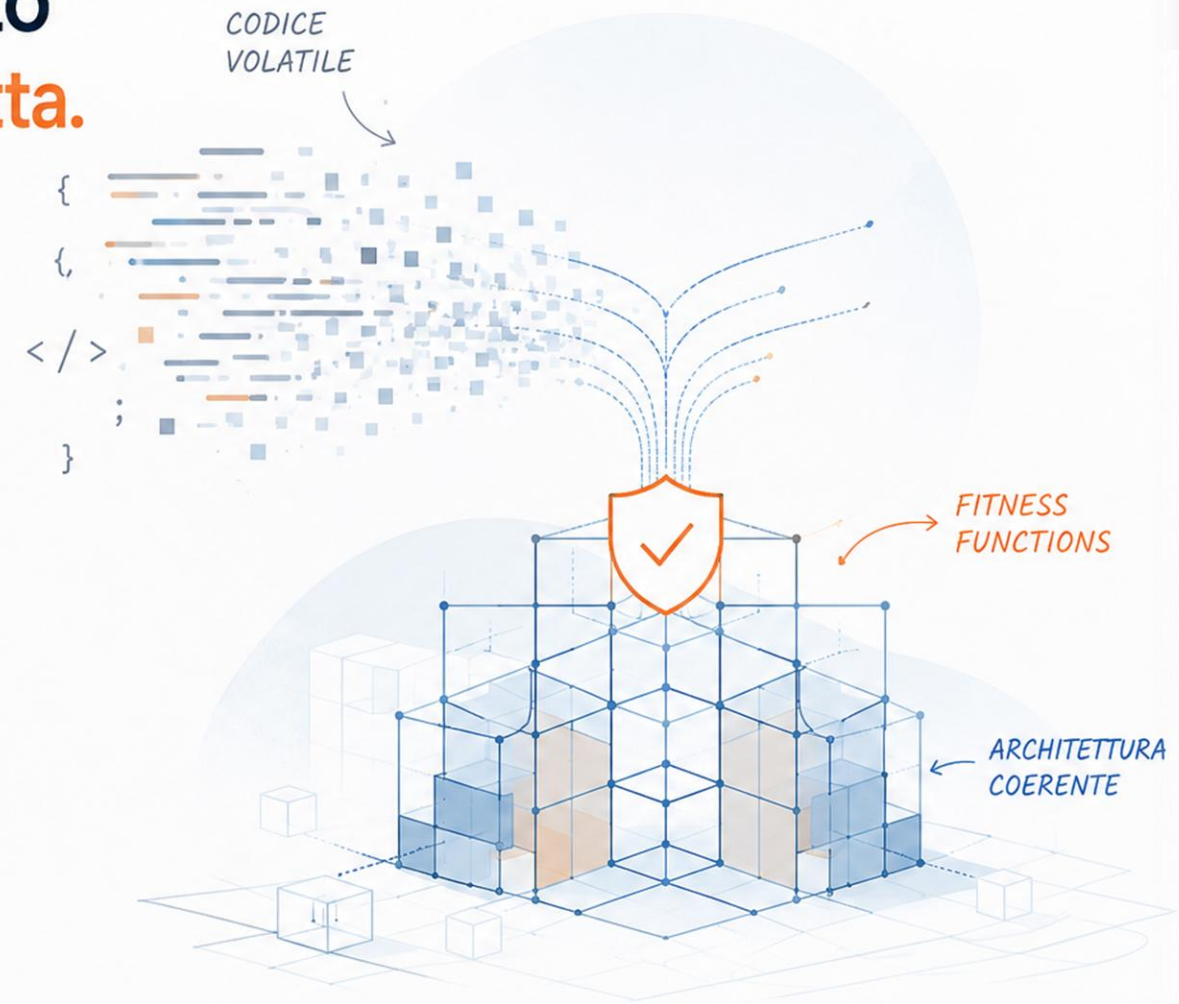
SCALABILITY
SCALE FEATURES, NOT THE WHOLE APP

RESILIENCE
FAILURE ISOLATED BY MODULE

EVOLVABILITY
ADAPT AND IMPROVE CONTINUOUSLY

Dal codice generato all'architettura protetta.

Gli LLM scrivono il codice.
Le fitness functions preservano l'architettura.



Il codice è volatile
gli LLM cambiano, il codice pure.

I vincoli sono stabili
le fitness functions non negoziano.

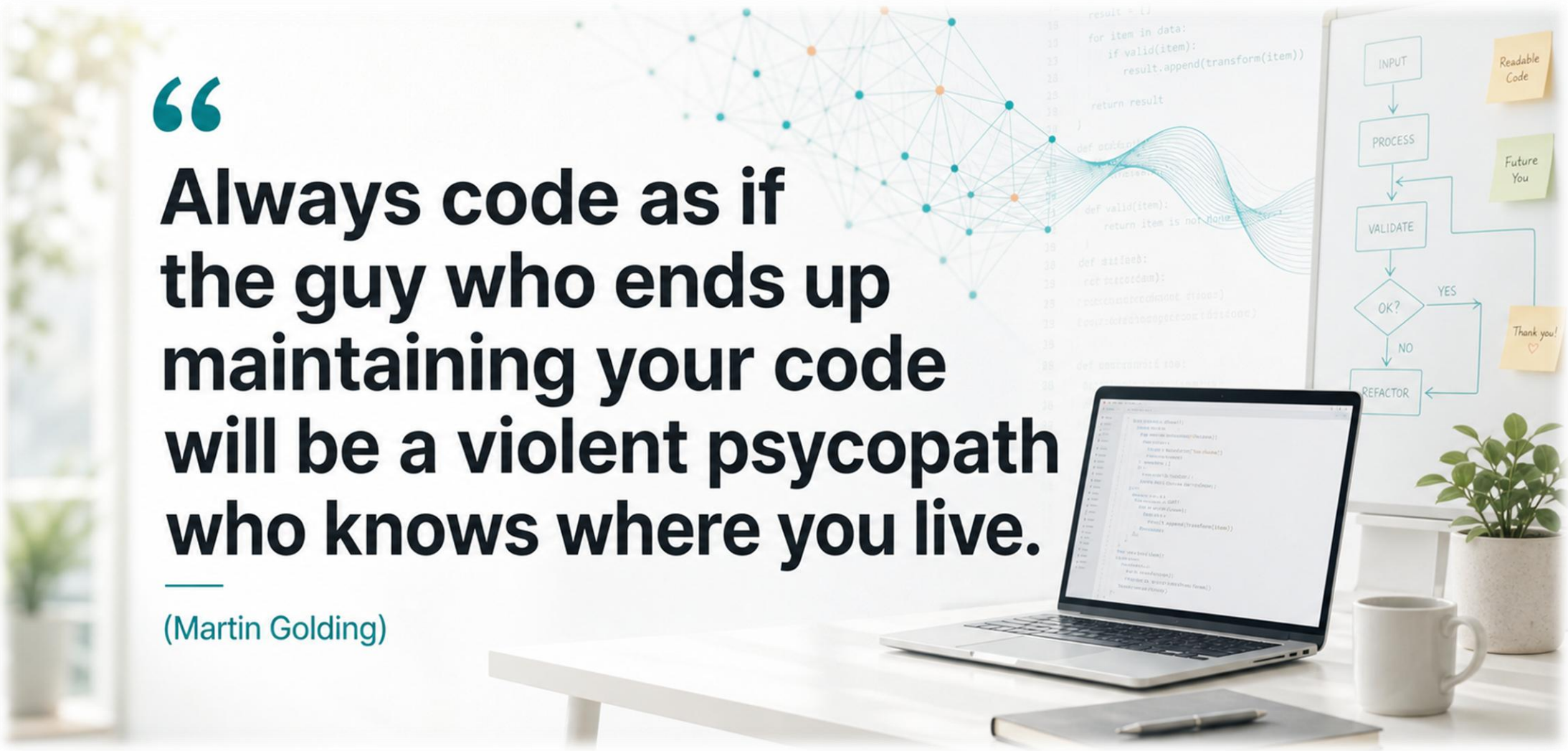
L'architettura resta coerente
anche quando il codice evolve.

I vincoli sono la nostra COSTITUZIONE ARCHITETTURALE


Demo Time!
PRACTICAL TIME

“Developers are drawn to complexity, like moths to a flame, often with the same outcome”

- Neal Ford


Always code as if
the guy who ends up
maintaining your code
will be a violent psycopath
who knows where you live.
(Martin Golding)
INPUT
PROCESS
VALIDATE
OK?
YES
NO
REFACTOR
Readable Code
Future You
Thank you!

Grazie!!!

Alberto Acerbis

Software Architect - Trainer

alberto.acerbis@intre.it

https://github.com/BrewUp/BrewUpErp

Microsoft® MVP Most Valuable Professional

XE Development user group