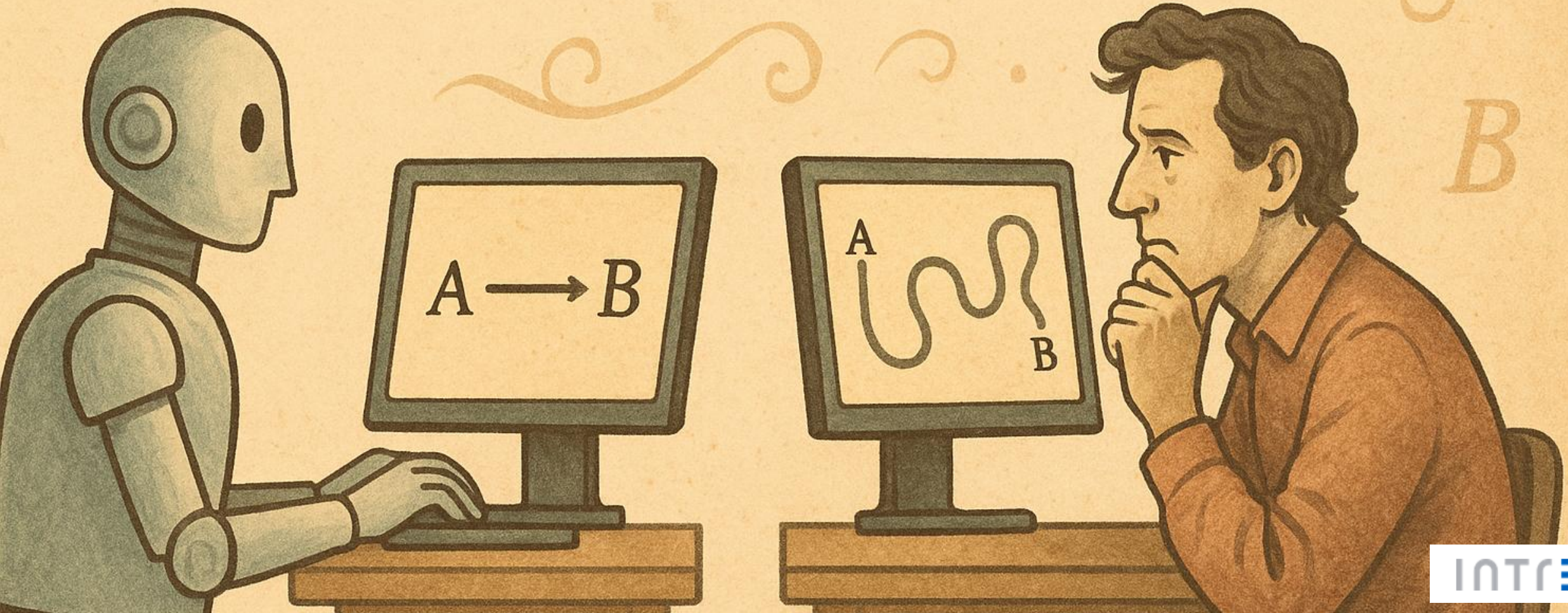
Panta Rei

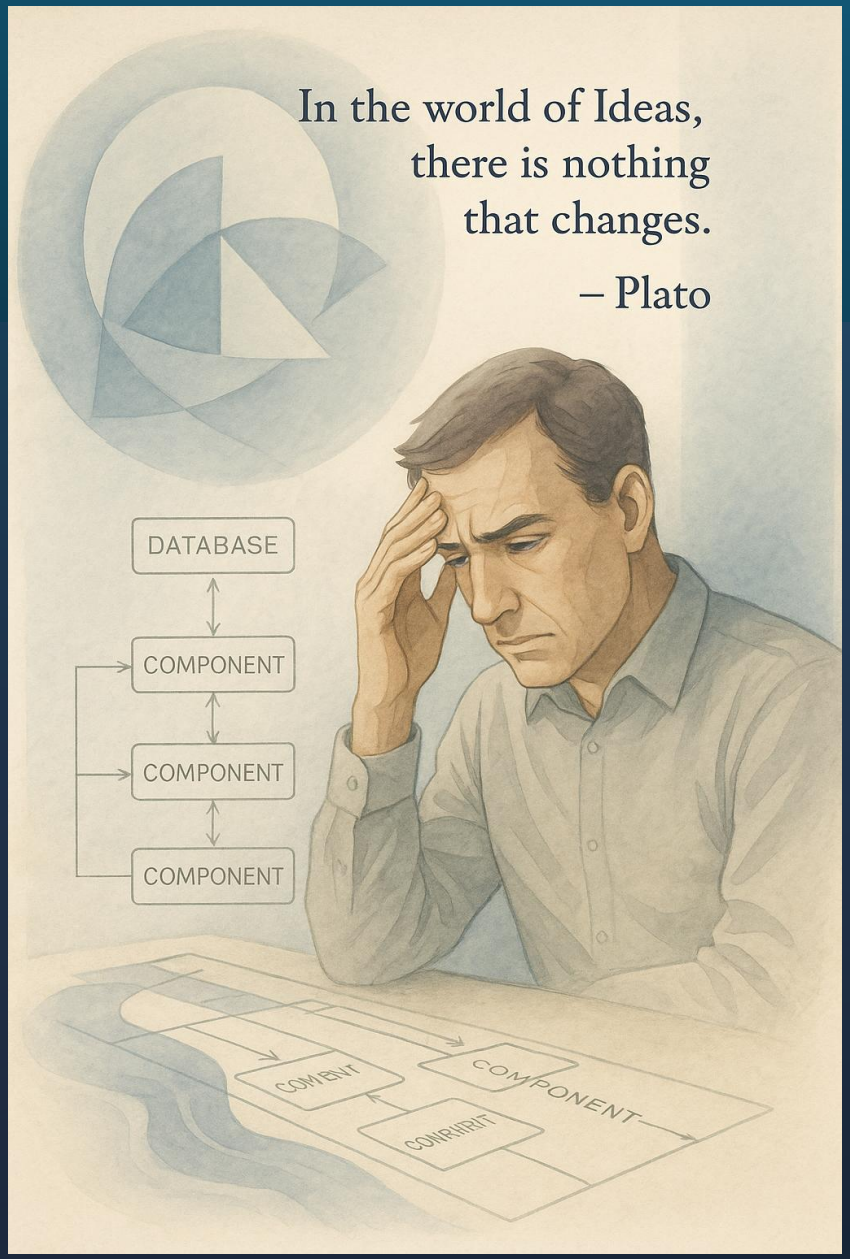
- **Architetture pronte al cambiamento**

Alberto Acerbis

INTR3

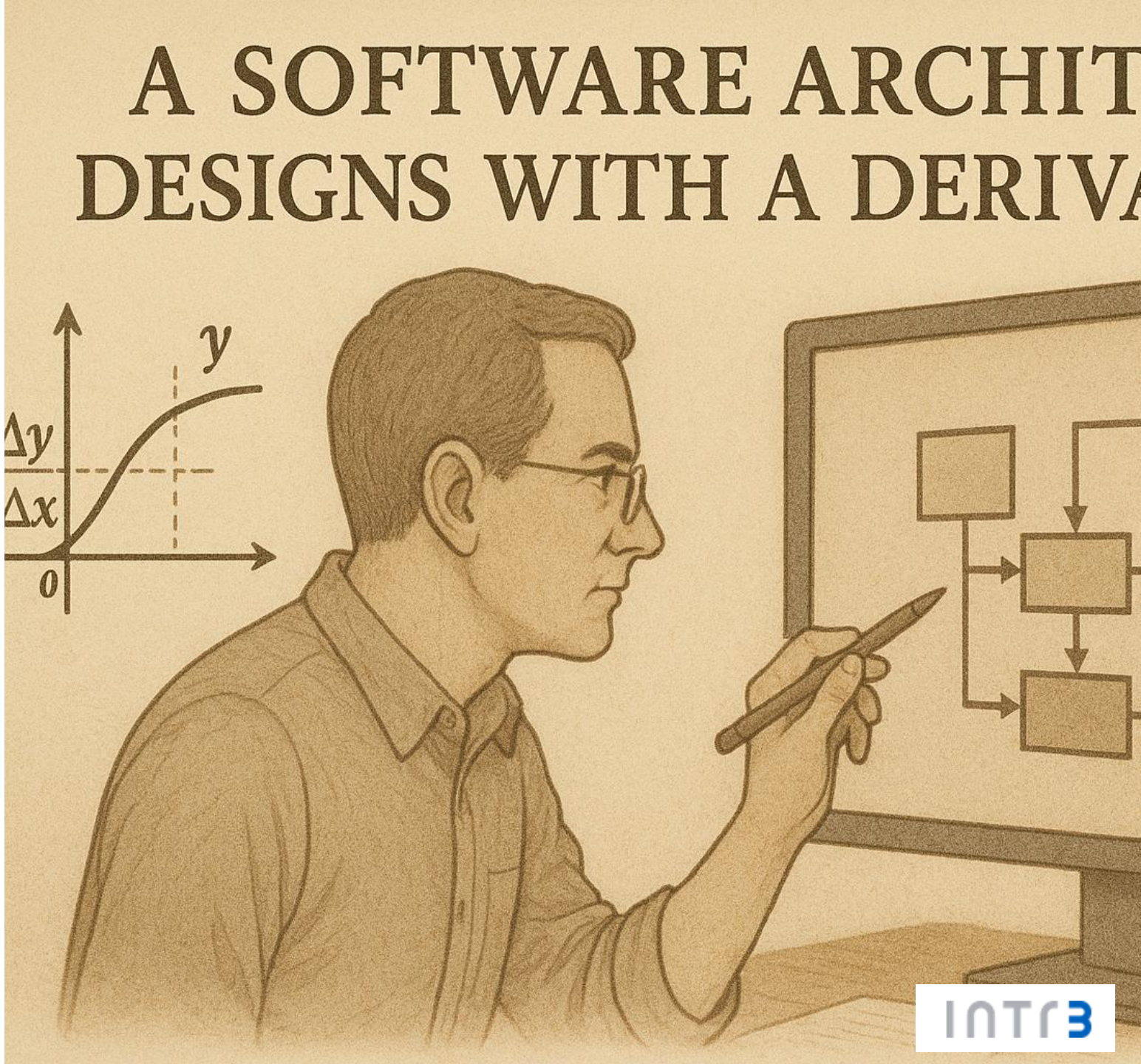
AI follows the path.
Humans design the landscape.




In the world of Ideas,
there is nothing
that changes.
– Plato
DATABASE
COMPONENT
COMPONENT
COMPONENT
COMPONENT
COMPONENT


„Nessun uomo attraverssa due volte lo stesso fiume
– Eraclito
API
Load Balancer
Web Server
Service
API
Storage
Database
Software
Database
Service
Cache

Architects Live in the First Derivative



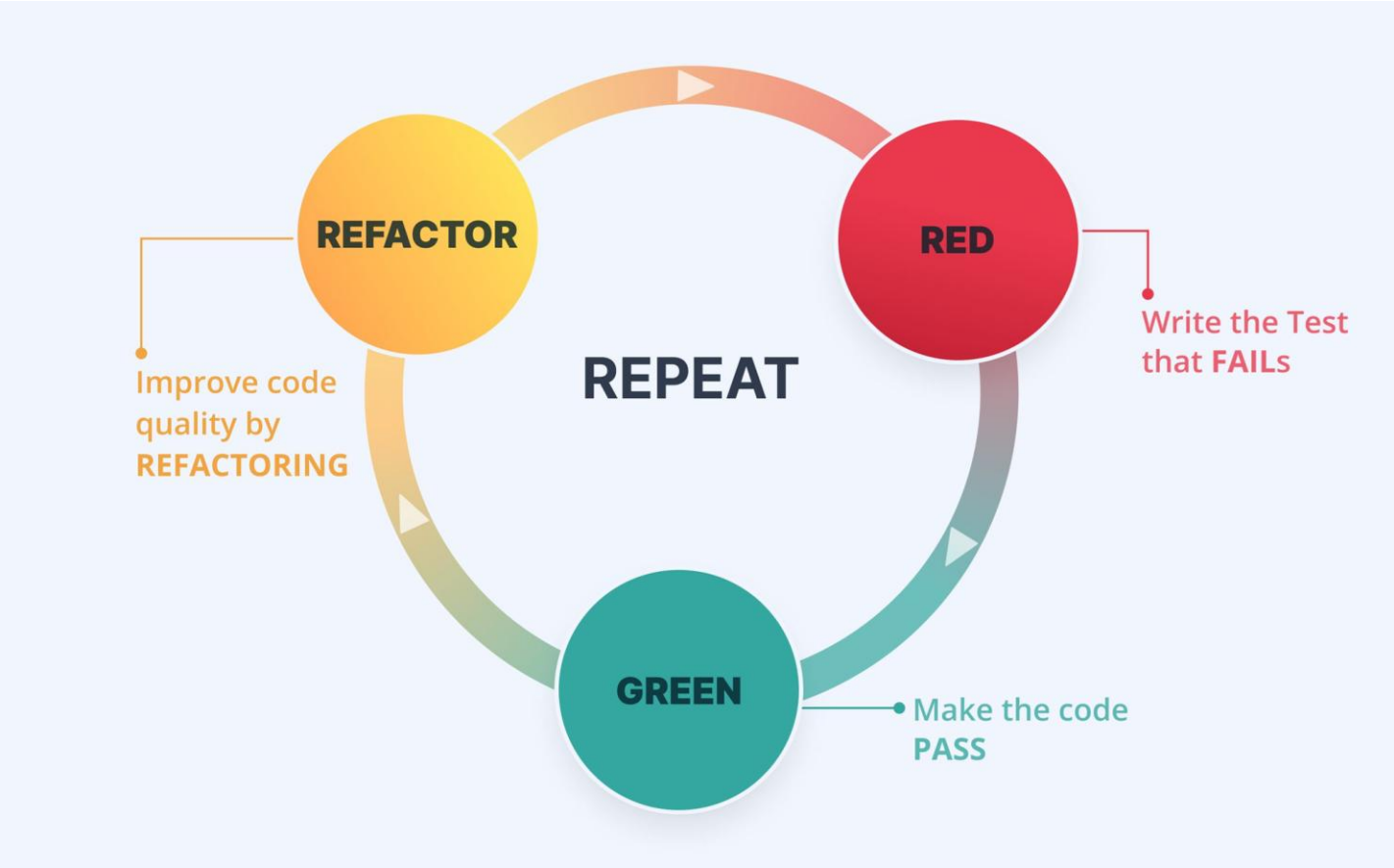
- La derivata di una funzione rappresenta la velocità con cui la funzione varia.
- Ci dice, in ogni istante, come la funzione si muove. Non cos'è, ma dove va!
- E' la lente per osservare il divenire

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Fattore Primario che influenza l’architettura

Il tasso di cambiamento

Refactoring : derivata rispetto alla conoscenza del dominio.

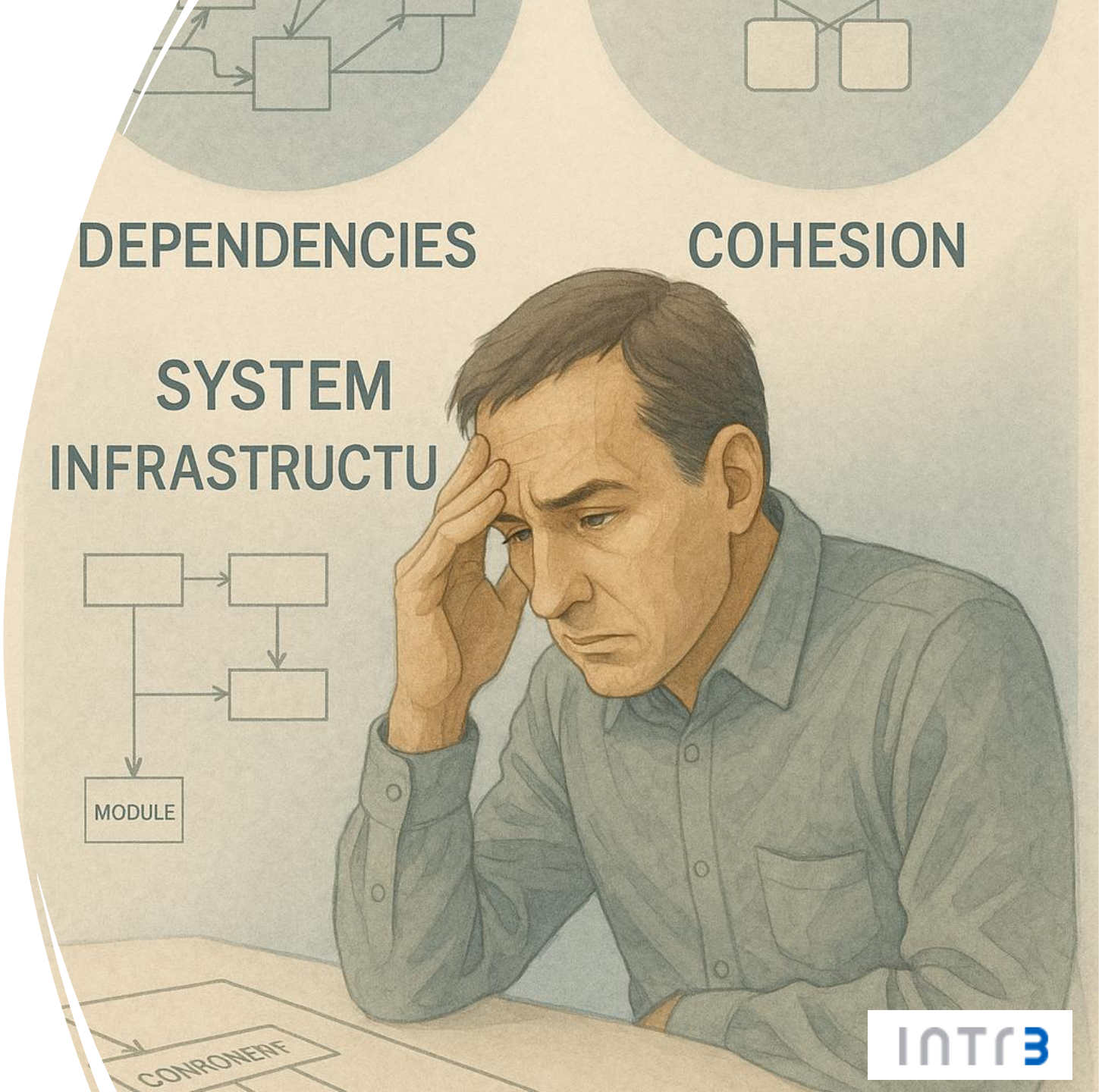


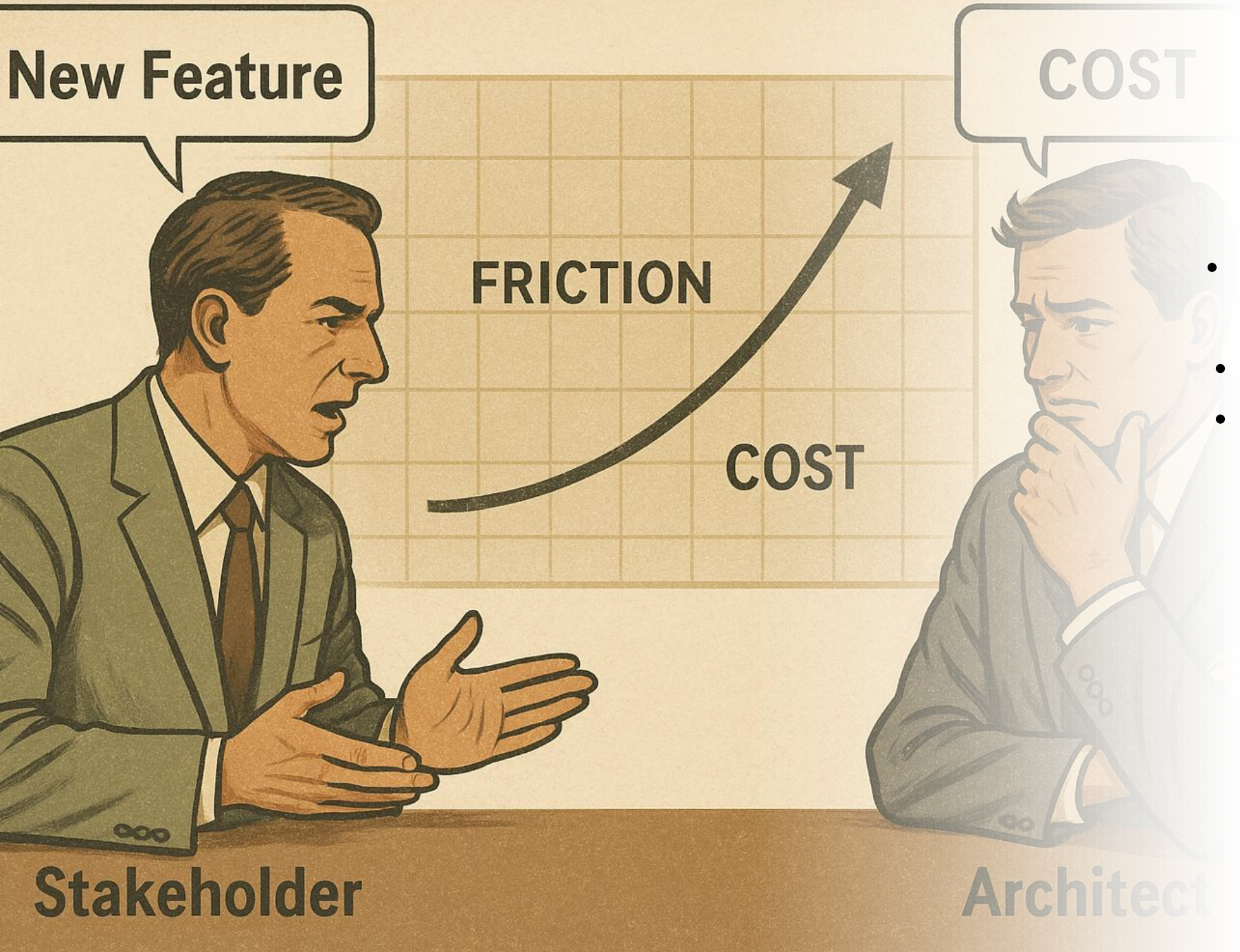
Scalabilità :
derivata rispetto al carico.

Migrazione tecnologica :
derivata rispetto al contesto.

Quali sono gli ostacoli da affrontare?

- Le dipendenze ci obbligano a modificare molti punti, anche a fronte di piccole modifiche
- I sistemi con meno dipendenze, (es. modulari), assorbono un tasso di cambiamento più elevato.

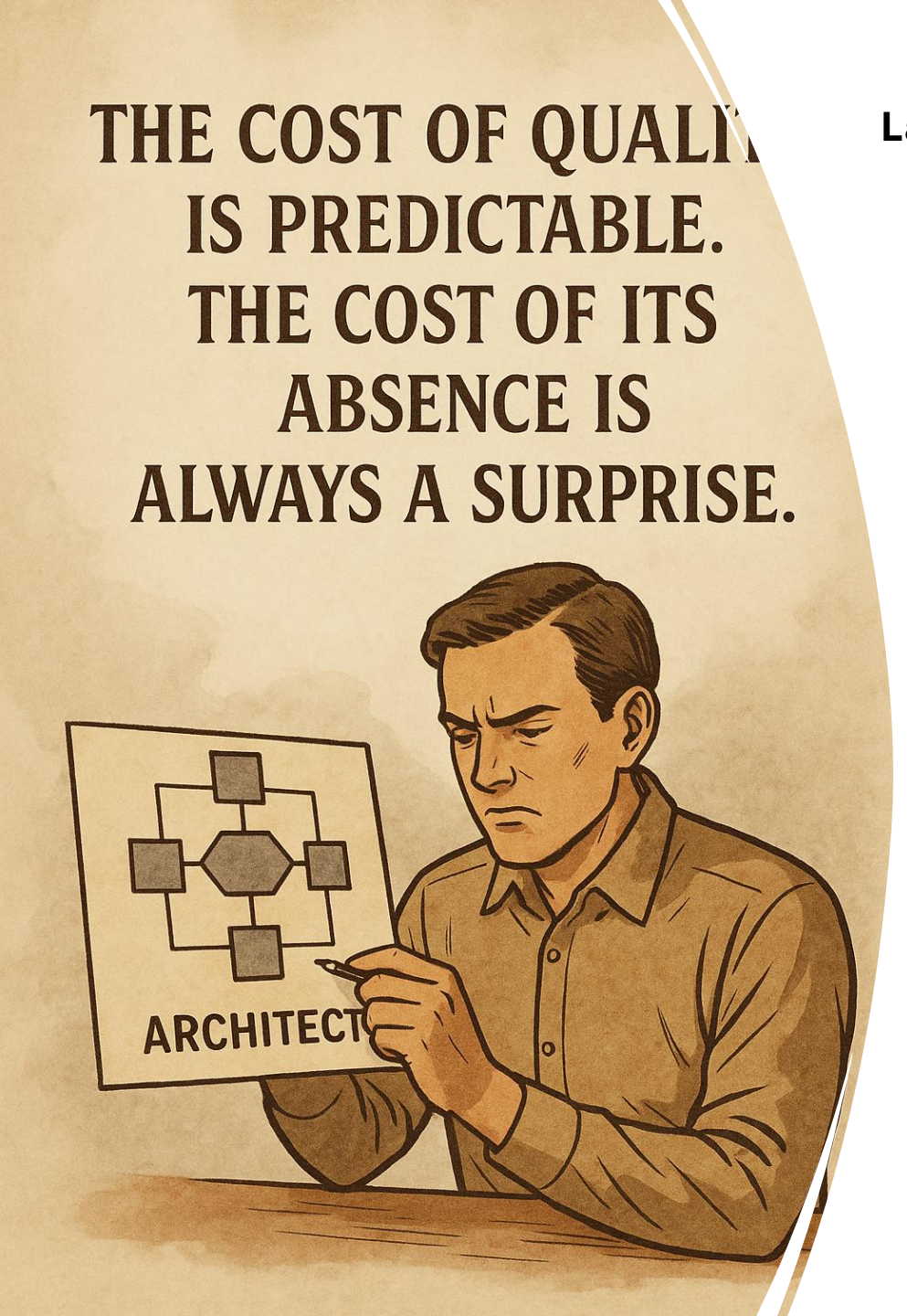




- **Frizioni**:

- Costi vs Rischio
- Stakeholder vs Architect

- **Paura**
  - Non ci sono test
  - Non conosco il dominio
  - Chi ha scritto il codice non lavora più qui!

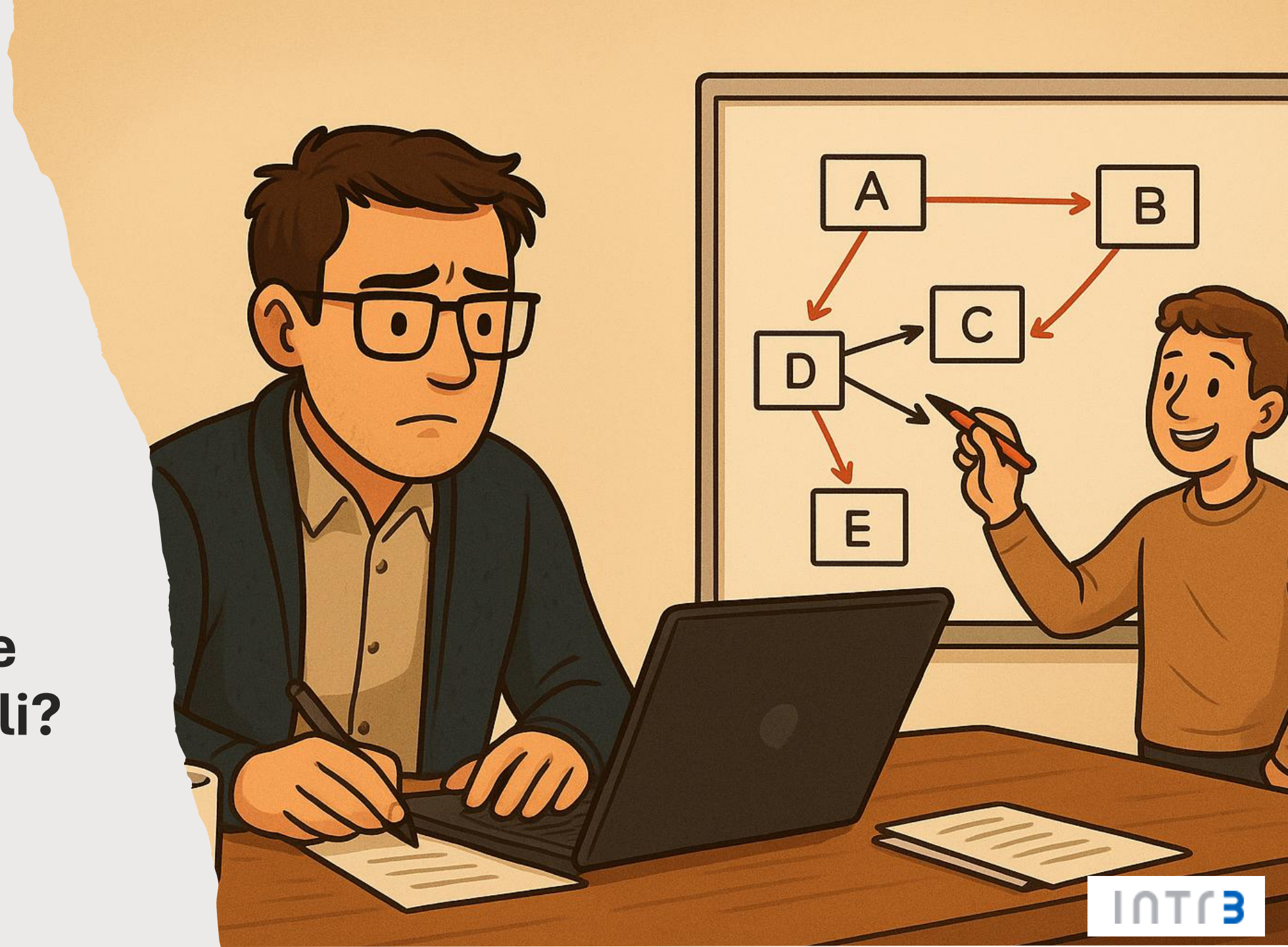


La qualità ha un prezzo. L’assenza di qualità presenta il conto!

- Il costo della qualità è prevedibile. Il costo della sua assenza è sempre una sorpresa.
- Una buona architettura costa meno di una cattiva in continua riparazione.
- La qualità non rallenta il cambiamento: lo rende possibile.

DEMO

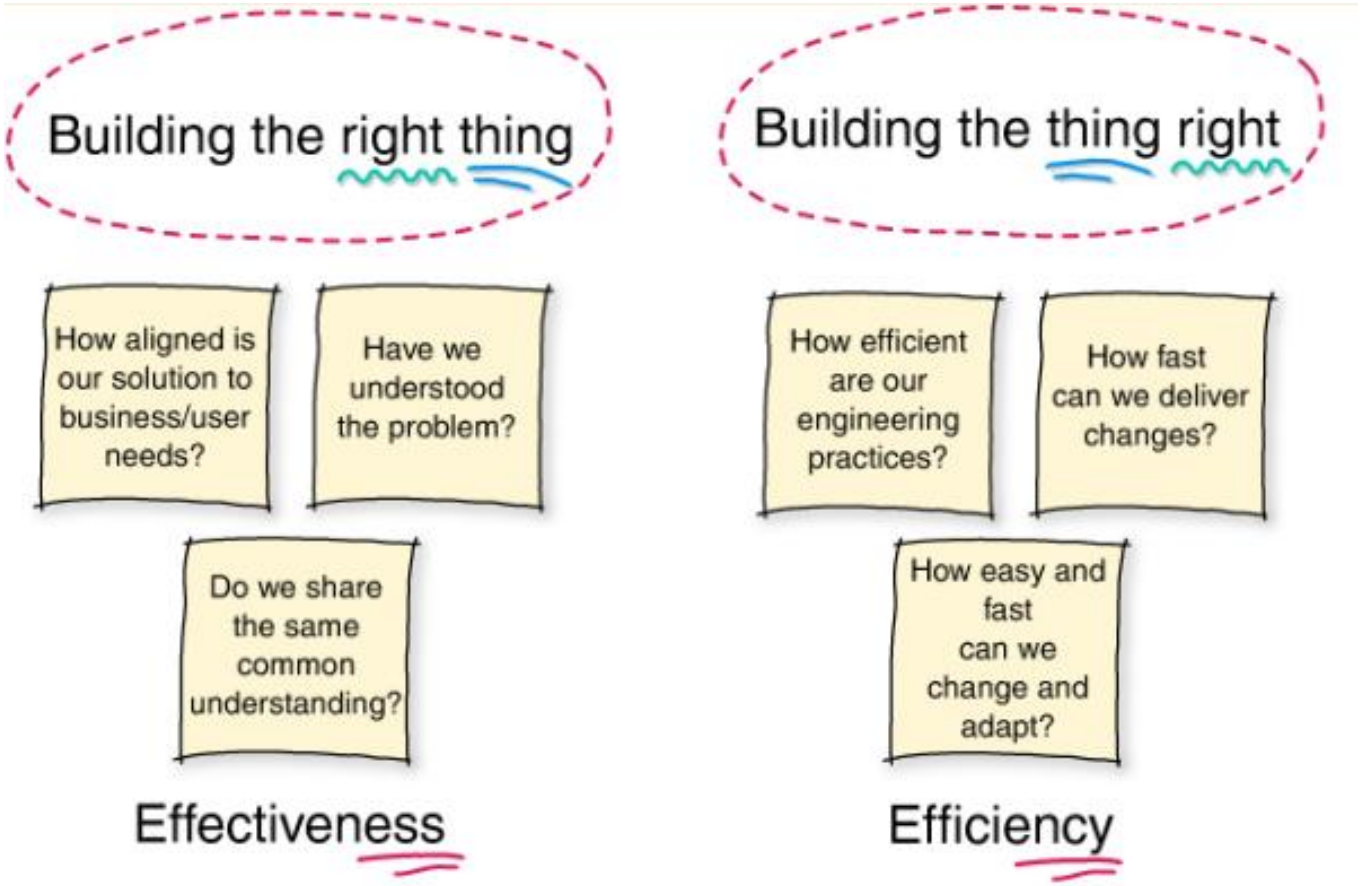
Posso dormire sonni tranquilli?



- A system is more than the sum of the behavior of its parts. It's a product of their interactions.

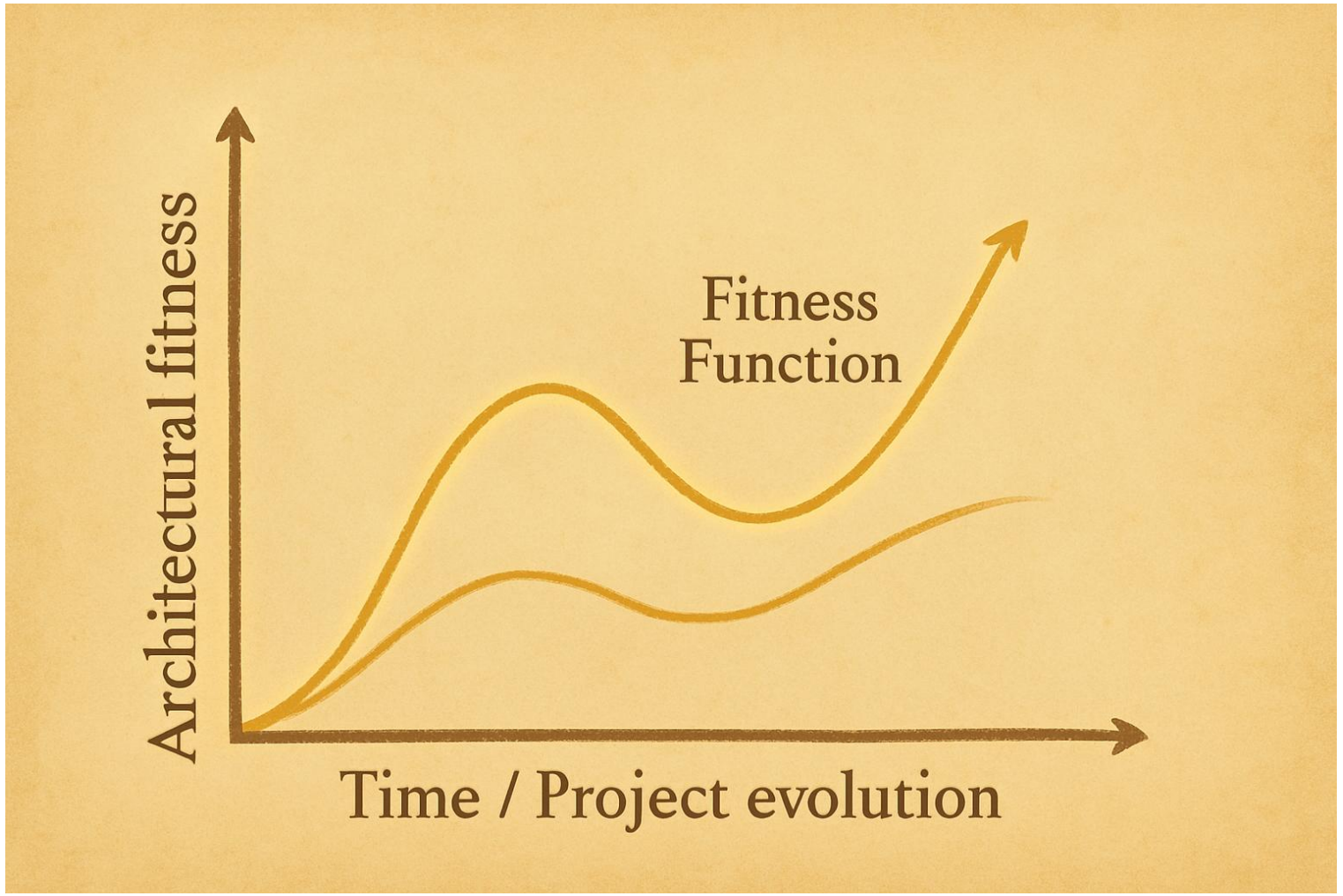
- The performance of a system depends on how the parts fit – not on how they act taken separately.

Dr. Assel Ackoff

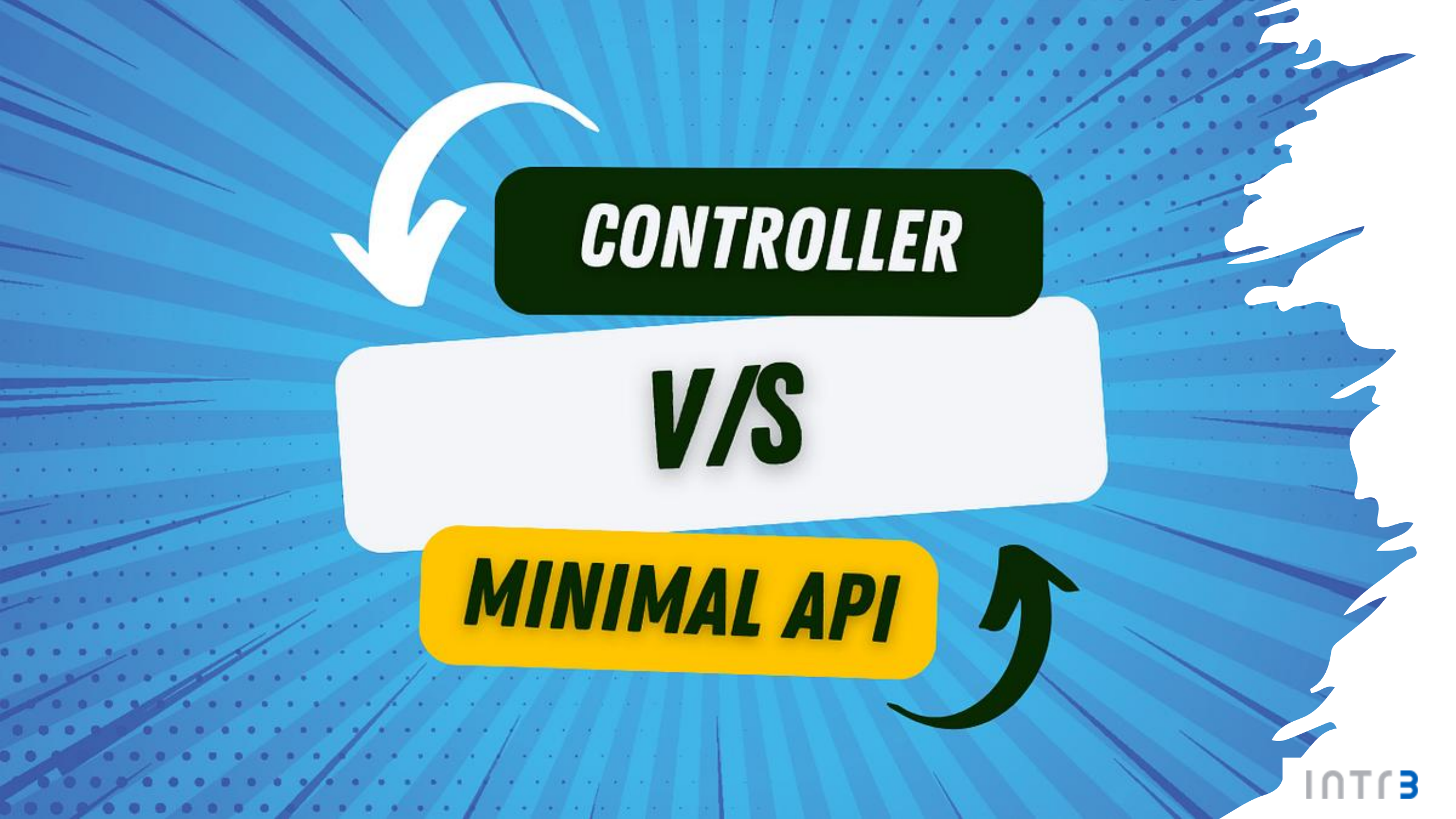


Fitness Function :

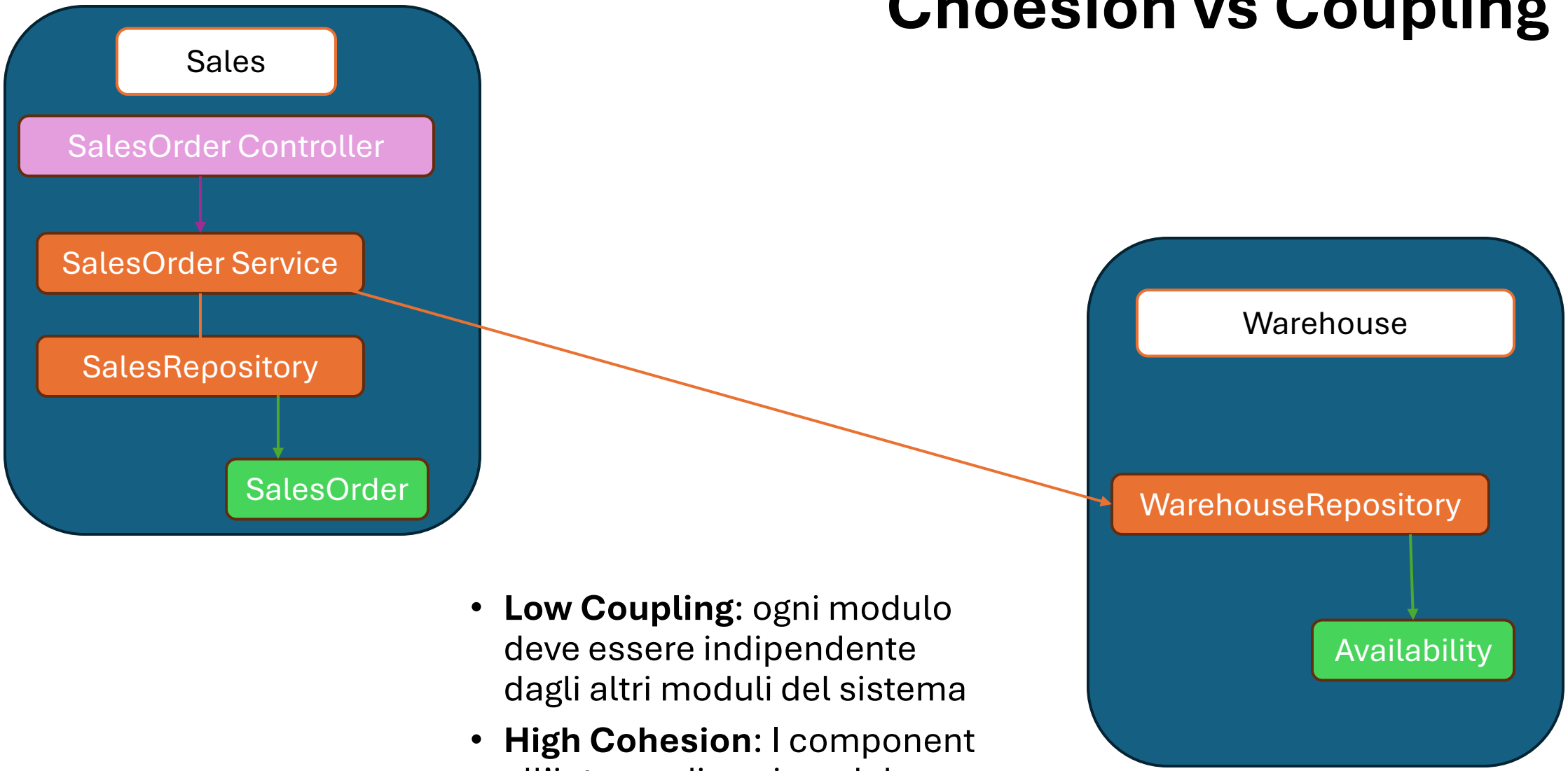
derivata rispetto all’evoluzione del sistema.



DEMO

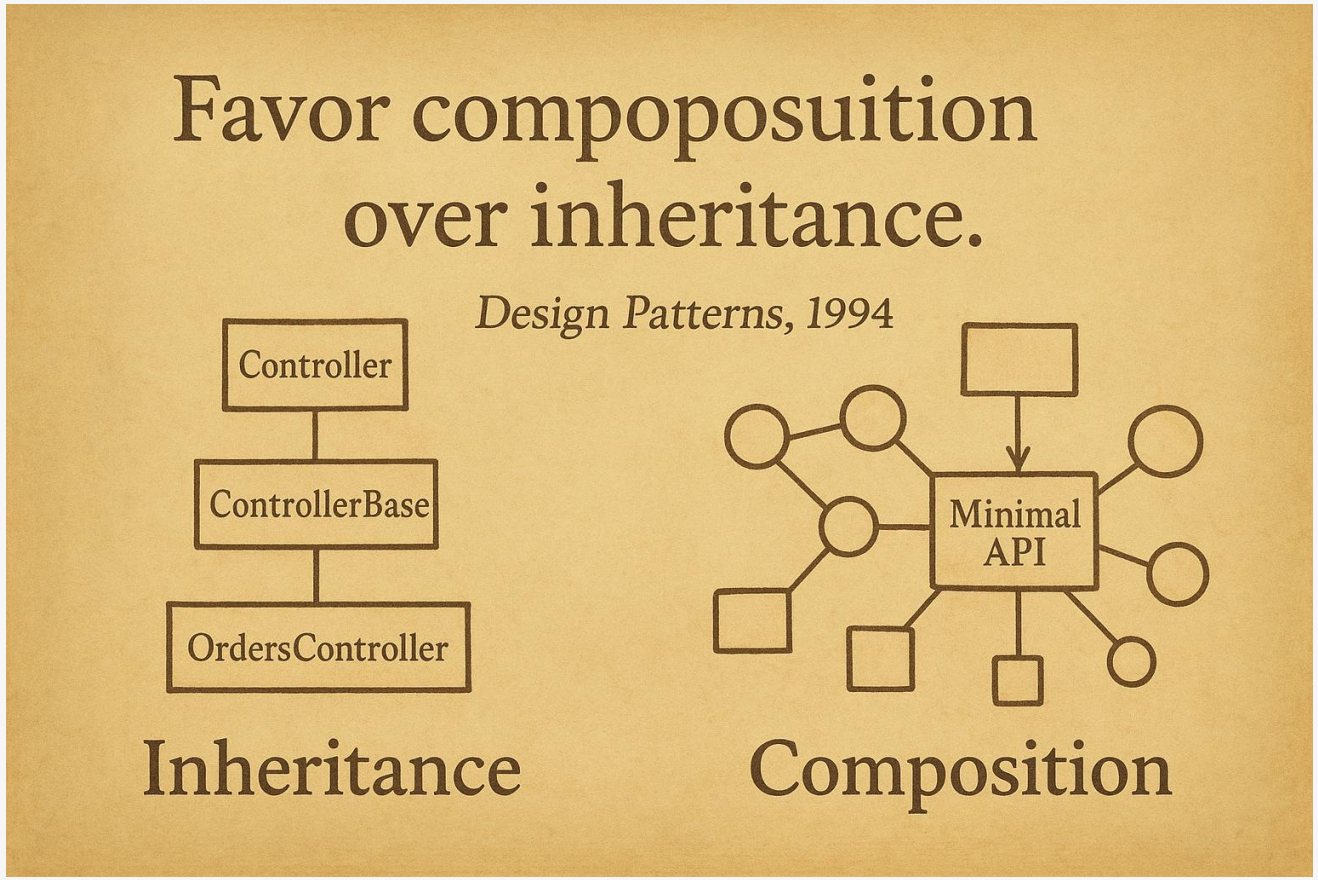

CONTROLLER
V/S
MINIMAL API

Choesion vs Coupling

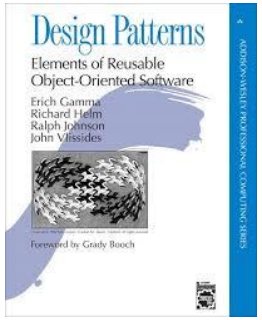


- **Low Coupling**: ogni modulo deve essere indipendente dagli altri moduli del sistema
- **High Cohesion**: I component all'interno di ogni modulo sono tutti correlate.

Ereditarietà vs Composizione

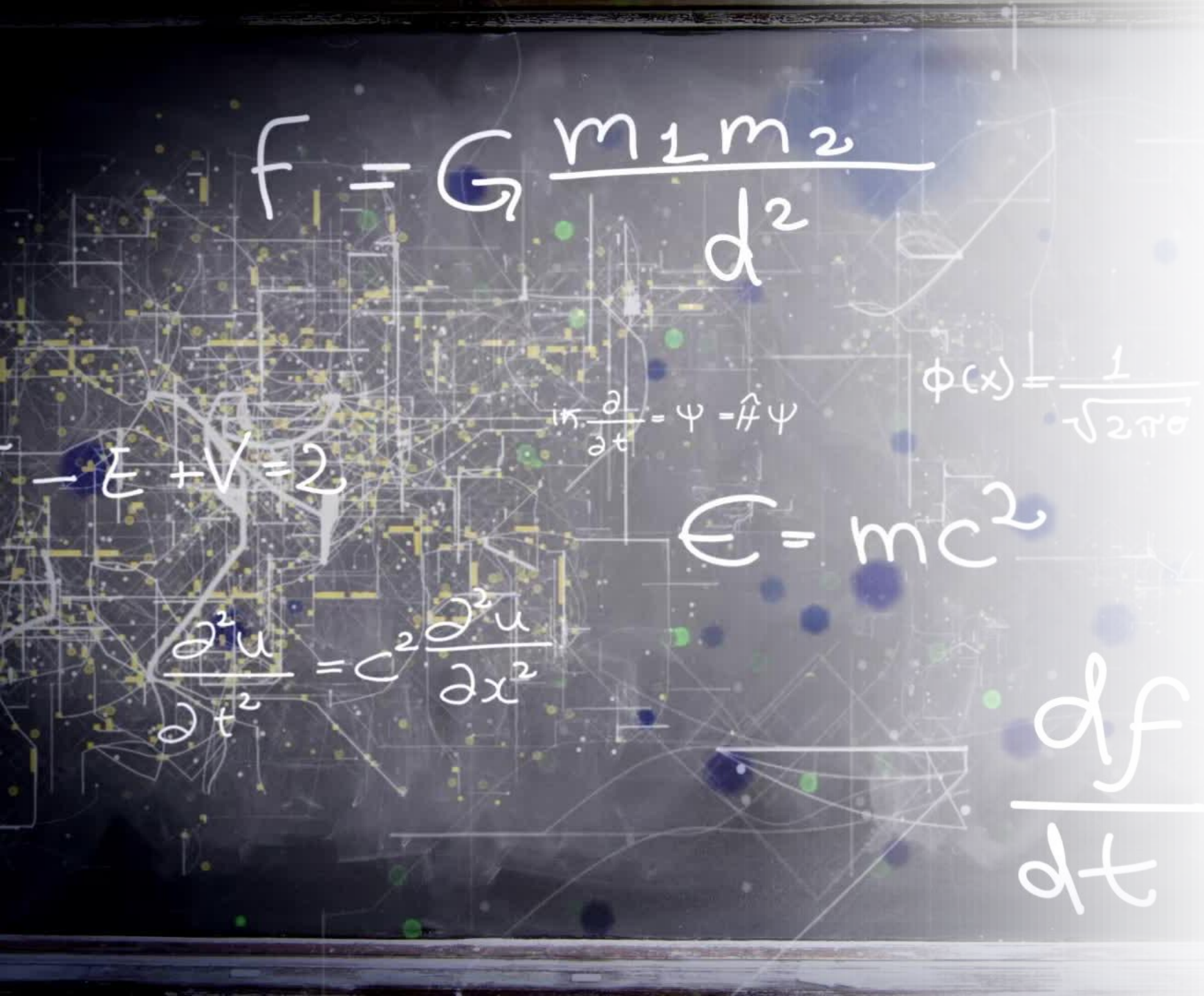


- White-box reuse vs Black-box reuse.
- La composizione aiuta a mantenere ogni classe incapsulata e focalizzata su un unico compito.



Favor composition over class inheritance

- Favoriscono alta coesione
  - Ogni endpoint è una funzione con un singolo scopo.
  - La relazione endpoint -> handler è un pattern matching.
  - Non c’è magia del framework
- Riducono l’accoppiamento implicito
  - Niente ereditarietà da ControllerBase
  - Nessun intervento implicito del framework per definire le route.



- Non dobbiamo trasformare C# in un linguaggio funzionale.
- Dobbiamo **trasformare il nostro modo di pensare il codice**.
- Ragionare il termini di funzioni pure e immutabilità ci aiuta a migliorare il codice.
- Utilizzare architetture modulari ci aiuta ad essere reattivi.
- A renderlo più manutenibile

Grazie!

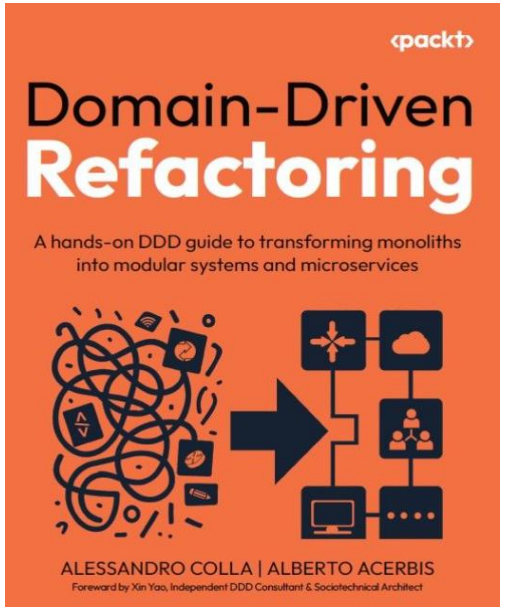
Alberto Acerbis

INTRA 25

Microsoft MVP Most Valuable Professional

alberto.acerbis@intre.it

https://github.com/Ace68/ArchitettureEvolutive



DOMAIN20