Information Hiding - creating clear deliniation between the internal implementation (what we can change) and the external interfaces.

Improved Development Time

More work done in parallel

Compreshesability

We can understand each microservice completely in isolation

Flexibility

delivery new functionality by combining microservices

Cohesion

"The code that changes together stays together"

Cohesion is high (business functionality)

Lary Constantine

Constantines Law

"A Structure is stable if cohesion is strong and coupling is low"

Coupling

Chanes in one service shouldn't cause changes to another service.

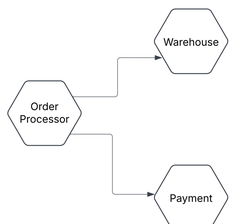
Loose coupling

Domain	Pass Through	Common Coupling	Content

Low / Loose (Most Desireable)

High / Tight (Least Desireable)

Domain Coupling - one microservice needs to make use of the functionality of another microservice.

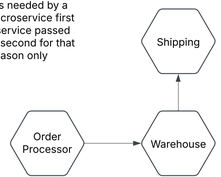


Temporal Coupling

Happens at the same time, has to complete to continue.

Pass Through

Data is needed by a third microservice first microservice passed data to second for that reason only



Shipping Manifest

ItemId

Solution 1

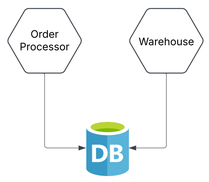
Switch to domain coupling i.e. Order Processor invokes Shipping directly.

Solution 3 - Order Processor has the responsibility of creating shipping manifest, passes as blob to Warehouse, warehouse passes to shipping.

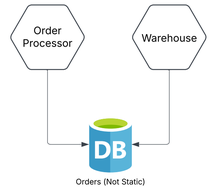
Solution 2 (Move Reponsibility of Creating Shipping Manifest to Warehouse).

Common Coupling

Multiple microservices share the use of a datastore



Read only



Reading / Writing

Id Status