Domain Driven Design(DDD)

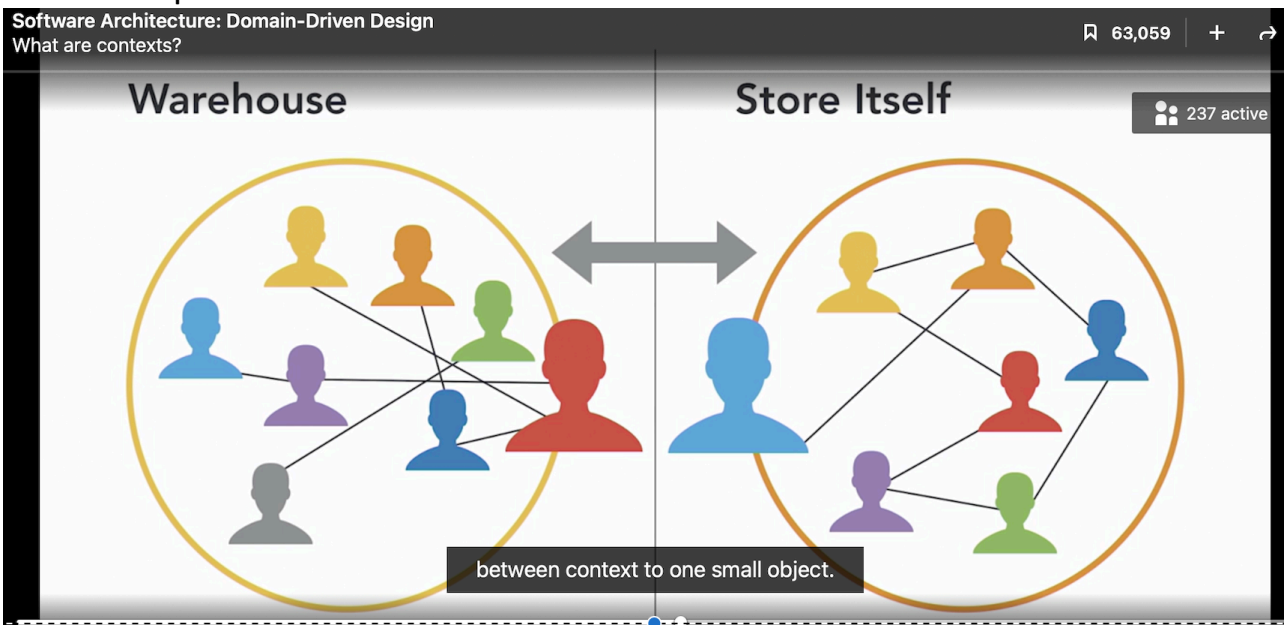
Monday, 13 May 2024 12:31 PM

DDD means Collaborative, Modeling , Incremental Developer and Business should work to
means release small functionality to prod and get feedback . Then later accommodate the

Bounded Context : Natural Division with in the business . For example if we are building th
is the context with in the store you can have 2 context as follows

1) Ware House Context - Where books is shipped
2) Sales Context -- For sale the books to end customers

Both context will have separate responsibility and distinct people are working . In DDD ea
they will not talk outside the context . In DDD there will be separate entity to talk betwee
world one person will



The individual entity with in the microservice is called as one person

Ubiquitous Language : The languages of both context is different for example the languag
Other side it is Pick off shelves , Box Books , size etc
Words described with the code it self is different for each context .

Way the entity in each context is different example Product entity is on both context but
size , weights on ware house side other side it has Image , SKU , Price . Below is design fo
see 3 context each one has separate responsibilities.

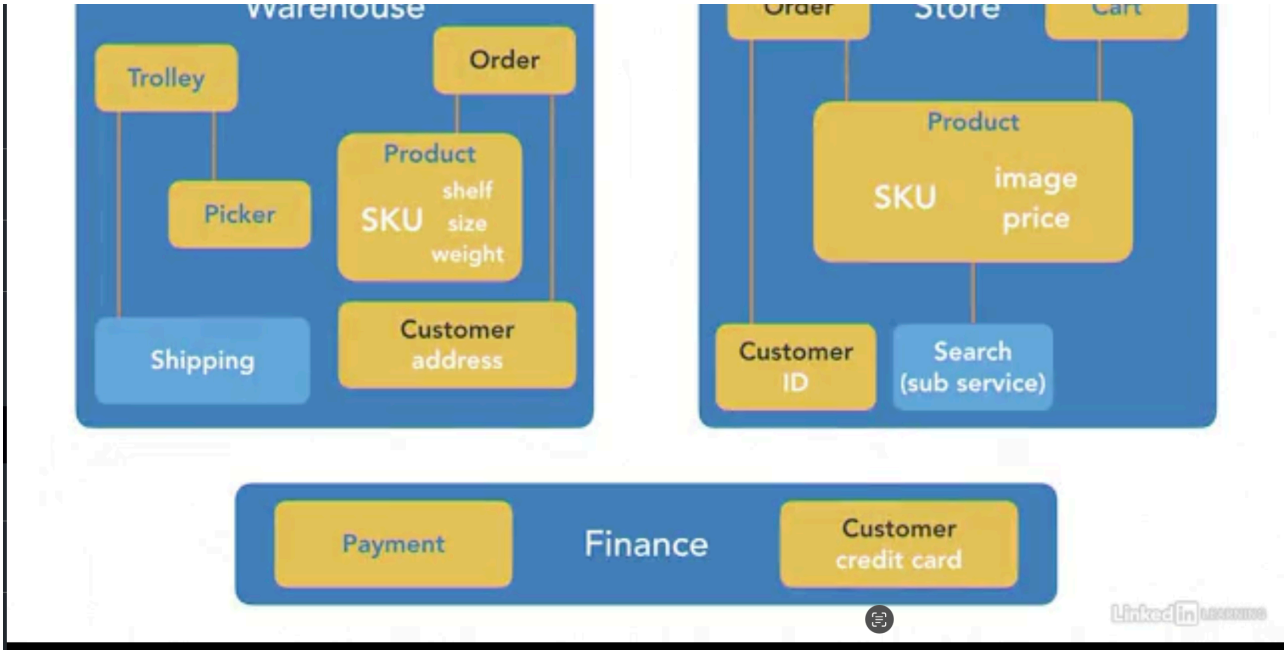
ougher on daily basis Inspect and Adopt
e feedback

he system for Book Store . Store it self

ch people with the context will talk but
n each context . It similar like in real

ge of Sale is sell , shelf or organize etc .

t each has different fields like shelf,
r Book store using DD . Here you can

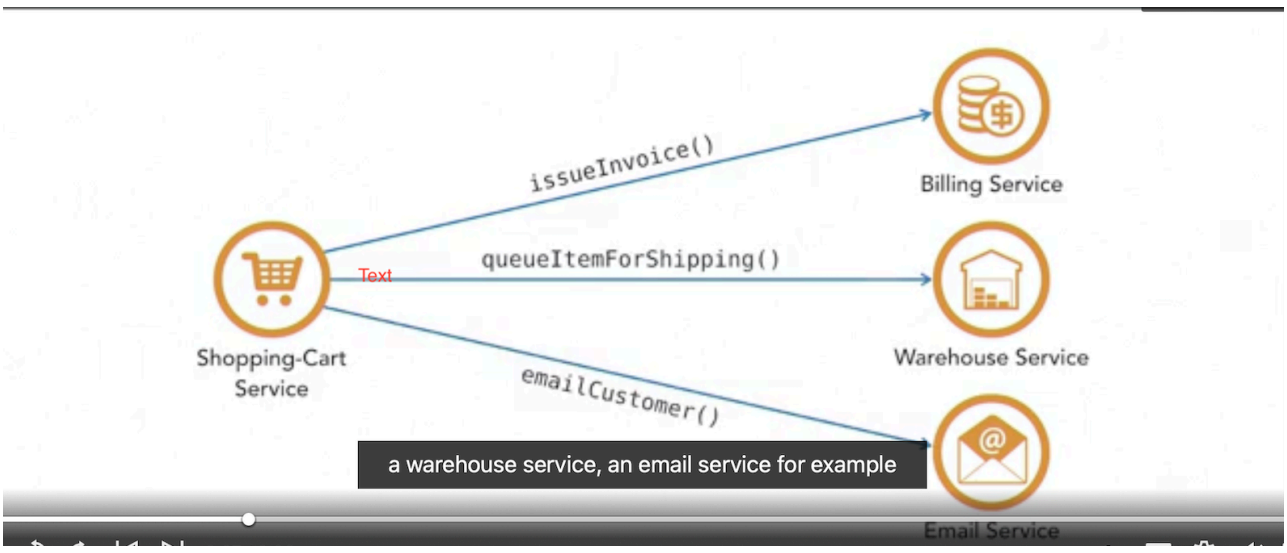


- ☑★ Every entity should be associated with one context
- ☑★ Move away from relational database thinking
- ☑★ Bad to have single product object in multiple contexts

Hashmap and concurenthasmap
RMM model for Rest needs to be ana

Orchestrated/Declarative system means one entity is talking to other entity For example i
diagram is telling 3 different services to do the task , In case billing invoice generated the
then it is an failure . The reason warehouse is not shipped is because that service is down

Declarative systems are very tightly coupled between each services

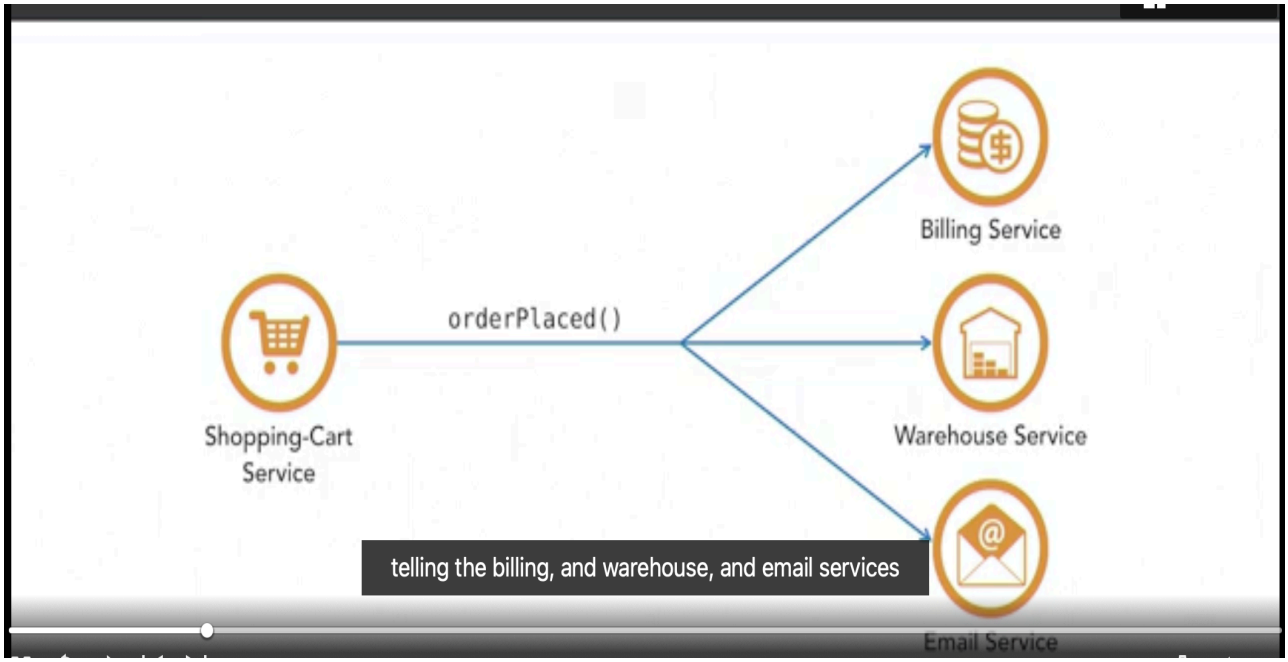


Choreography/Reactive system is the solution for declarative system . Here shopping card

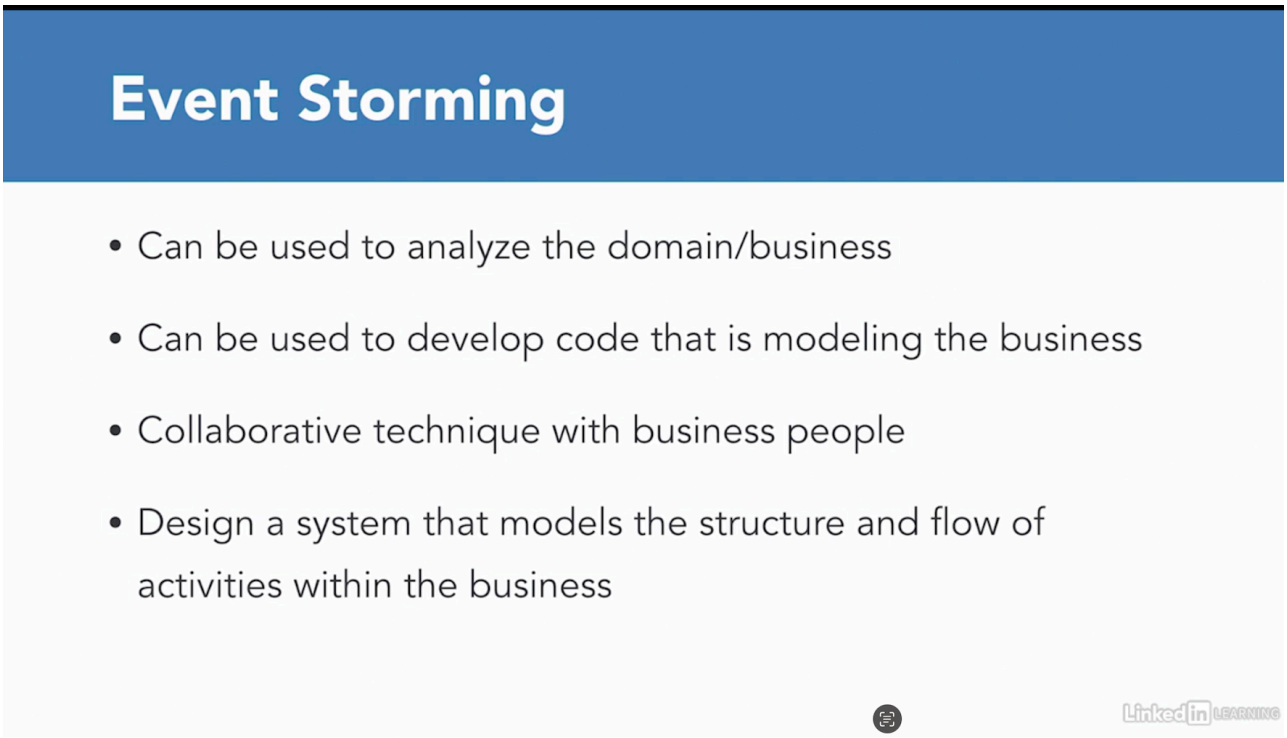
if shopping card service in below
invoice but warehouse fails to ship
at specific time.

d service will publish the message to

Kafka or any messaging system so that this message can be consumed by all 3 different se
command query request segregation .Here we removed the tightly coupling of services.



Process API is used to get t



he process created during java

ervices and do that task . This is what

Don't model too much at once
Focus on specific story
Model that part of the system in order to implement the story
Once done concentrate on implementing

Orange color sticky note is event
Blue color sticky note is action
Read color sticky note is Things that we have to do
Purple color sticky note is policy that controls how the action play out
Yellow color sticky note is Human activity
Pink color sticky notes from external systems

https://miro.com/welcome/MVd1NUI3cXRueG1jTXFsZm1qdFYwNjUxV2NLTmtRdHVON0F
aU80Q1p3bXwzNDU4NzY0NTg5MjEyNjQ0NzM0fDQ=?share_link_id=388919487599

12 factor methods in
Distributed caching technique (fault)
Cache

<taglib id="PortletTLD"> <taglib-uri>http://java.sun.com/portlet</taglib-uri> <taglib-locat
portlet.tld</taglib-location> </taglib>

https://epam.udemy.com/course/domain-driven-design-and-microservices/?src=sac&kw

RjdlljdmF6c0NhUU13eWl2cWNNWm5v

tion>/WEB-INF/tld/std-

=domain+driven+arch