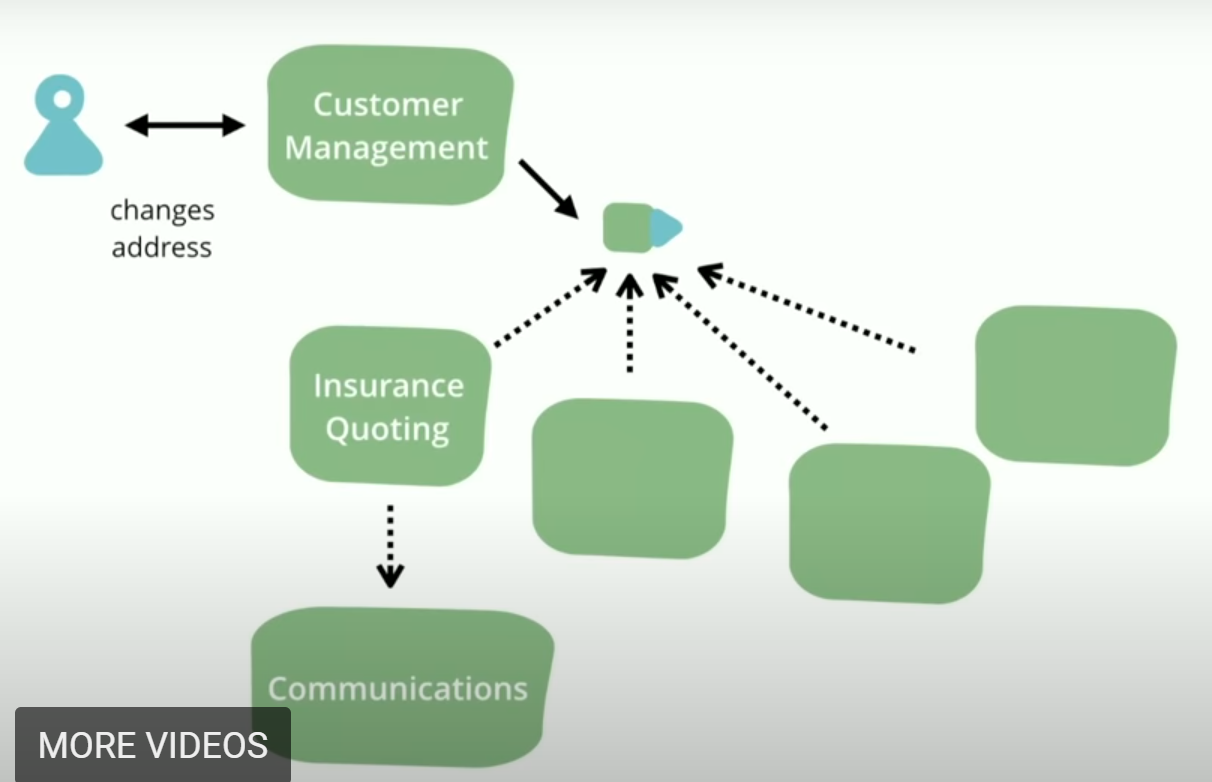
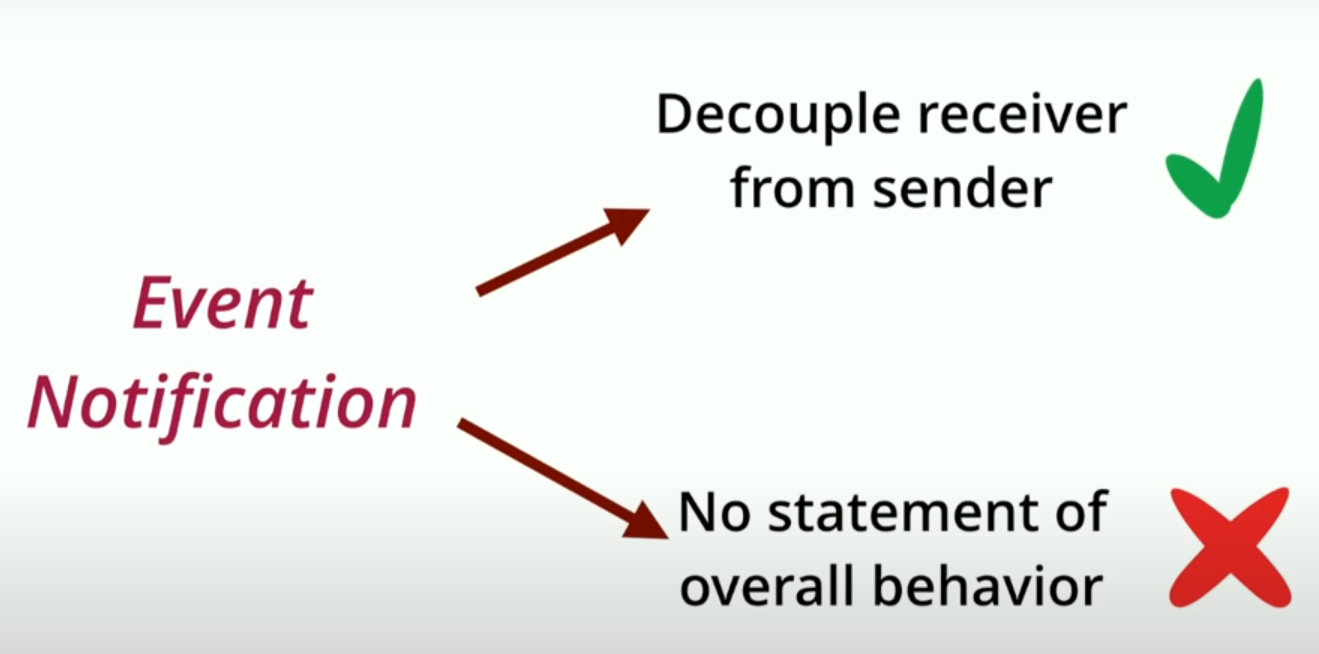
Customer Management calling each new service and providing details. Customer Management must know how to talk to each service and maintain compatibility. How about the reverse? Other services to come and take information but customer management is not disturbed.  
  
A **Message** can be of an Event or a Command or Event + Command. An **Event** is about the past and **Command** is about an instruction. Designing around events is recommended as it encourages decoupling and increases availability.

**Event**: Information about the past – that has happened. Address changed. Inventory shipped. This is not a request but just information for other services. This is what I have done, I do not care about the next steps – I am just done, If you have anything to do with this, it is up to you. More subscribers can be added, and the event producer need not even have to worry about it. New consumers can easily plugin to the channel of events. Sender/Publisher would not even be aware of the new consumer/receiver.

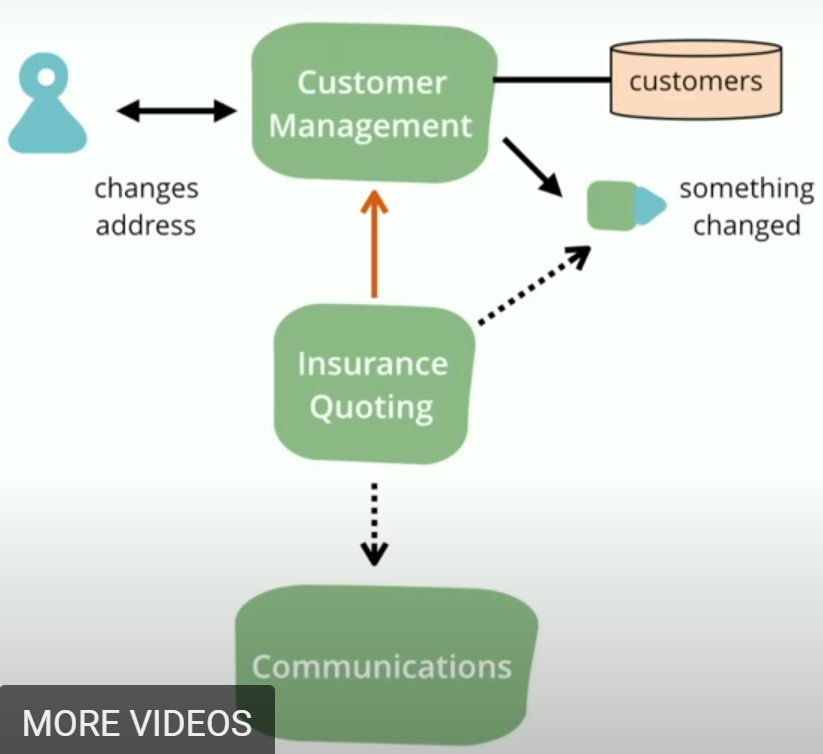
**Event Notification**

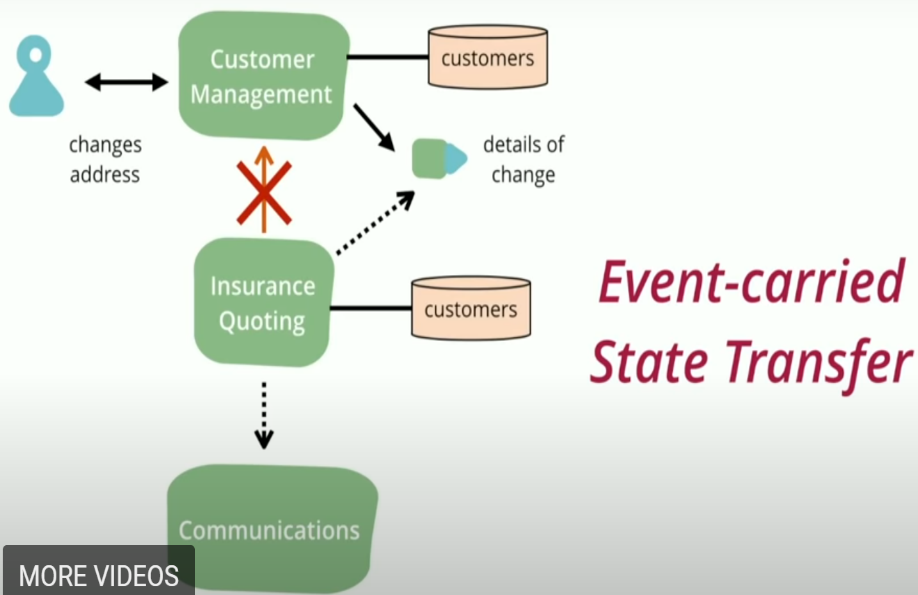


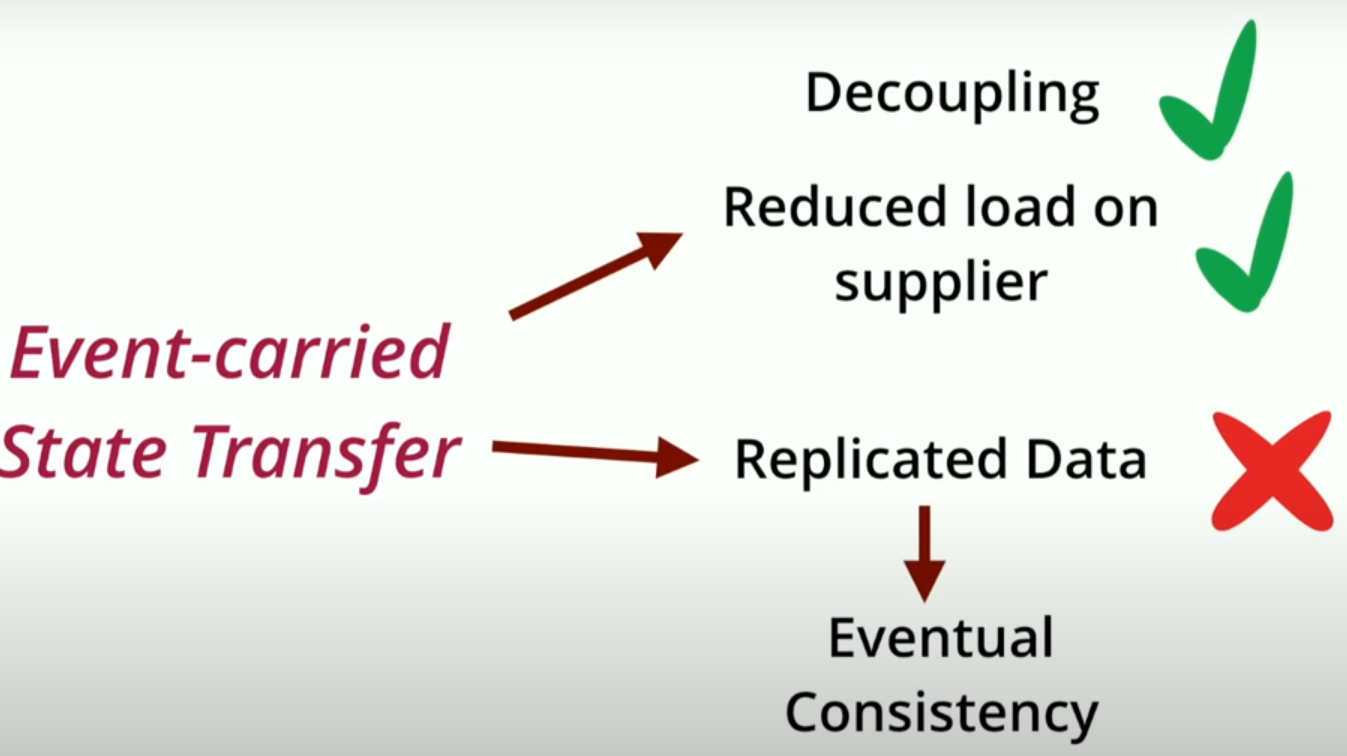


The difficulty in Event Notification is tracing and understanding how things have happened. There is no continuity, one drops event, the other picks later. It is not instantaneous handshake to visualize connectivity.

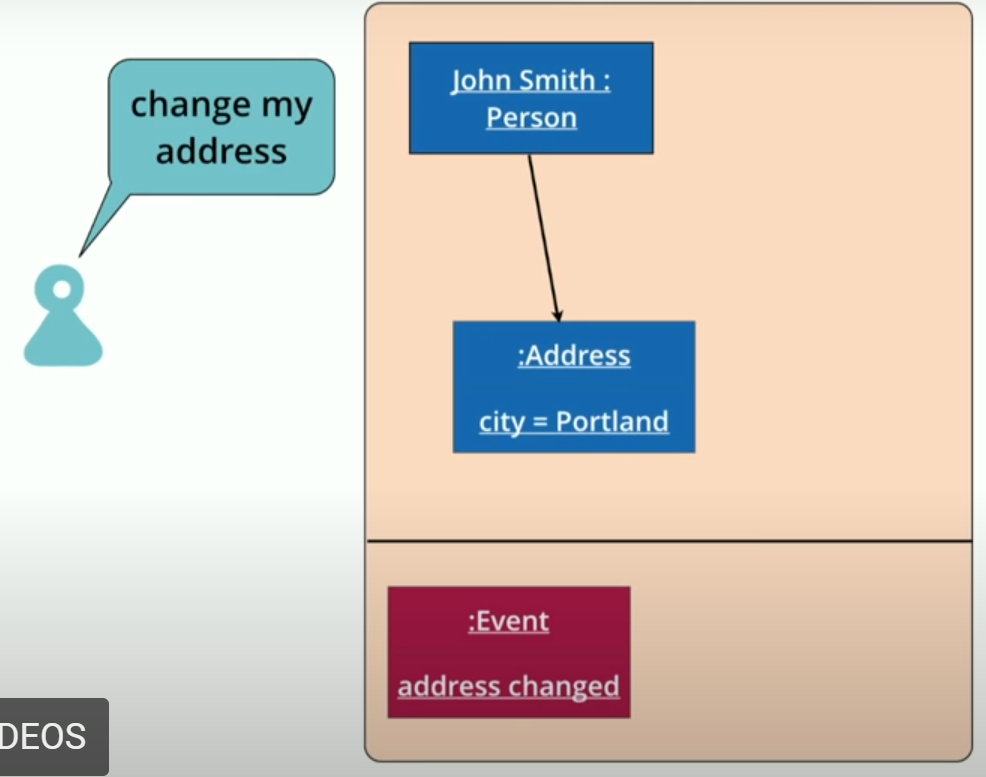
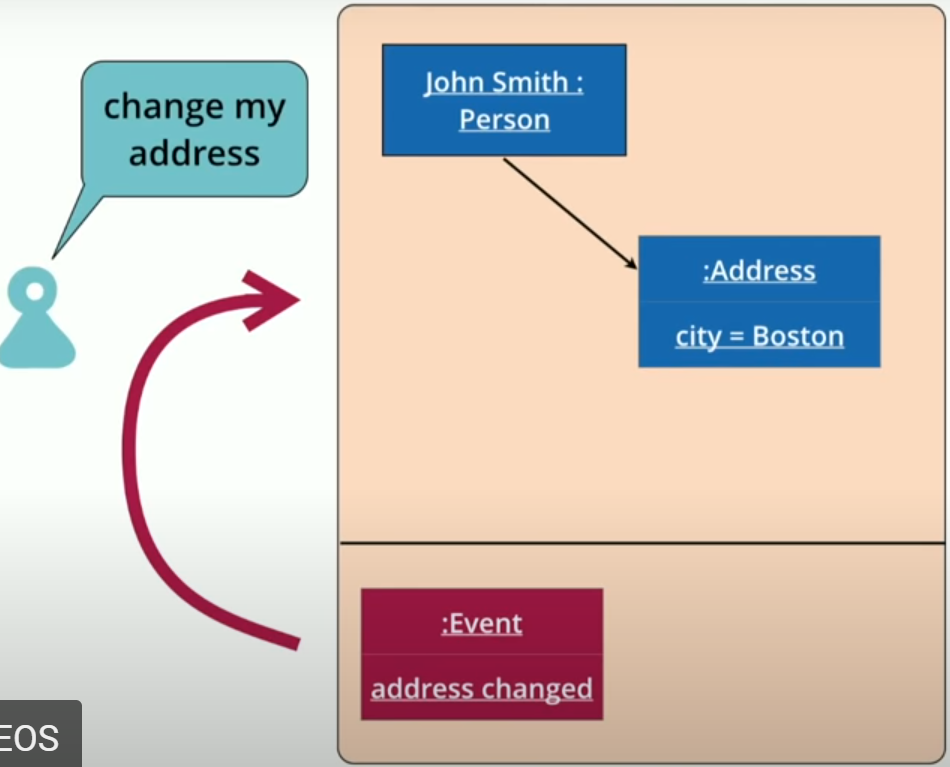
**Event-Carried State Transfer**

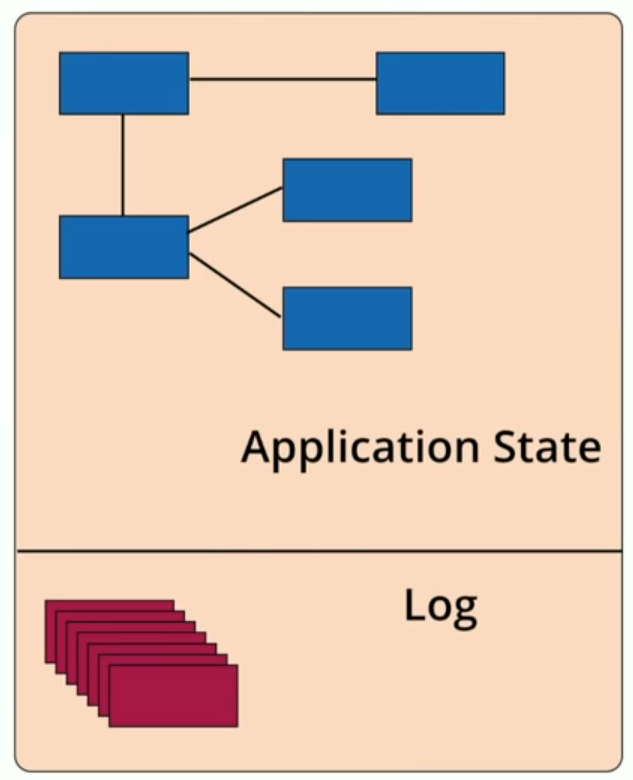
Does event sent by the sender has all the information required by receiver? If not, we would call the sender again to get more information. This is READs but all receivers might bombard the sender with too many requests and can affect the availability of sender.

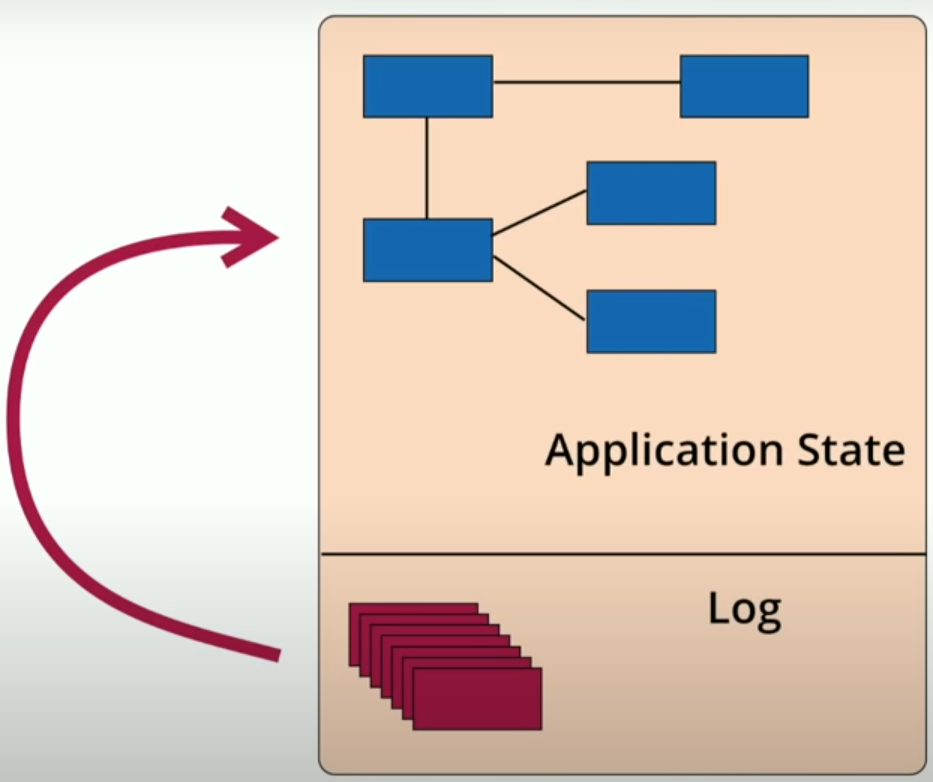
 So how about receiver keeping a track of all the events and deduce the information required when needed? Hence required data from the sender is copied/duplicated (only required data) into DB at receiver. Like What is old address? Receiver can go back into the DB history copied during past events and get old address. But data duplication. What if the copied/duplicated data now does not solve the use case we have right now??? We end up going to sender again. Not very much adopted pattern but if implemented well, it would give lot more availability and decoupling between sender and receiver.

 When we talk about a lot of decoupling and feel proud about independent and highly available, the side effect or the hidden cost is lack of consistency or eventual consistency.

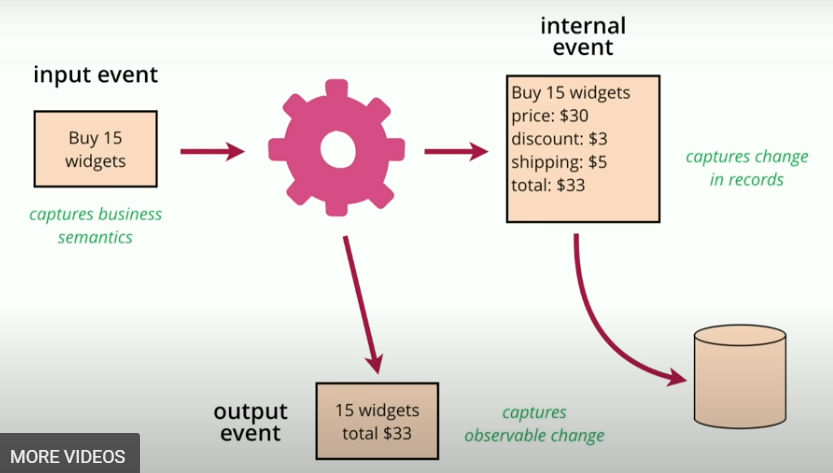
**Event Souring**

**** ****

****

****

****

What if the bug is fixed and now the calculations are correct??? The numbers would not match properly.  


Actual total is 33 but because of bug we assumed it as 33 and published the same. Now if we try to replay after but fix, we would end up with 32 not matching the same.

All these things like bug fixes and versioning do not impact much if we only think of replaying/restoring only from past 24 hours like LMAX does.

**Command Query Responsibility Segregation CQRS**

