Architecture is mainly a trade off analysis

Microservices architecture is iterative first disintegrating and integrating

Why need Micro services

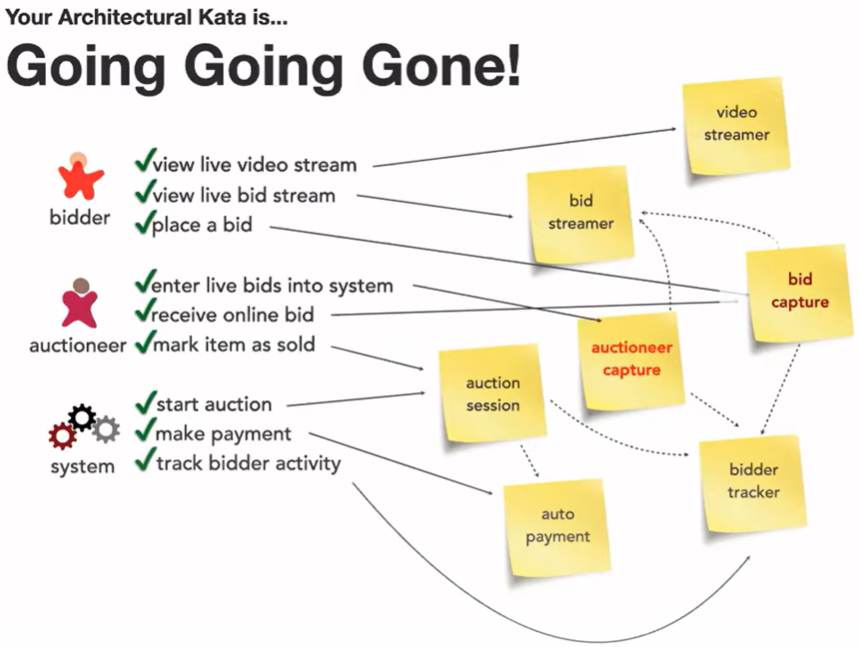
A red swiss army knife

Description automatically generated A multi tool with many different tools

Description automatically generated

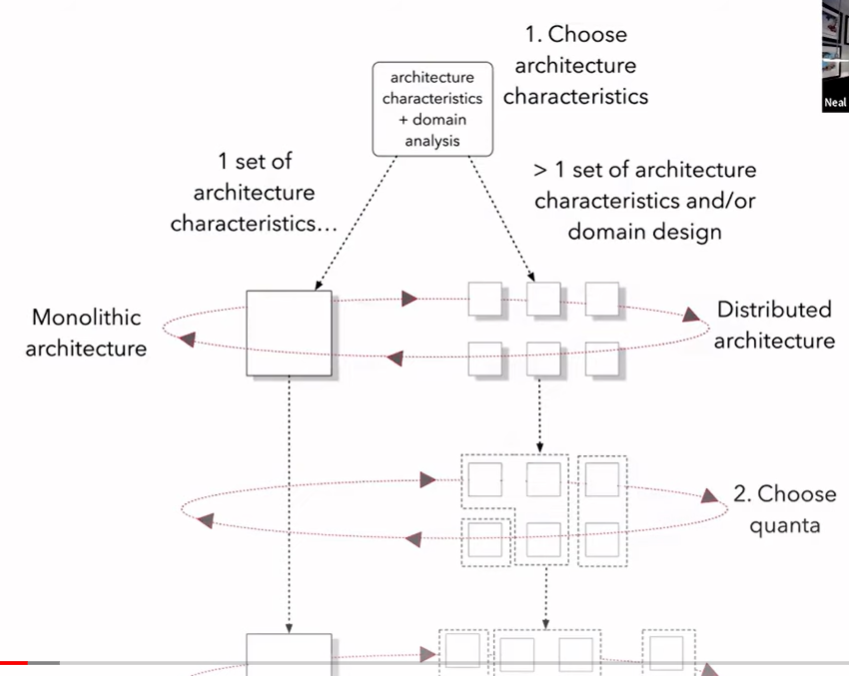
A diagram of a structure

Description automatically generated

 A diagram of components with black text

Description automatically generated

From Domain standpoint bid capture and auctioneer capture these are same but from architetcure standpoint they are very different. Auctioneer is one but bidders are thousands. So scalability , reliability is different.



Following uis a toolbox

Characteristics of microservices  


**parses**

**Service Granularity**

Two fdifferebce firces to decide granuality

1. Granularity disintegrators
2. Granuality integrators

**Granularity disintegrators**

* F- Service functionality – single responsibility. Too much chit and slow system
* CV-Code Voltility - if some thing changes more often then others than it can be taken out
* S-Scalability through put
* S-Security
* E-Fault tolerance reliability
* Plus your organization : disintegrators

**FSC(V) SSE**

**Granularity integrators**

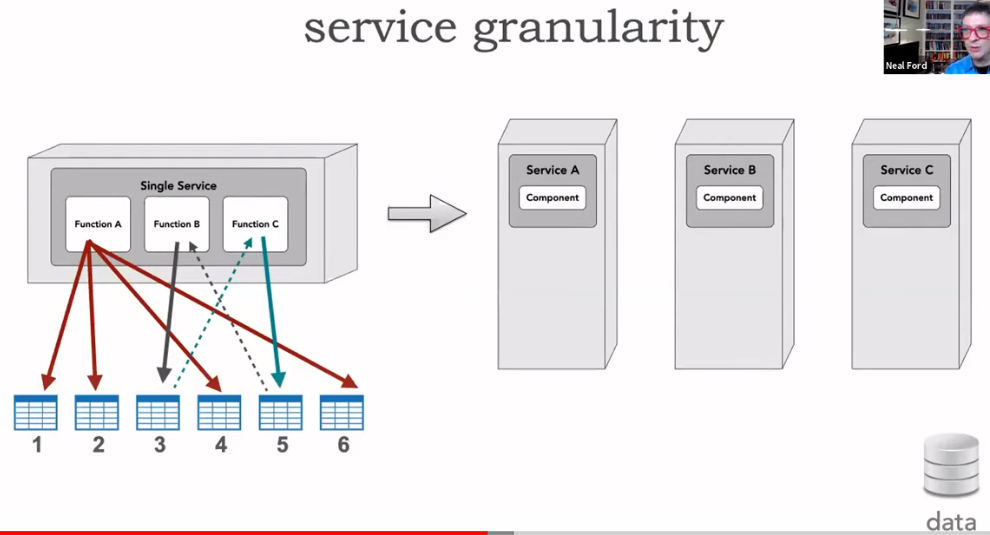
* **T-Transactionality: to avoid distributed transactions across micro services**
* **D-Data dependency: referential integrity less relication**
* **W-Workflow and chroregraphy:**

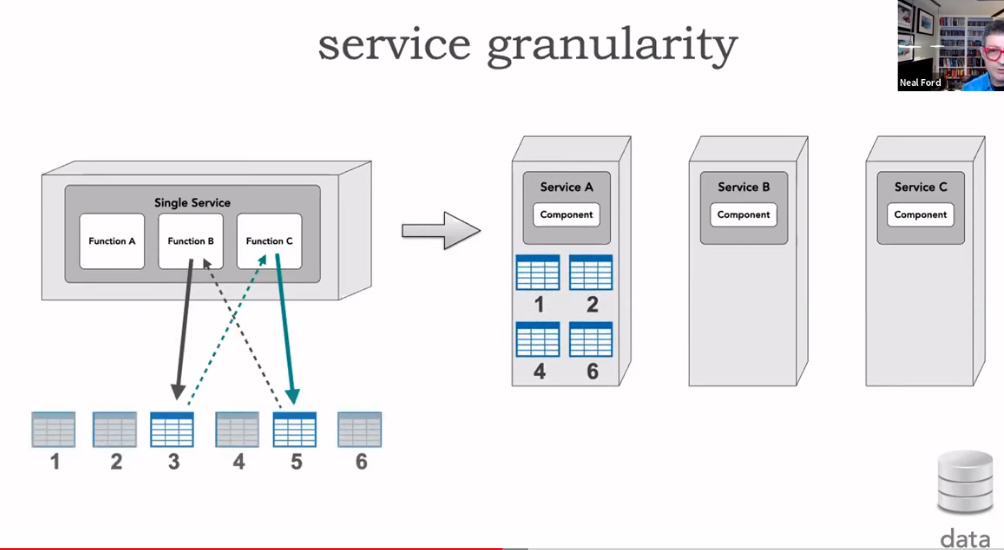
**Transactionality**

A diagram of a service granularity

Description automatically generated

**Data Dependency**





A diagram of a service granularity

Description automatically generated

**Following idea breaks encapsulation of service C and any change in DB5 will break ServcieB**

A diagram of a service granularity

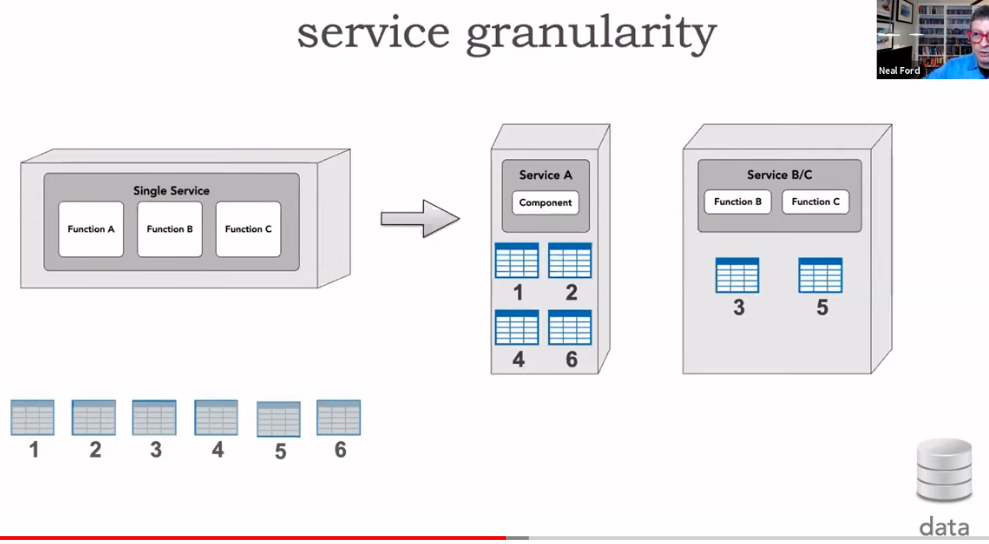
Description automatically generated

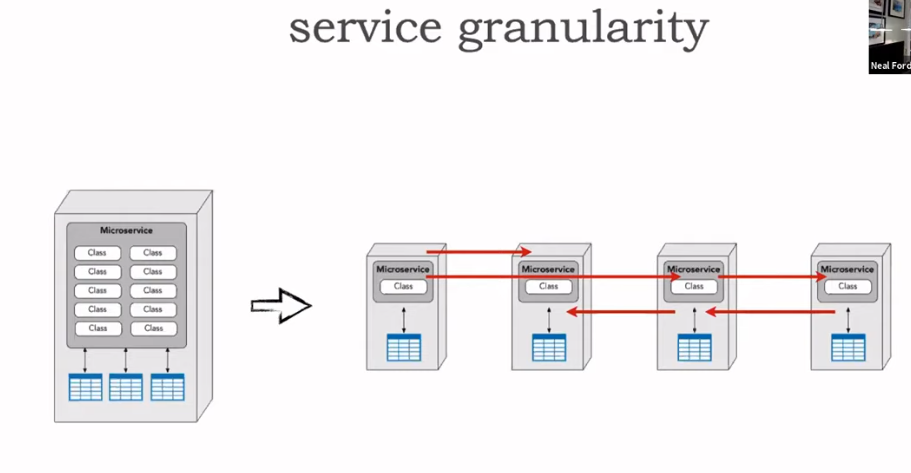
Following is fine archirtecturteally but may introduce performance issues.

A diagram of a service granularity

Description automatically generated

**If a lot of chat between B and C then bundle them unless they have some other architectural differences i.e. parses Perf, availability, reliability, scalability, exception/fault tolerance, security**





A diagram of a service

Description automatically generated

A diagram of a service

Description automatically generated

A diagram of a service granularity

Description automatically generated

A diagram of a service granularity

Description automatically generated

**Communication**

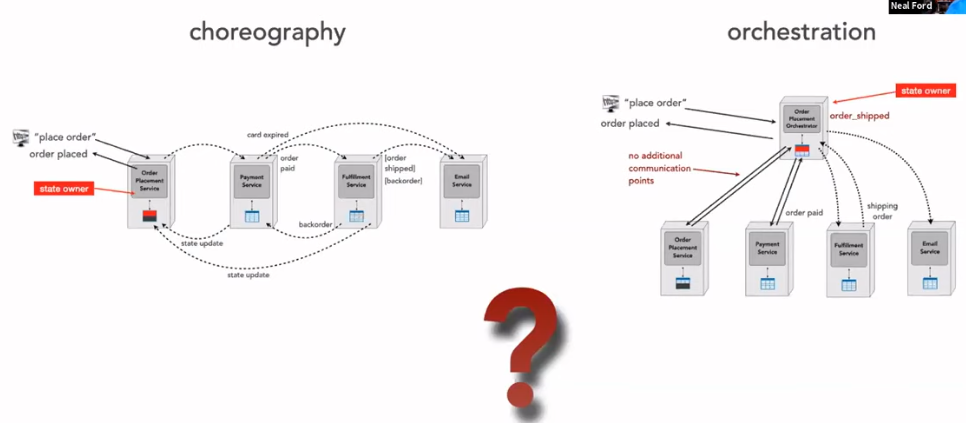
A chart of different types of goods

Description automatically generated with medium confidence

**Choreography**

A close-up of a chart

Description automatically generated

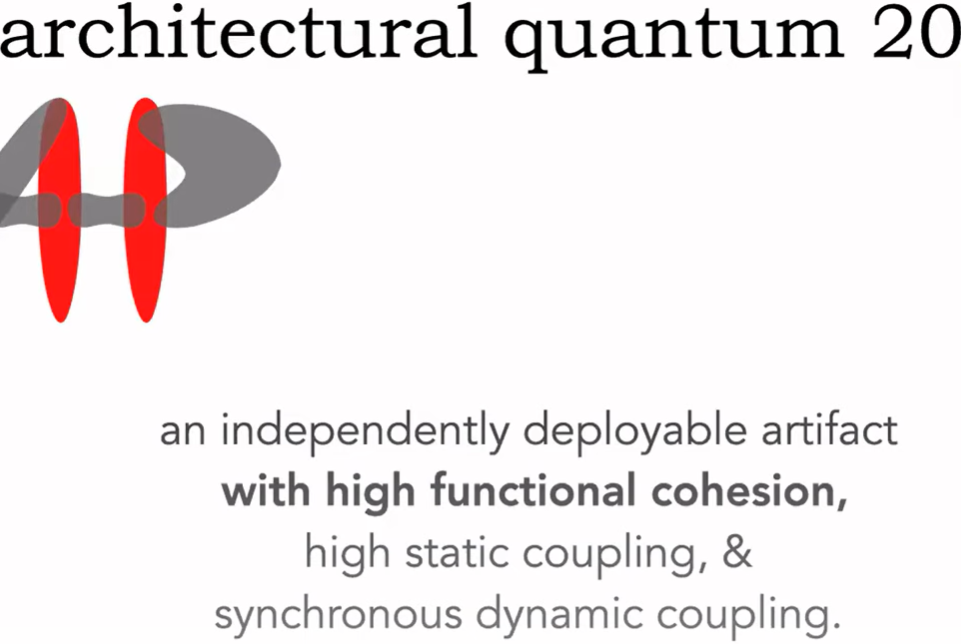


A diagram of workflow

Description automatically generated

A diagram of a workflow

Description automatically generated



**Static dependency means all those including transitive which are required to bootstrap a service.**

