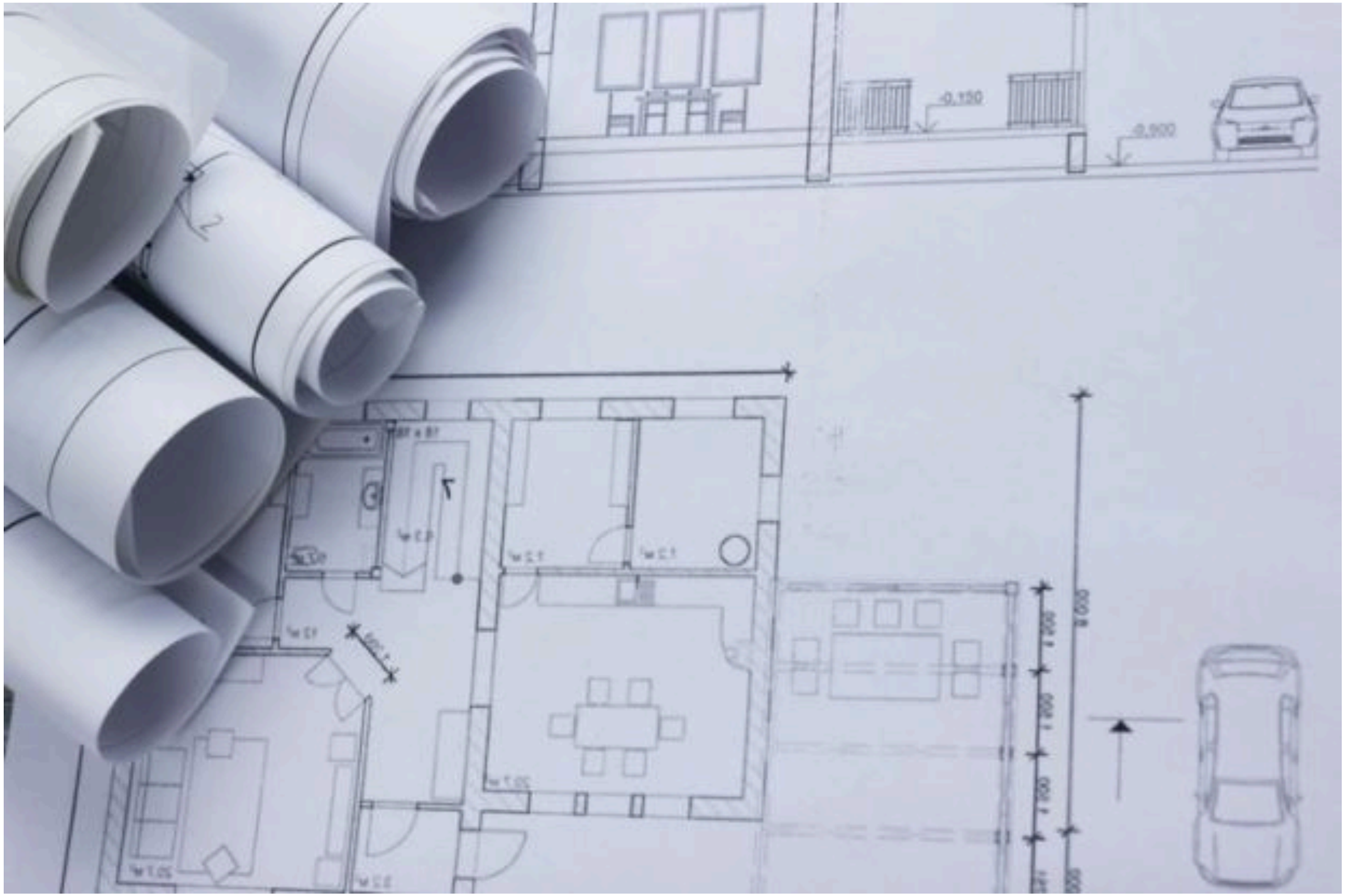


Thinking and Designing Architecture

» Skan.ai - chief Architect

» Ai.robotics - chief Architect

» Genpact - solution Architect

» Welldoc - chief Architect

» Microsoft

» Mercedes

» Siemens

» Honeywell

Mubarak

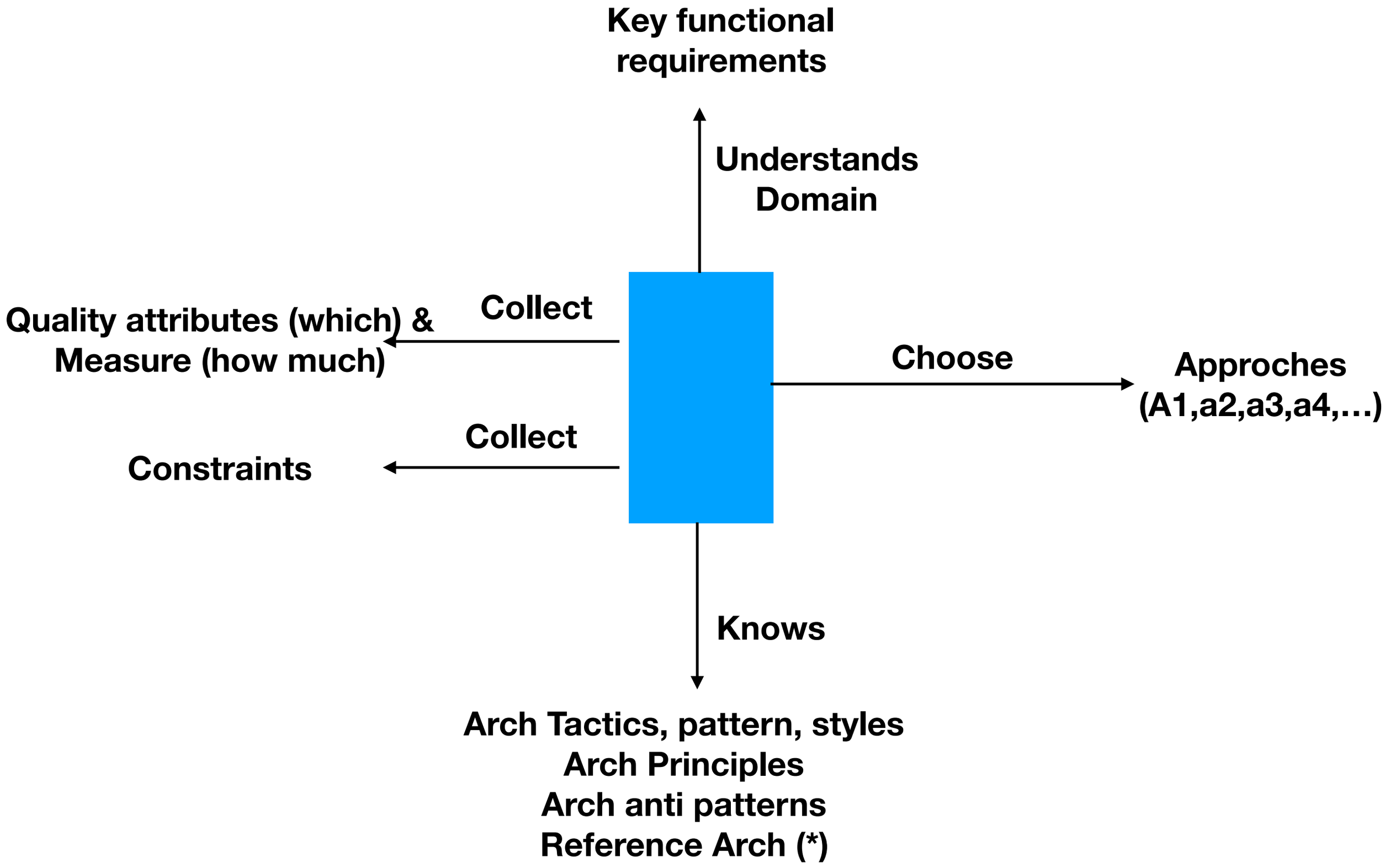
- Arch Requirements (QAW)
  - Context, key fun, quality, constraints, assumption
- Arch Decisions
  - Logical, Deployment, …
- Arch Eval (ATAM, ARID, SAAM)
  - Justification

Part I

Architecture nut and bolts
What is a Architecture Blue print?
How is Architecture different from design ?
What is difference between Application Architecture and Enterprise Architecture ?
What is Role of a Application Architect ?
Application Architect vs Solution Architect
Application Architect vs Vertical Architect

Architecture vs Design

Quality	Measure	Approach
• Performance (Resource) • Maintainability • Reliability (Trust) • Robustness (Rugud) • Scalability (volume) • Availability • Security (Trust) • Usability	• tps • Response time • Latency • % uptime • % downtime • No of clicks • Probability •	• Versioning • Modularity • Concurrency • Caching • Lazy loading • ACID •


Key functional requirements
Understands Domain
Collect
Quality attributes (which) & Measure (how much)
Choose
Approches (A1,a2,a3,a4,…)
Collect
Constraints
Knows
Arch Tactics, pattern, styles
Arch Principles
Arch anti patterns
Reference Arch (*)

Engineering vs Tuning

Attribute	Measure	Approach
• Performance <— • Maintainability <— • Security (Trust) <— • Scalability (Volume - I/O, CPU, Memory) <— • Usability x • Availability <— • Reliability (Trust) <— • Robustness (Rugud) •	• Response time • Memory • CPU • I/O • Latency • TPS • …	• Caching • Parallel • Pooling • Lazy Loading • Compression • Chunking • Reusability/ Extensible • Modular /Component • Low Coupling • EDA • ACID - transaction • Input Validation

5 Communicate

Key functional requirements (80:20)

2

Understands Domain

NFR

3

Quality attributes, Measure (Architectural Requirements)

Collect

Choose

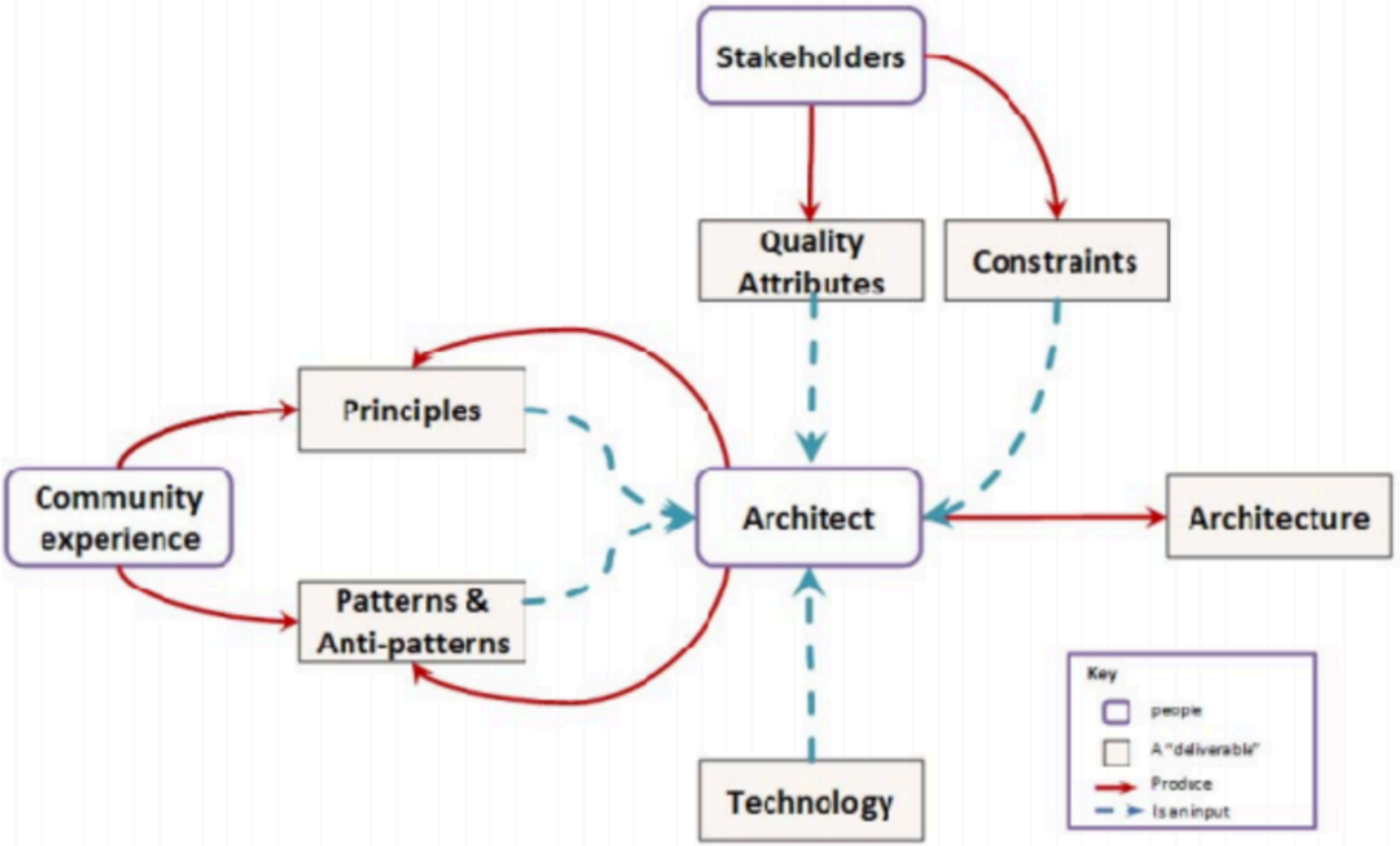
4

Approaches (Architectural Decisions)

1

Knows

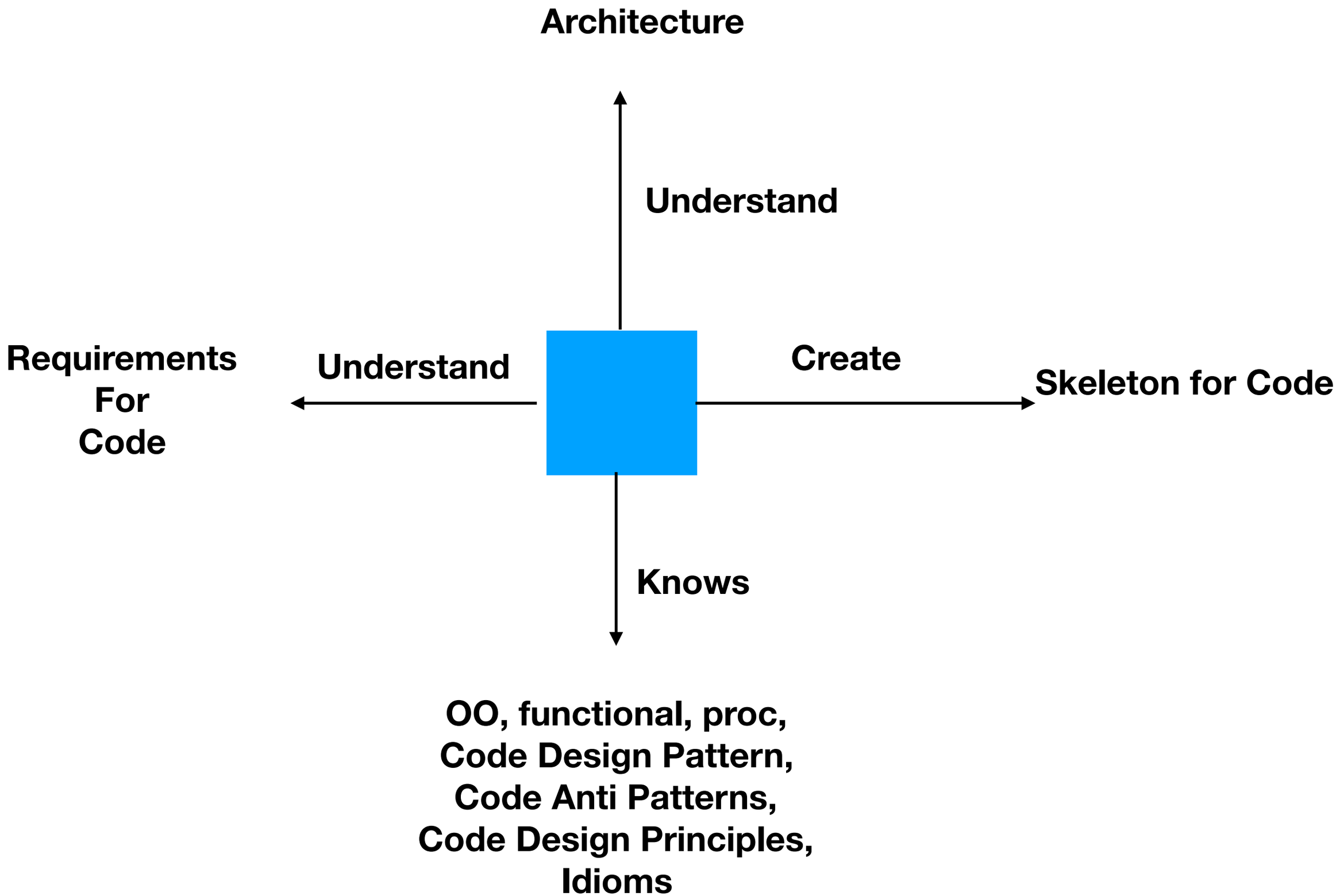
Arch Tactics, pattern, styles
Arch Principles
Arch anti patterns
Reference Arch (*)


Stakeholders
Quality Attributes
Constraints
Principles
Community experience
Patterns & Anti-patterns
Architect
Architecture
Technology
Key
people
A "deliverable"
Produce
Is an input

Pre

Engineering vs Tuning

Post


Architecture
Understand
Requirements For Code
Understand
Create
Skeleton for Code
Knows
OO, functional, proc,
Code Design Pattern,
Code Anti Patterns,
Code Design Principles,
Idioms

Architecture Design
UI Design
Test Design
Module Design
Class Design
Code Design

Architecture vs Design

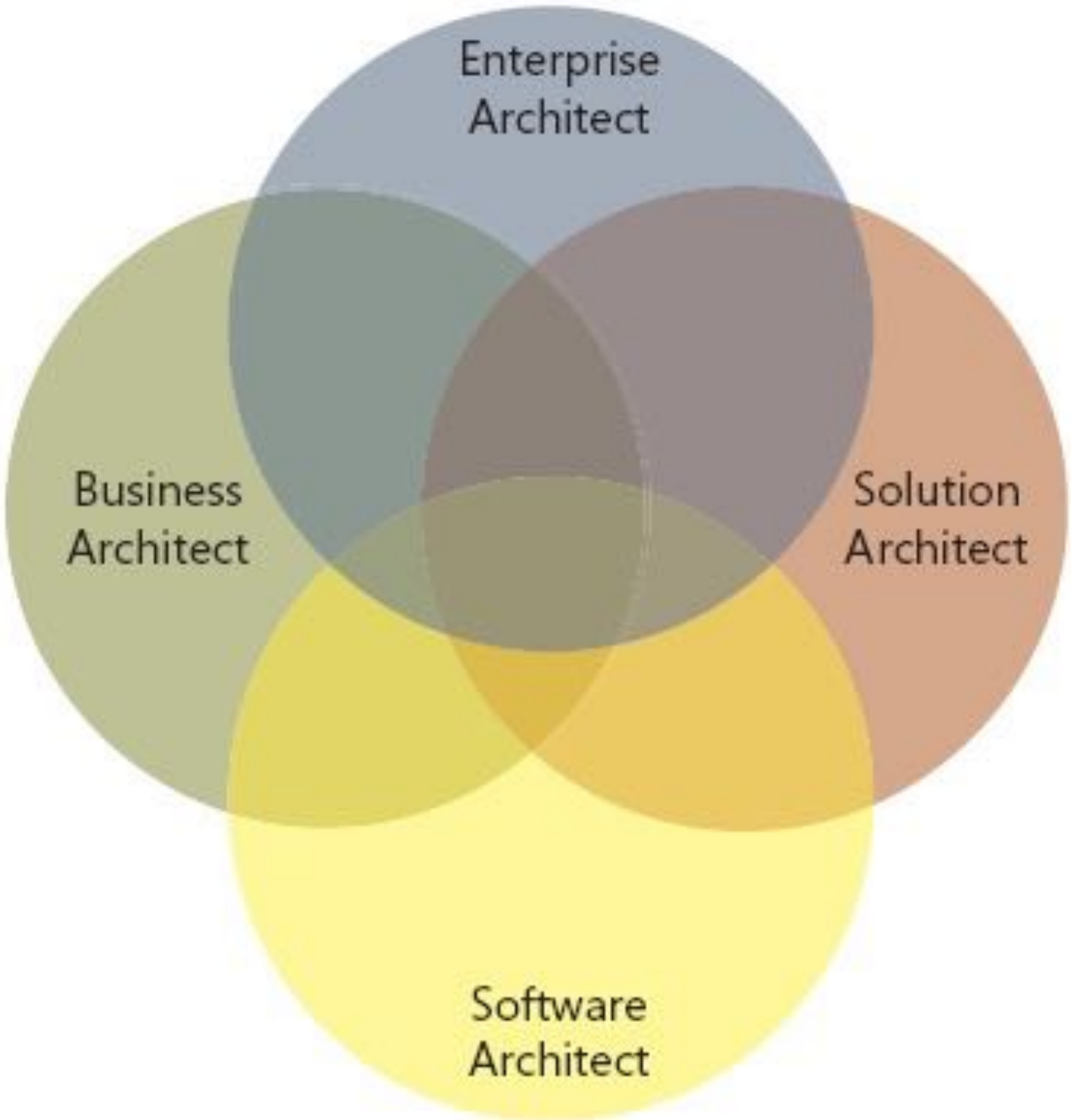
Code Maintainability

System quality

Detail Design
Module Design
Class Design
Low Level Design
Code design
Implementation Design

Types of Architect

Architect Roles



Product Architect

Solution Architect

Enterprise Architect

TOGAF
DODAF
Zachman

Align

Application Architect

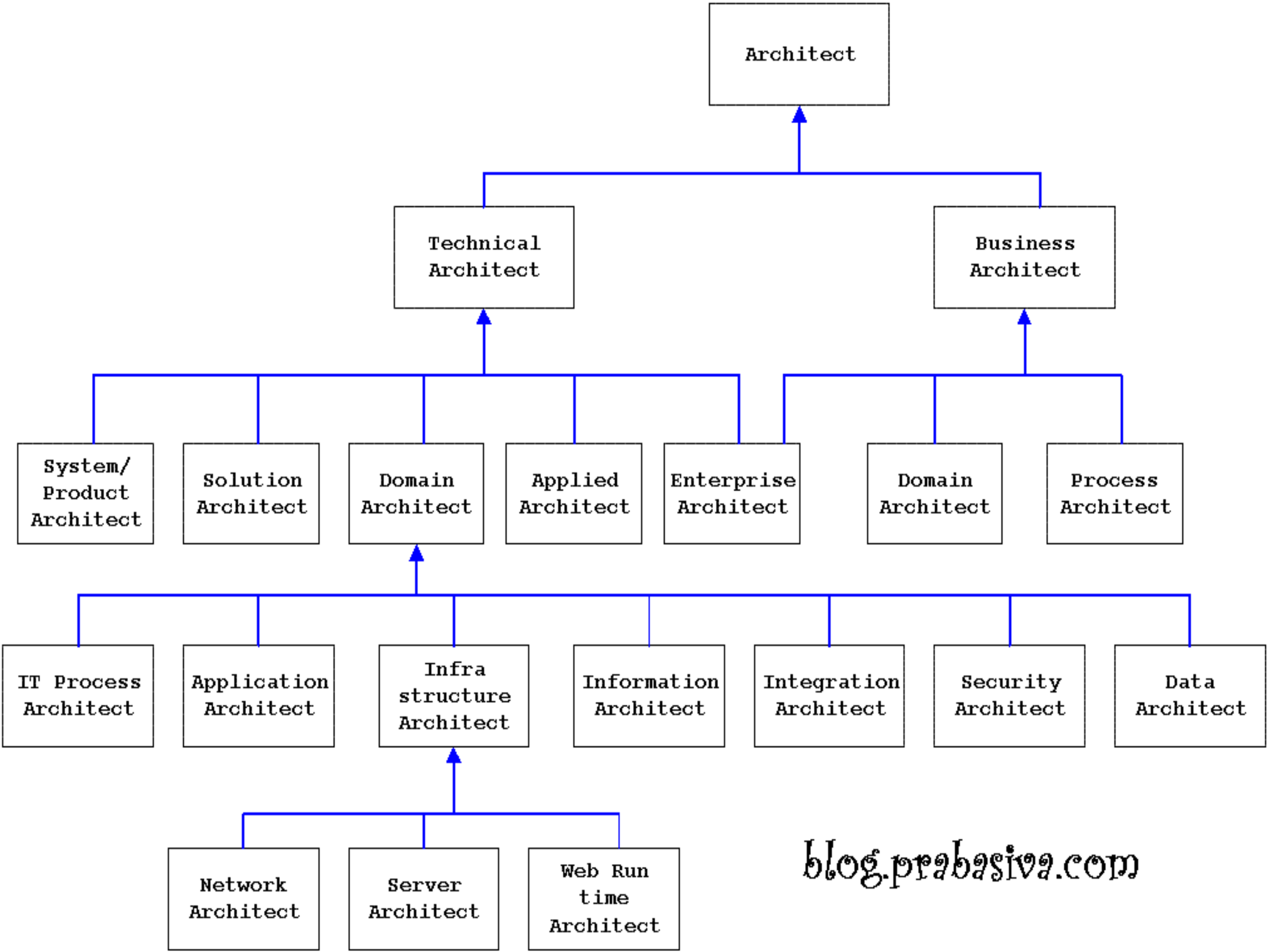
Quality

Vertical Architect

Domain Architect

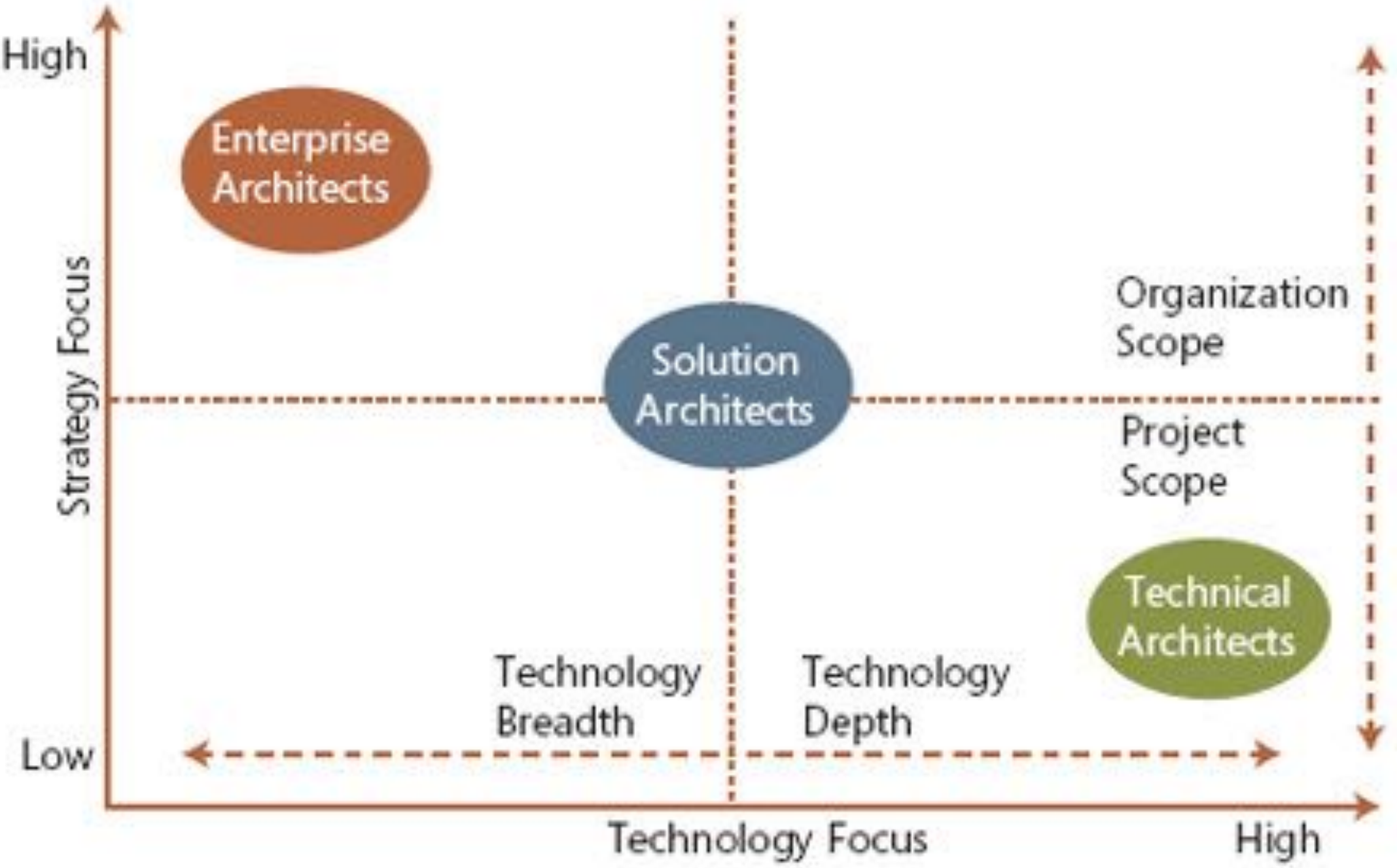
BPM

Quality


Architect
Technical Architect
Business Architect
System/ Product Architect
Solution Architect
Domain Architect
Applied Architect
Enterprise Architect
Domain Architect
Process Architect
IT Process Architect
Application Architect
Infra structure Architect
Information Architect
Integration Architect
Security Architect
Data Architect
Network Architect
Server Architect
Web Run time Architect
blog.prabasiva.com

	Business	Enterprise	Solution	Application
Soft Skills				
Communicates well with key stakeholders (C-level officers)	#	*	#	#
Communicates well with stakeholders (Director / Manager level)	*	*	*	#
Public Speaking	~	*	~	#
Writing Skills	*	*	~	#
Hard Skills				
Business Process Engineering	*	*	~	#
Programming	#	#	*	*
Requirements Analysis	*	~	~	#
Software Design	#	~	*	*
Scope	Process	Organization	Single Solution	Single Project

Solution Architect

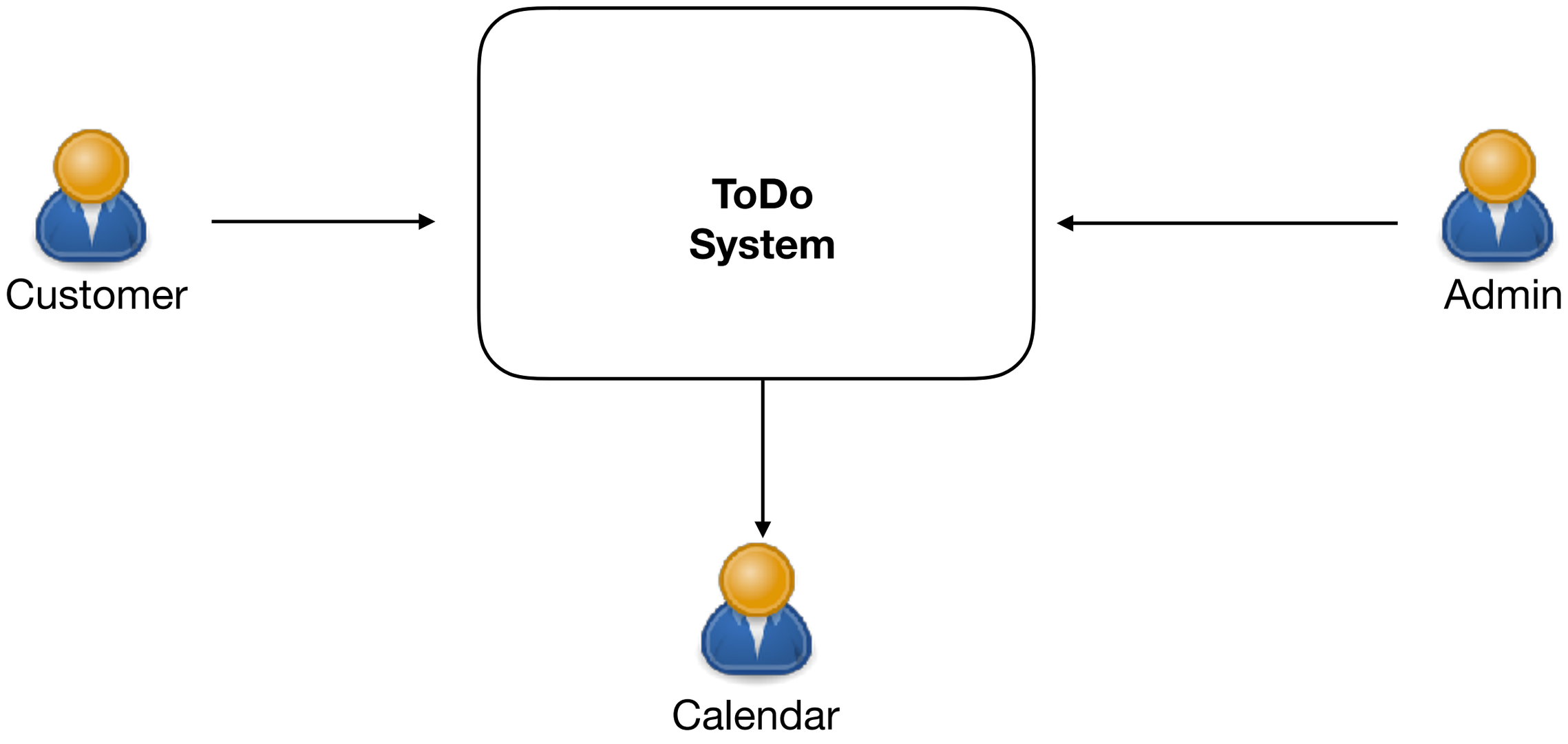


Part II

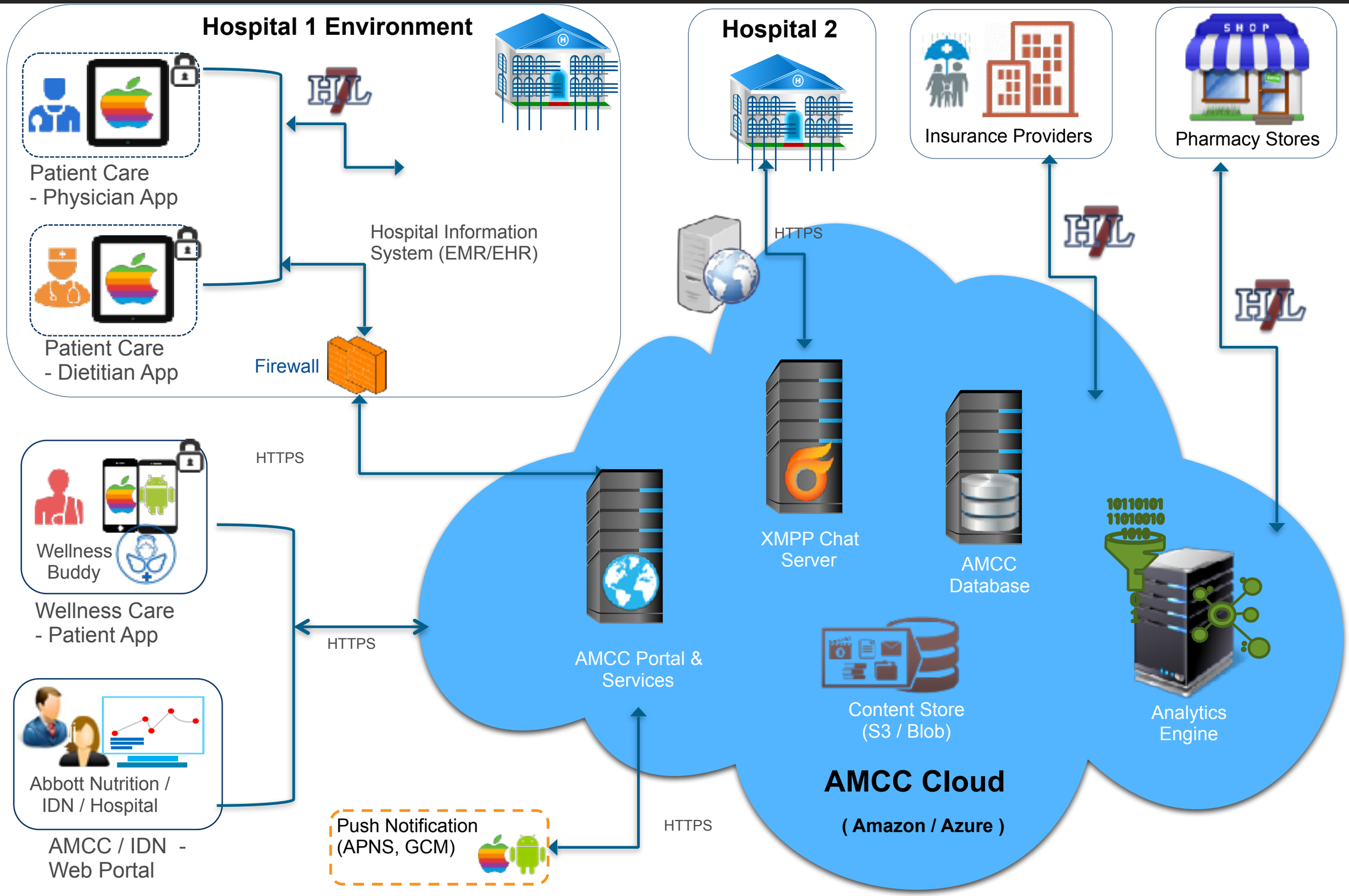
Collect Architectural Requirements
Understanding Quality of Service -
What is Quality Model
Quality Attribute scenarios
Quality Attribute Workshop (QAW)
Identification of architectural drivers
Implicit and explicit characteristics
Gathering Performance scenarios
Gathering Maintainability scenarios
Lab : Architecture driven requirements engineering using QAW

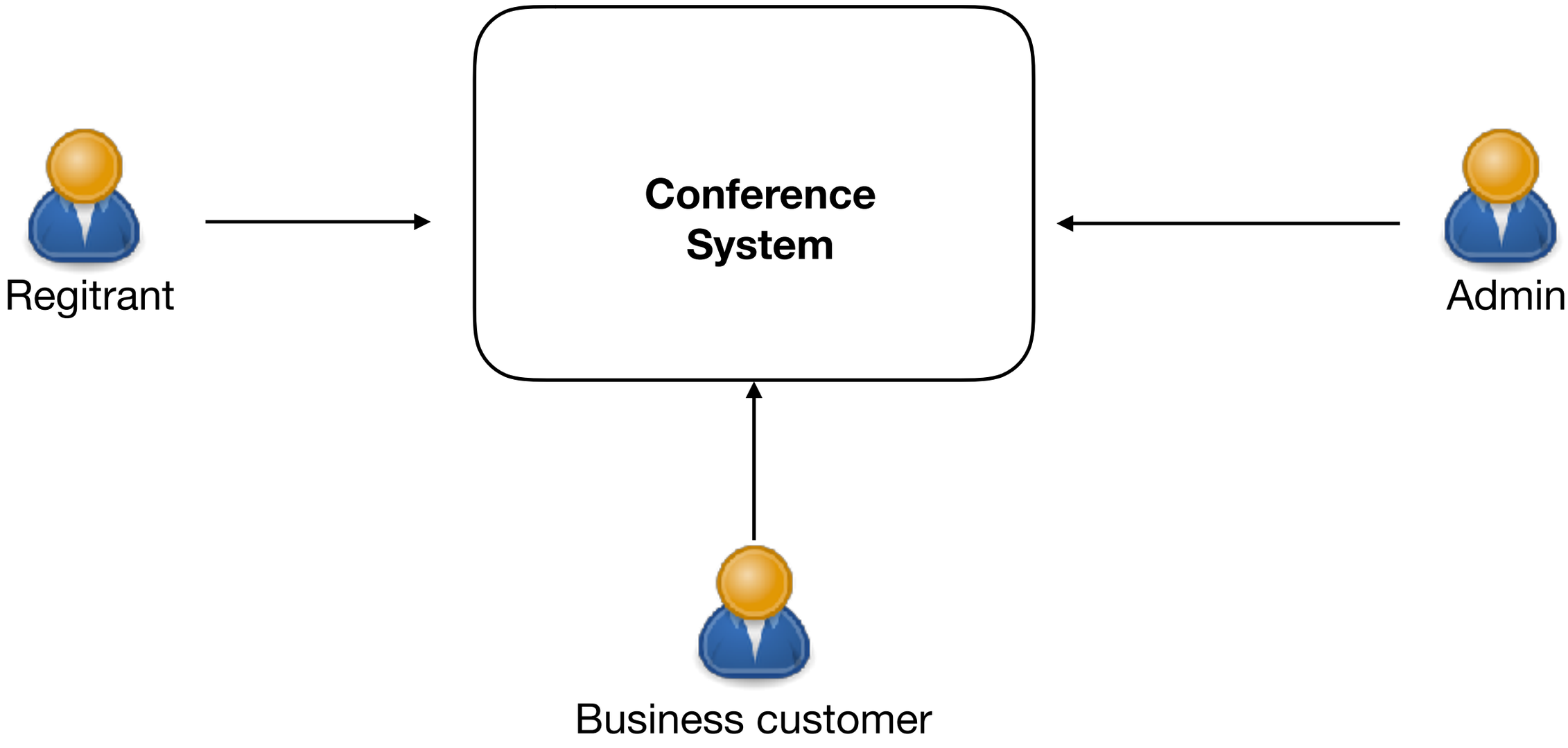
Arch Req

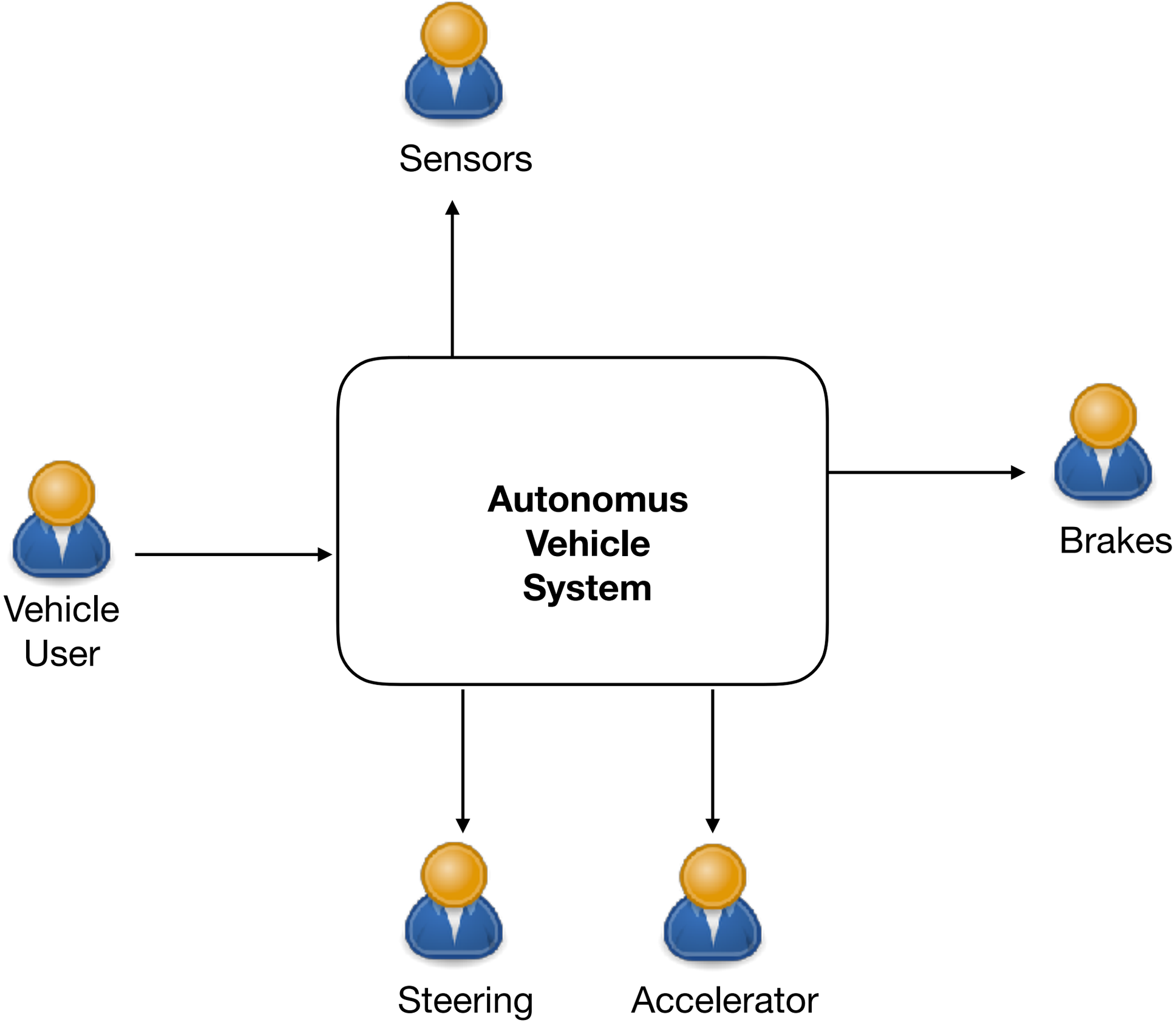
1. Context view

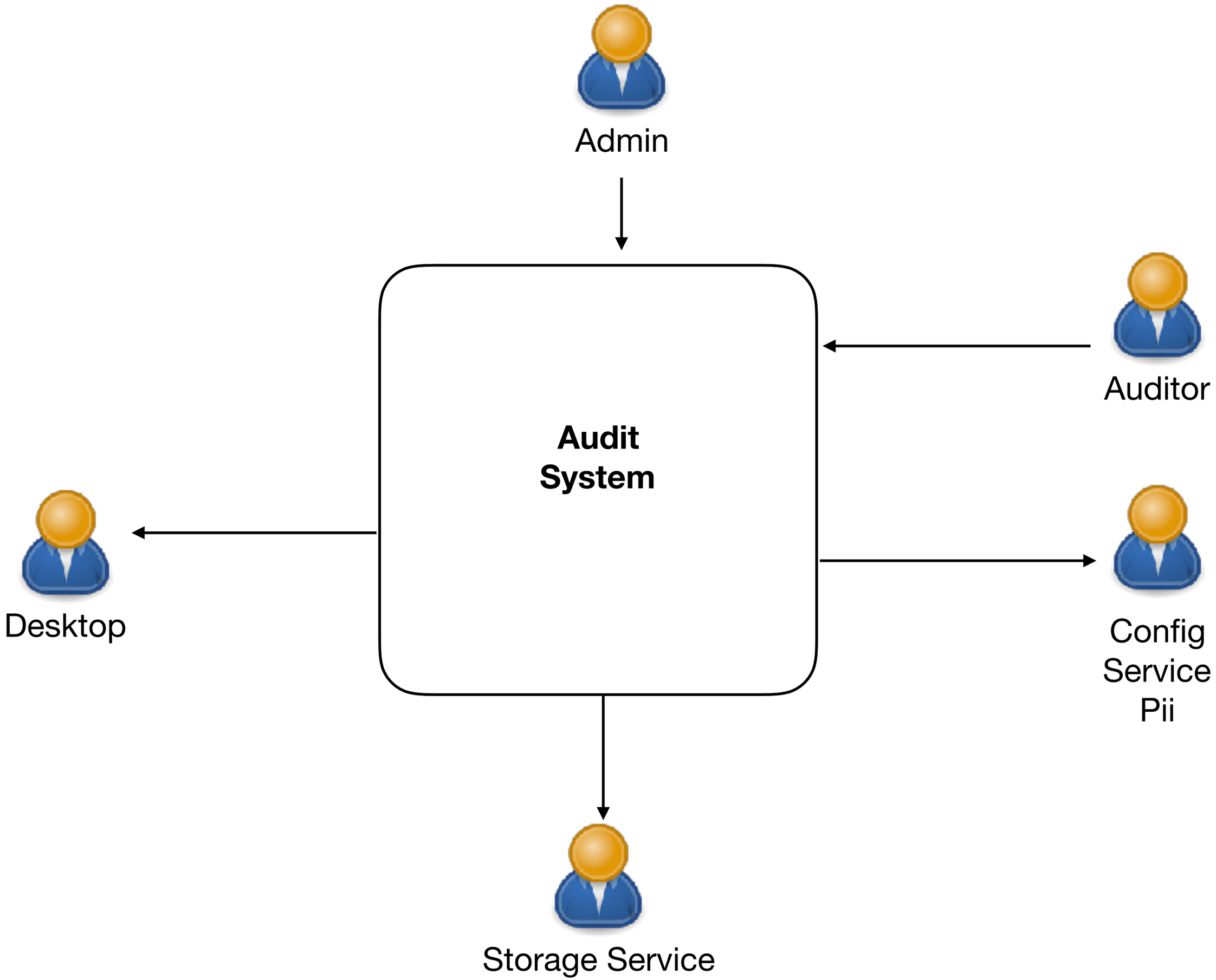


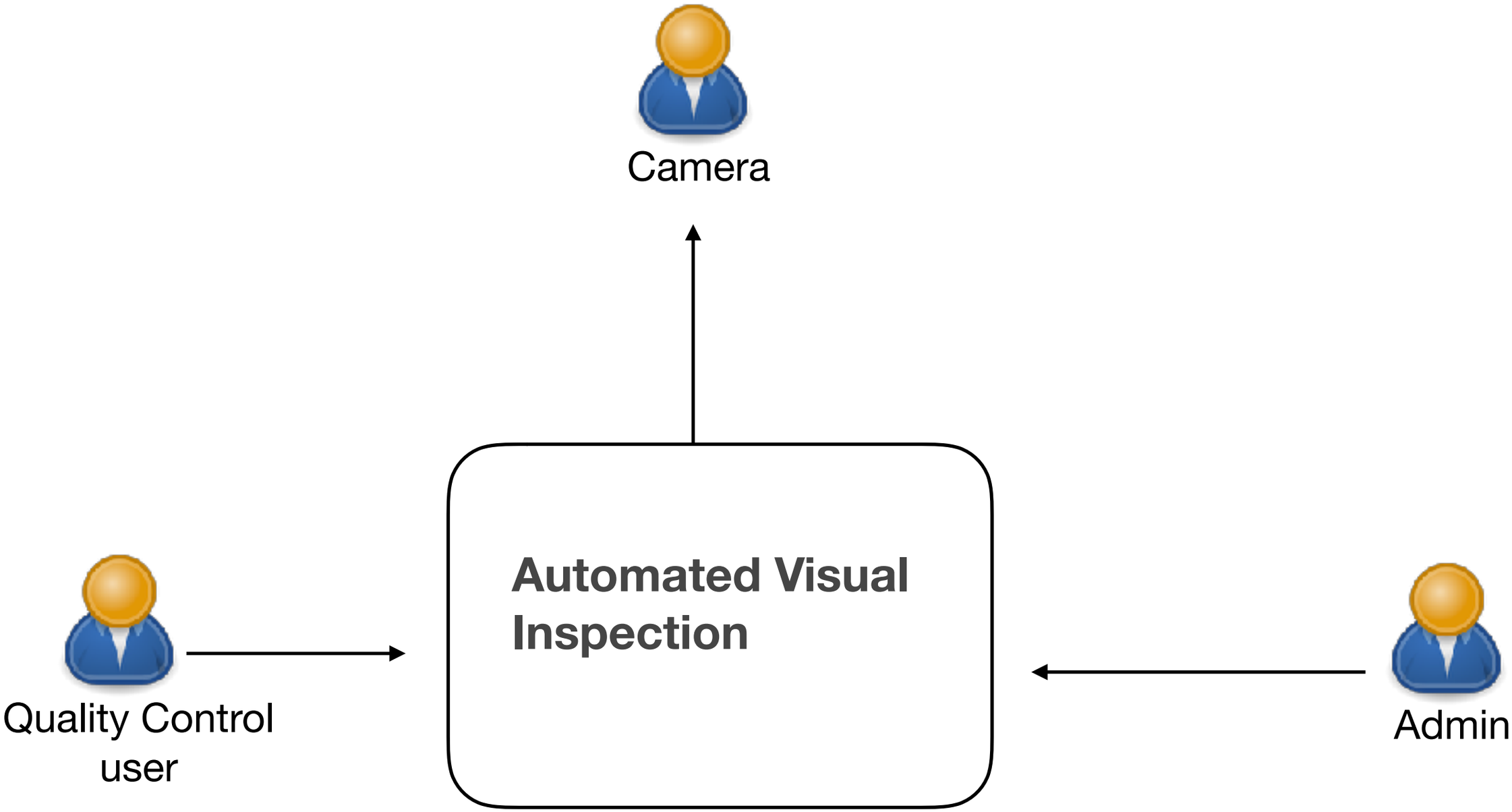
ONS Digital Health Solution – High Level Solution Architecture



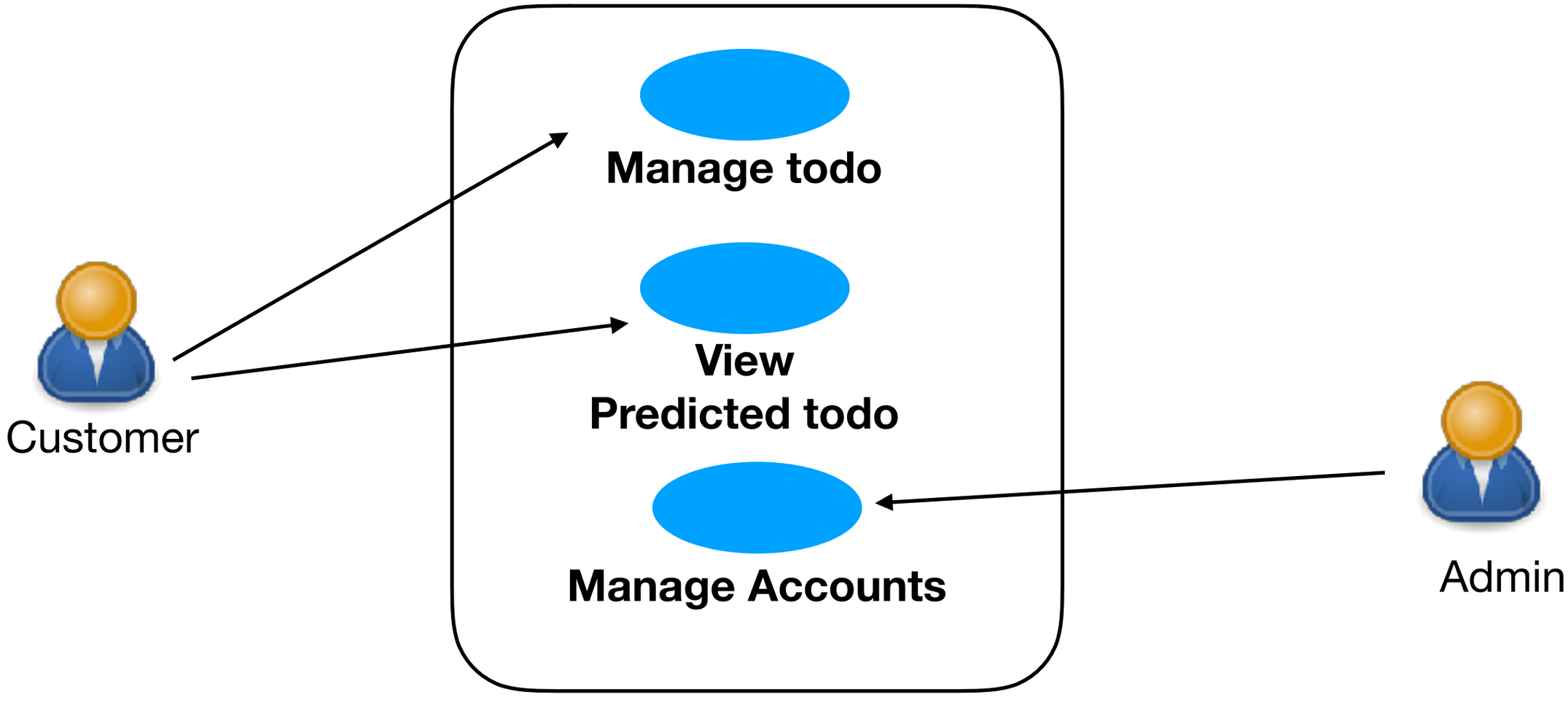

Conference System
Regitrant
Admin
Business customer

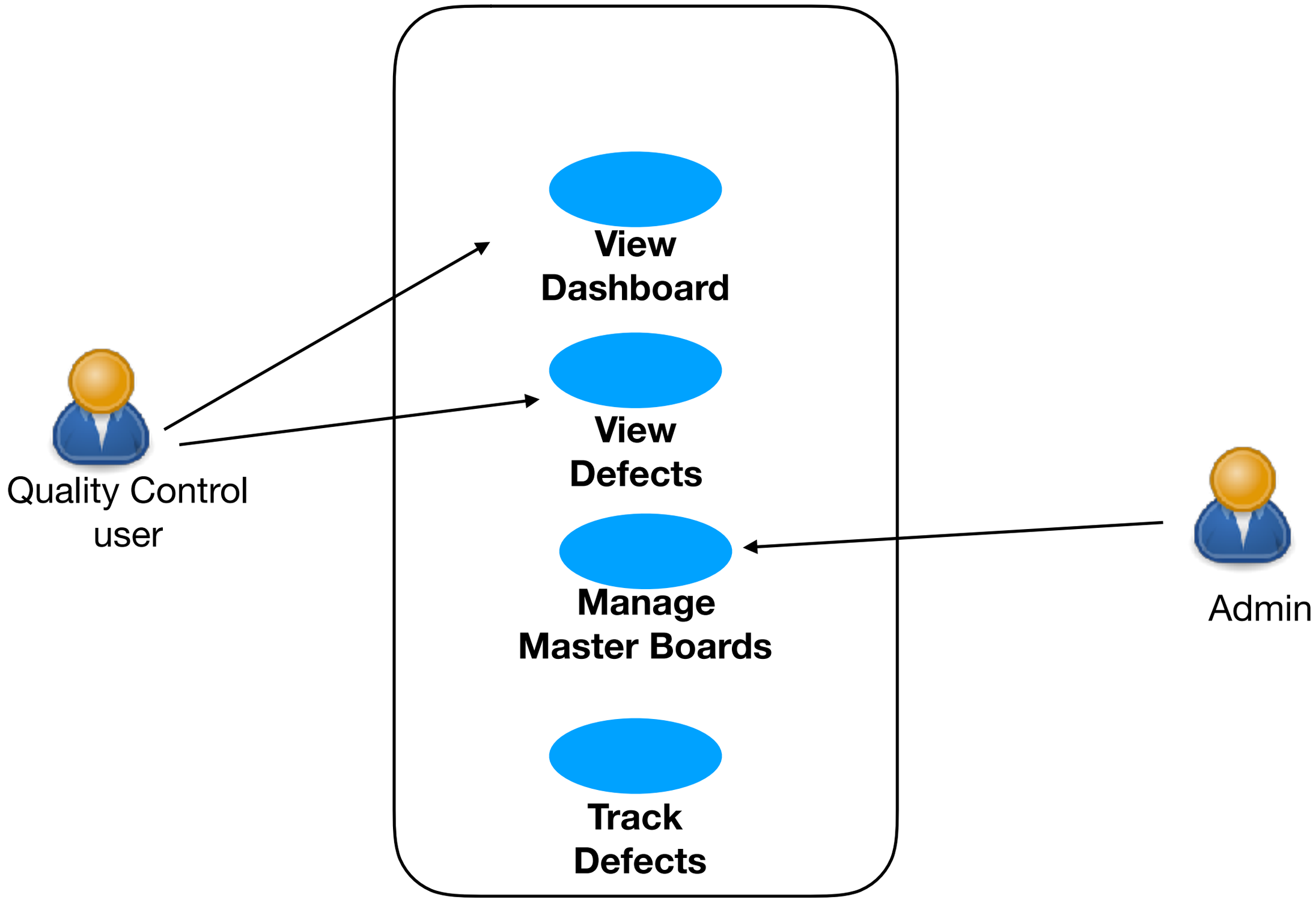

Sensors
Vehicle User
Autonomus Vehicle System
Brakes
Steering
Accelerator

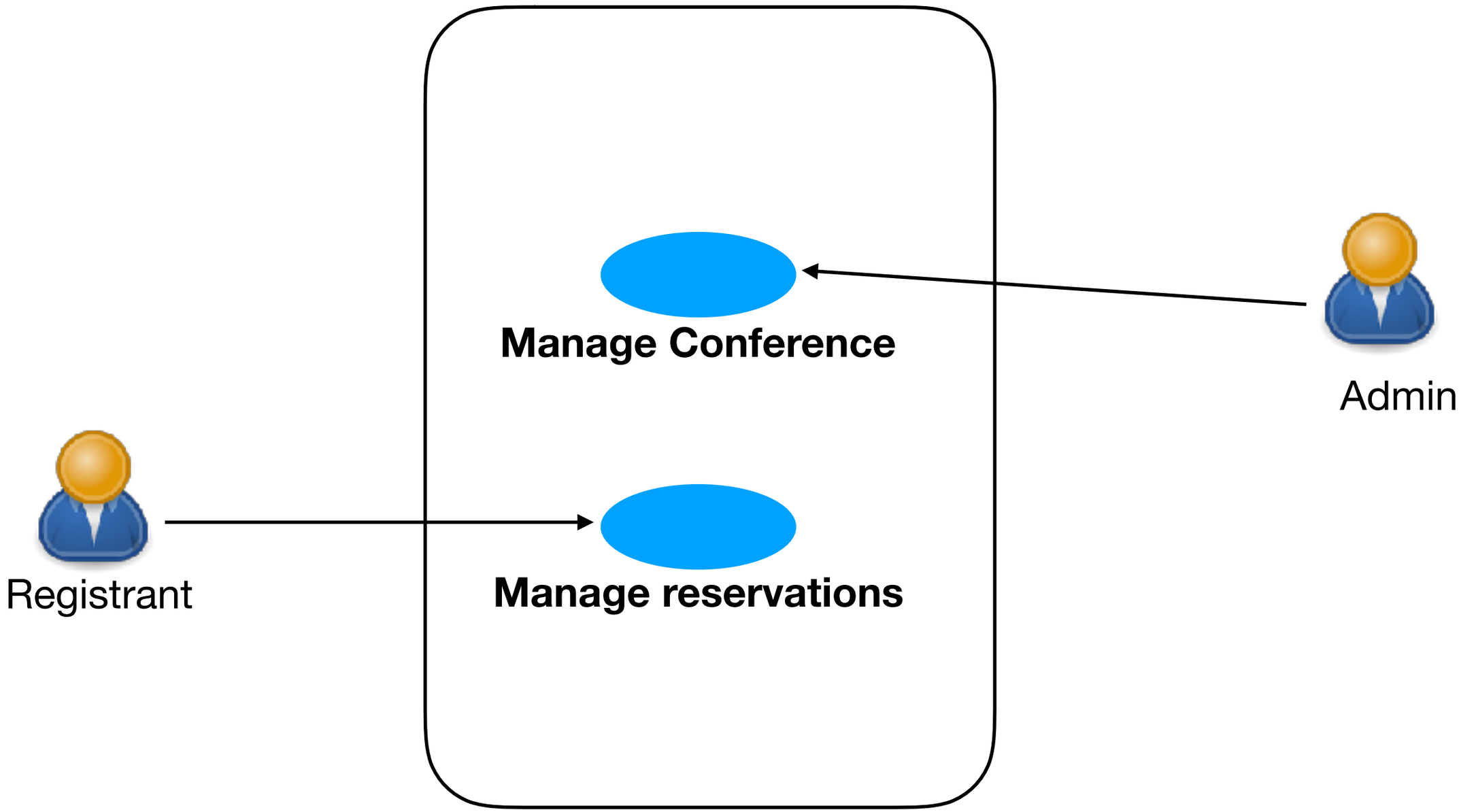

Admin
Auditor
Audit
System
Desktop
Config
Service
Pii
Storage Service

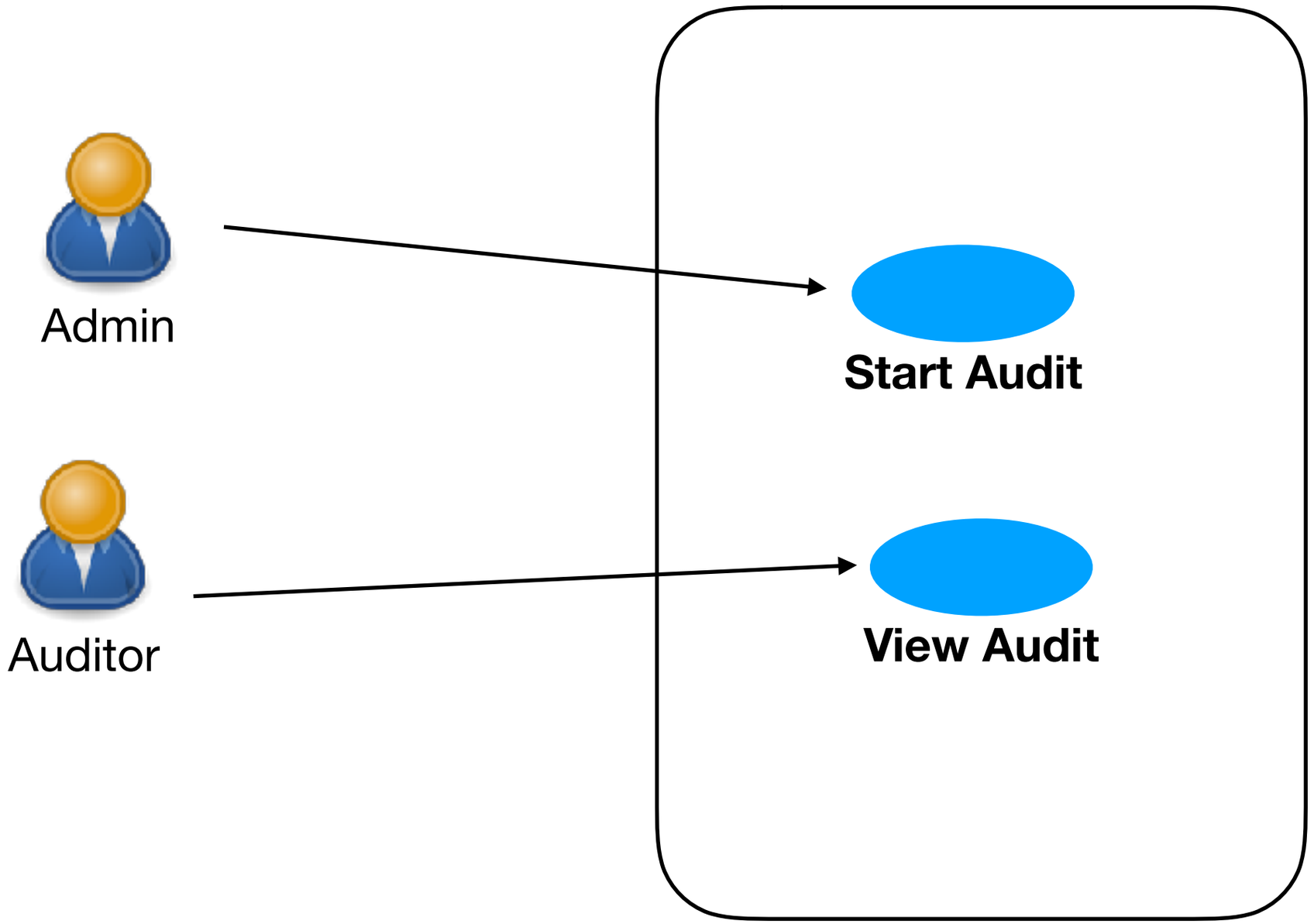

Camera
Automated Visual Inspection
Quality Control user
Admin

2. Functional view

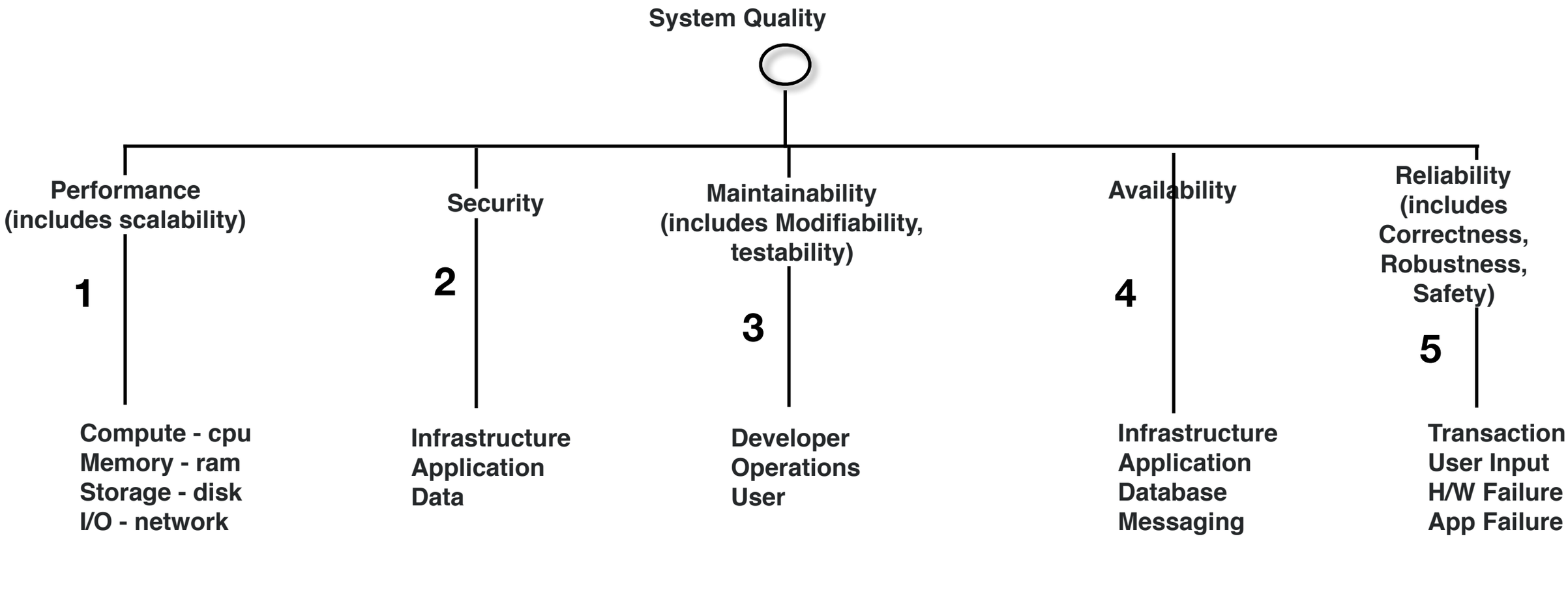



View
Dashboard
View
Defects
Manage
Master Boards
Track
Defects
Quality Control
user
Admin


Manage Conference
Admin
Registrant
Manage reservations


Admin
Start Audit
Auditor
View Audit

3. Quality View



Usability
Portability
Interoperability

Quality Model

- IEEE
- ISO
- MacCal
- Bohem
- ..

As a user I want to **add a todo** In the app

As a user I want to **add a todo** In the app . The todo is added In **< 3 secs**

Quality Attribute Scenarios

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a user	I want to add a todo	In the portal	During peak load.	The todo is added	In < 3 secs
Reliability	As a user	I want to add a todo	From the portal	Duplicate request	Duplicate entry is detected	In 100% of cases
Maintainability	Developer	Change the Configuration engine	In the Frame work	During maintenance	The configuration mechanism is replaced	In < 1 week
Security	unknown identity	requests to add a new todo	In the portal	During Normal load.	block access to the data and record the access attempts	100% probability of detecting the attack, 100% probability of denying access
Availability	The processor	Failed	In the db server	During Operational Hours	Secondary is made Primary	In < 5 minutes

BA, product owners, …
Dev, QA, Ops

Seed scenarios
List of scenarios (Today | growth | exploratory)

Source: Processor
Stimulas: Stop working during Peak Hours
Artifact: in the central system
environment: during Peak Hours (8 am - 12 pm)
Response : Notification to operator, system Administrator
....
Measure: not more than 5 minutes to get a new processor running

A processor stops working during Peak Hours in the central system
during Peak Hours, Notification to operator, system Administrator
….. Should … not more than 5 minutes to get a new processor running

Performance

Scenario Portion	Possible Values
Source	Internal to the system, External to the system
Stimulus	Periodic events, sporadic events, Bursty events, stochastic events
Artifact	System, component
Environment	Normal mode; overload mode, Reduced capacity mode, Emergency mode, Peak mode
Response	Process stimuli; change level of service
RespMeasure	Latency, deadline, throughput, jitter, miss rate, data loss

source - “Software architecture in Practice

1. What is the expected response time for each use case?
2. What is the average/max/min expected response time?
3. What resources are being used (CPU, LAN, etc.)?
4. What is the resource consumption?
5. What is the resource arbitration policy?
6. What is the expected number of concurrent sessions?
7. Are there any particularly long computations that occur when a certain state is true?
8. Are server processes single or multithreaded?
9. Is there sufficient network bandwidth to all nodes on the network?
10. Are there multiple threads or processes accessing a shared resource? How is access managed?
11. Will bad performance dramatically affect usability?
12. Is the response time synchronous or asynchronous?
13. What is the expected batch cycle time?
14. How much can performance vary based on the time of day, week, month, or system load?
15. What is the expected growth of system load?
16. Increased network bandwidth consumption?
17. Increased database server processing, resulting in reduced throughput.
18. Increased memory consumption, excessive cache misses
19. Increased client response time, reduced throughput, and server resource over utilization.

source - msdn

Modifiability

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	Wishes to add/delete/modify/vary functionality, quality attribute, capacity or technology
Artifact	Code, Data, Components, Configurations, Interfaces, Resources
Environment	At runtime, compile time, build time, design time
Response	Locates places in architecture to be modified; makes modification without affecting other functionality; tests modification; deploys modification
Response Measure	Cost in effort, Cost in money, Cost in time,Cost in number, size, complexity of affected artifacts, Extent affects other system functions or qualities, New defects introduce

source - “Software architecture in Practice

1. Is this system the start of a new product line?
2. Will other systems be built that more or less match the characteristics of the system under construction? If so, what components will be reused in those systems?
3. What Changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements ?
4. Does business rule values change frequently ?
5. Does Business process change frequently ?
6. What existing components are available for reuse?
7. Are there existing frameworks or other code assets that can be reused?
8. Will other applications reuse the technical infrastructure that is created for this application?
9. What are the associated costs, risks, and benefits of building reusable components?
10. Excessive dependencies between components and layers
11. The use of direct communication prevents changes to the physical deployment of components and layers.
12. Reliance on custom implementations of features such as authentication and authorization prevents reuse.
13. The logic code of components and segments is not cohesive.
14. The code base is large, unmanageable, fragile, or over complex.
15. The existing code does not have an automated regression test suite.
16. Lack of documentation may hinder usage, management, and future upgrades.

source - msdn

Testability

Scenario Portion	Possible Values
Source	Unit tester, Integration tester, System tester, Acceptance tester, End user, Automated testing tools
Stimulus	Analysis, architecture, design, class, subsystem integration, system delivered
Artifact	Portion of the system being tested(Piece of design, piece of code, complete system)
Environment	Design time, Development time, Compile time, Integration time, Deployment time, Run time
Response	Execute test suite & capture results, Capture cause of fault, Control & monitor state of the system
RespMeasure	Effort to find fault, Effort to achieve coverage %,Probability of fault being revealed by next test, Time to perform tests, Effort to detect faults, Length of longest dependency chain, Time to prepare test environment, Reduction in risk exposure

source - “Software architecture in Practice

1. Are there tools, processes, and techniques in place to test language classes, components, and services?
2. Are there hooks in the frameworks to perform unit tests?
3. Can automated testing tools be used to test the system?
4. Can the system run in a debugger?
5. Complex applications with many processing permutations are not tested consistently, perhaps because automated or granular testing cannot be performed if the application has a monolithic design.
6. Lack of test planning.
7. Poor test coverage, for both manual and automated tests.
8. Input and output inconsistencies; for the same input, the output is not the same and the output does not fully cover the output domain even when all known variations of input are provided.

source - msdn

4. Constraints view

It should work on IE 11, Chrome and Firefox

Should work on windows, unix, Mac platforms

Should deploy on Azure Cloud

---

5. Assumptions View

1	Will use open source Technologies only
2	
3	

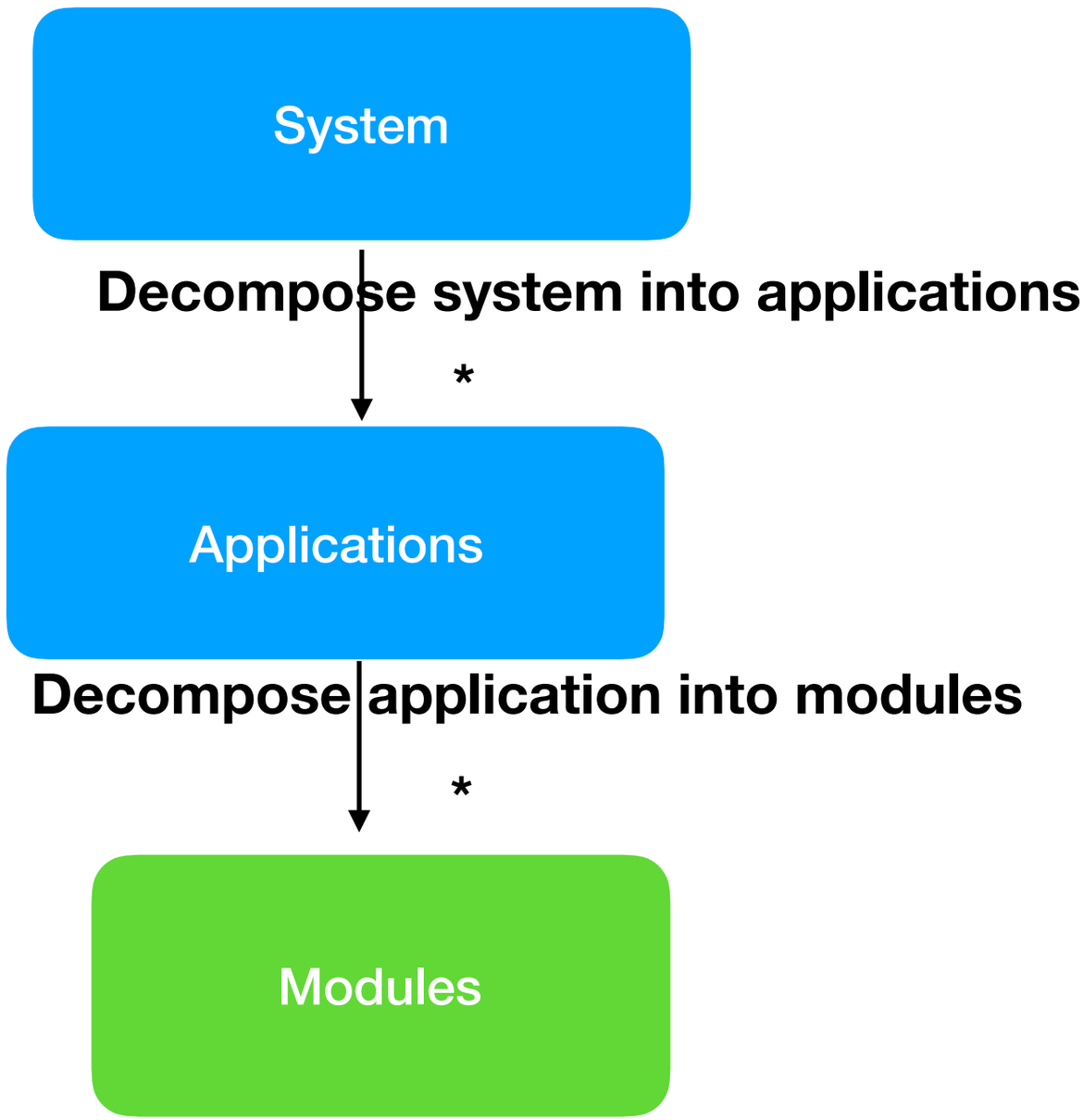
Part III

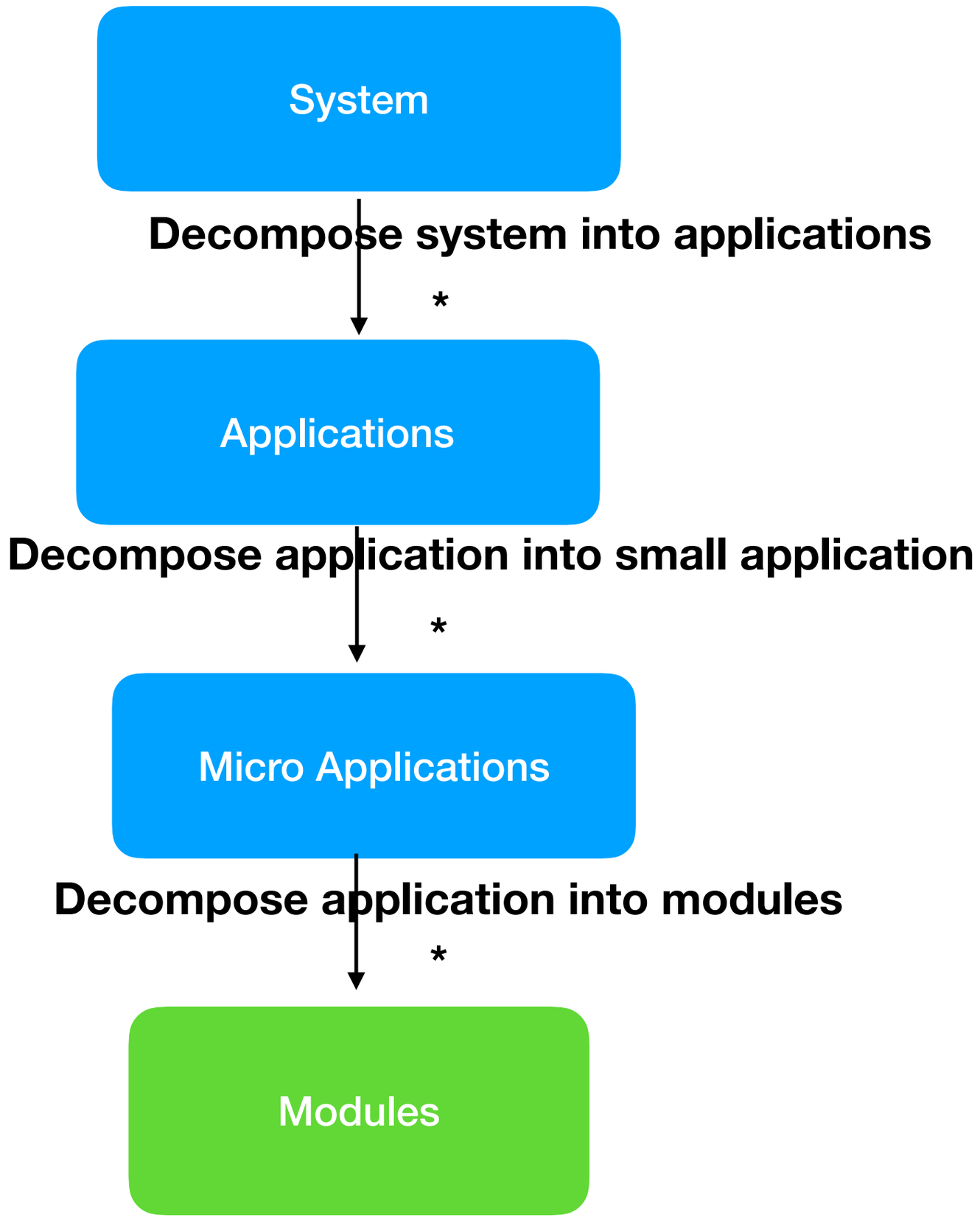
Building Architecture
Understanding Types of Applications
Client, Native, Web, background
Server, API, Backend
Brokered
Decomposing system into Applications
layered, Hexagonal
pipes and filters
Micro kernel
Component based
Micro service
Event driven Architecture
MVC, MVP, MVVM

Arch Design

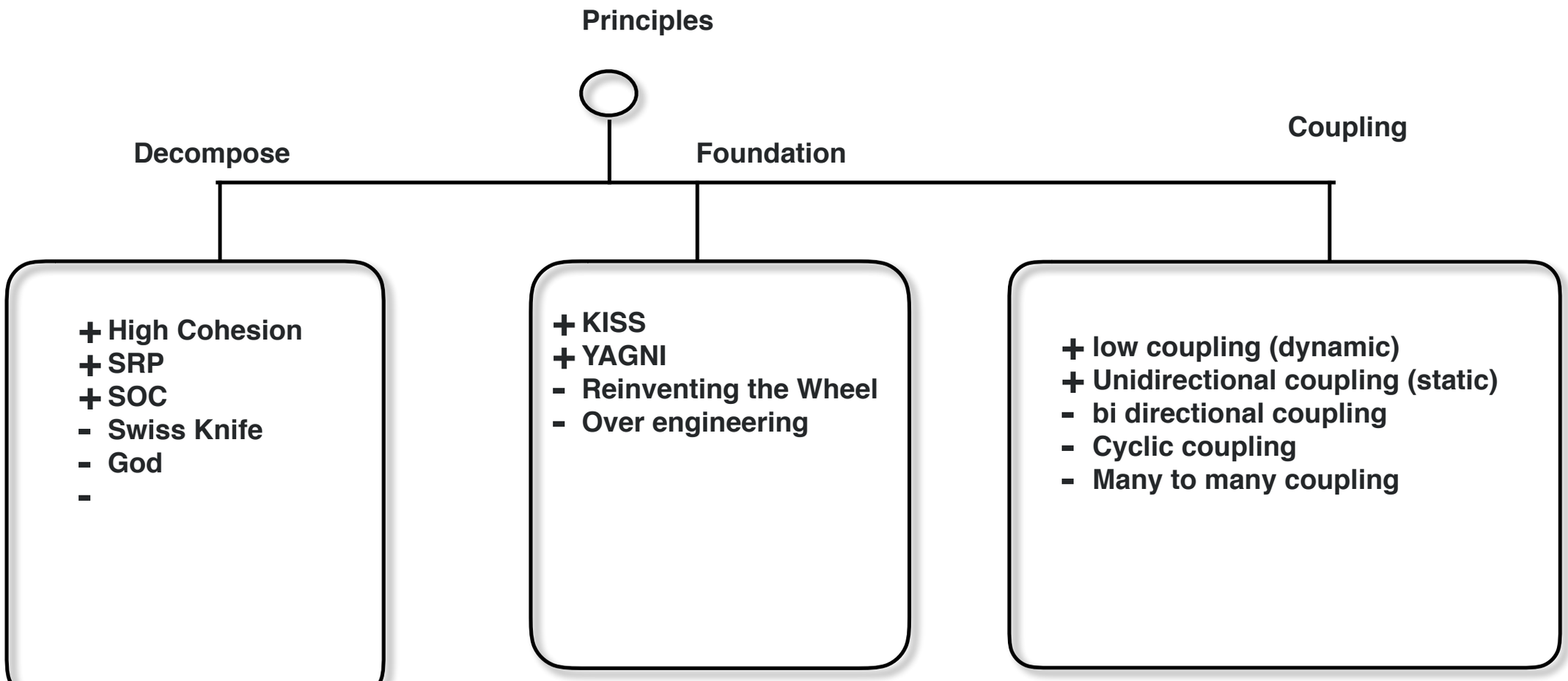
Decisions

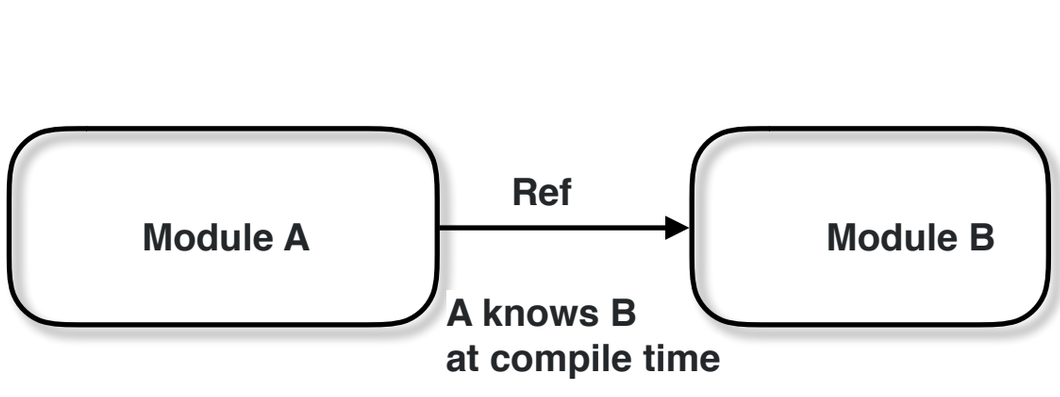
Logical View

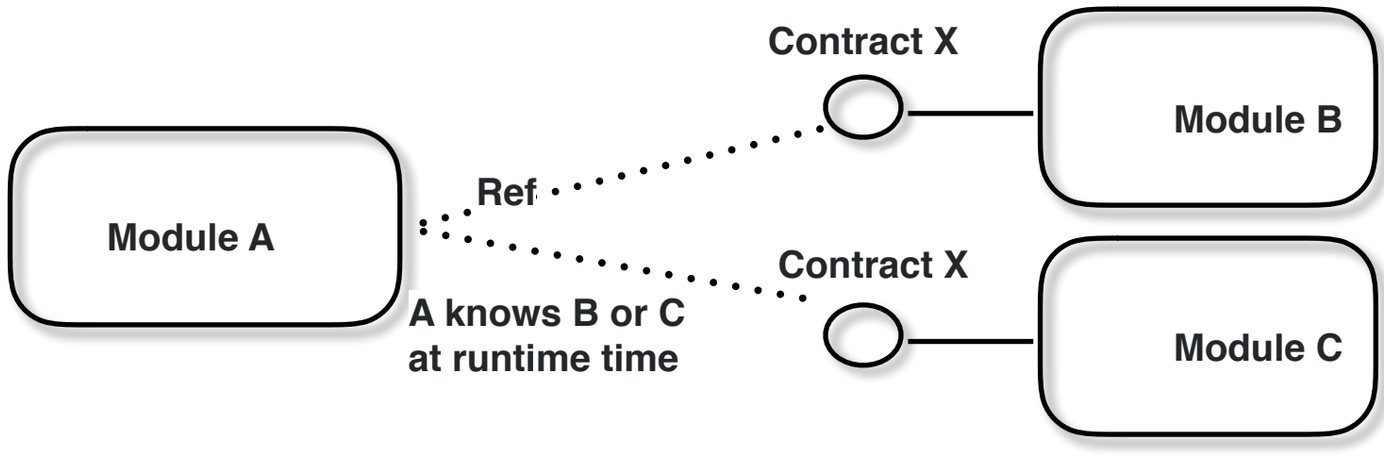


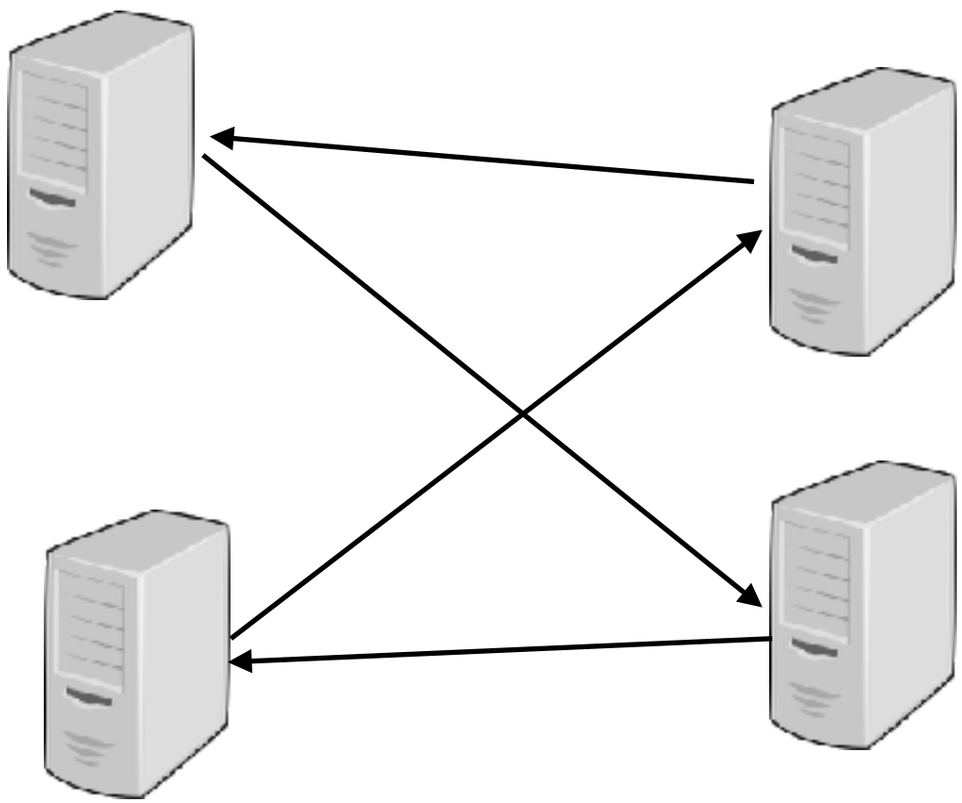

System
Decompose system into applications
*
Applications
Decompose application into small application
*
Micro Applications
Decompose application into modules
*
Modules

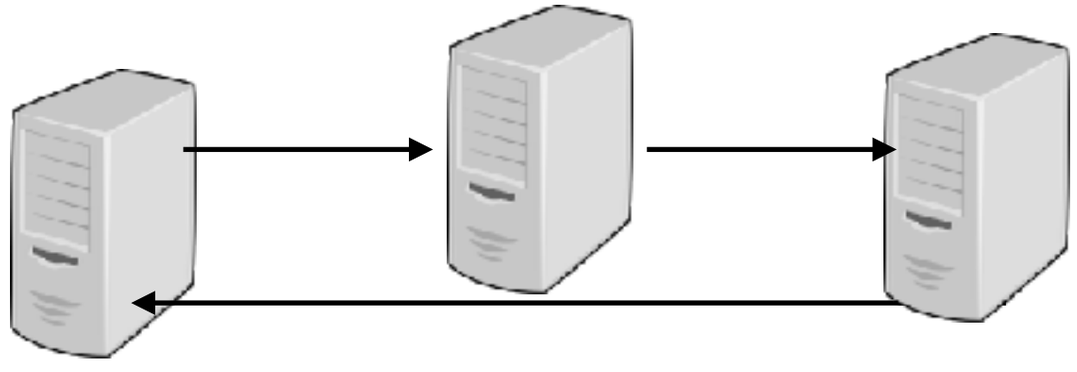
Architecture Principles



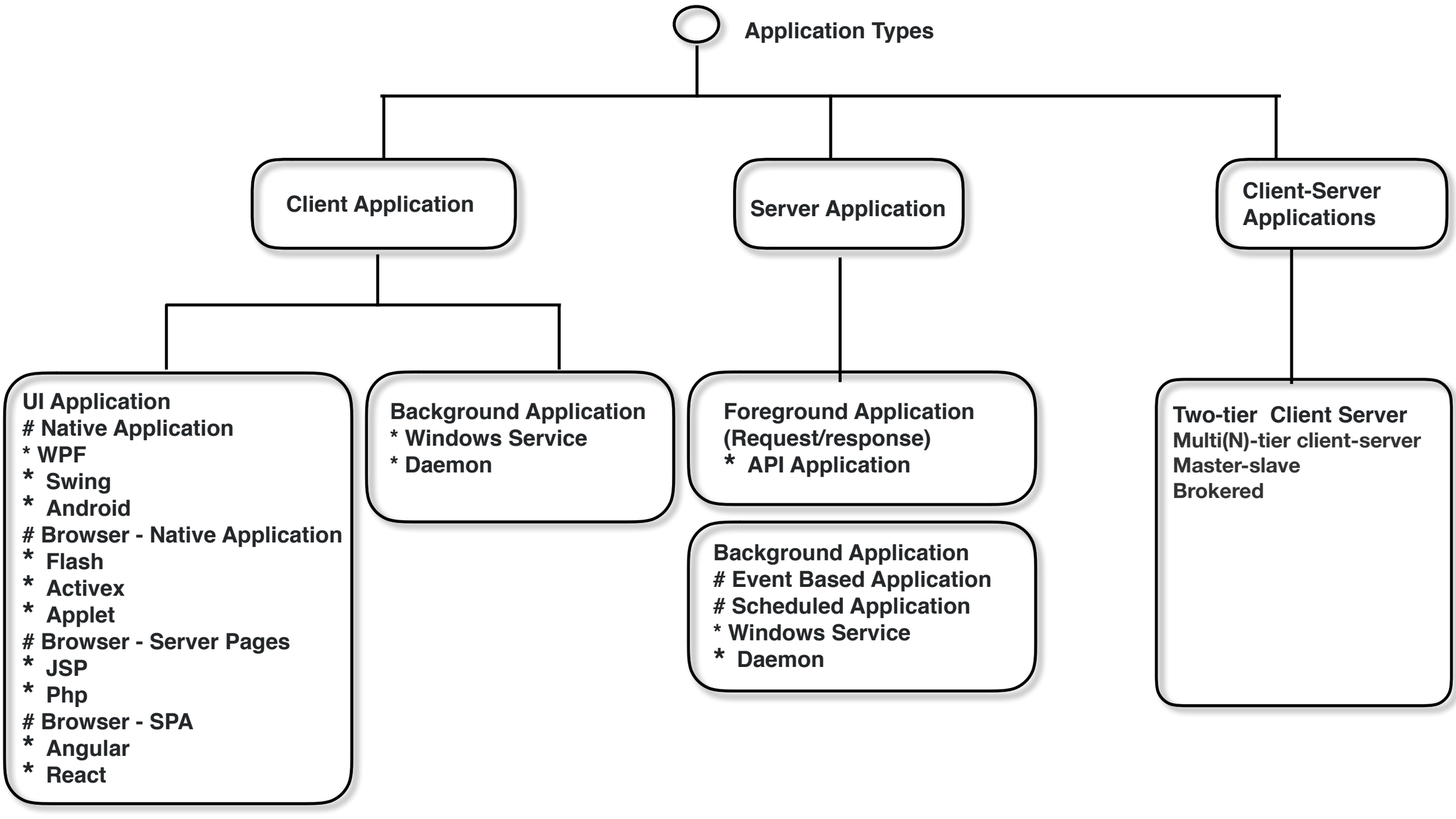


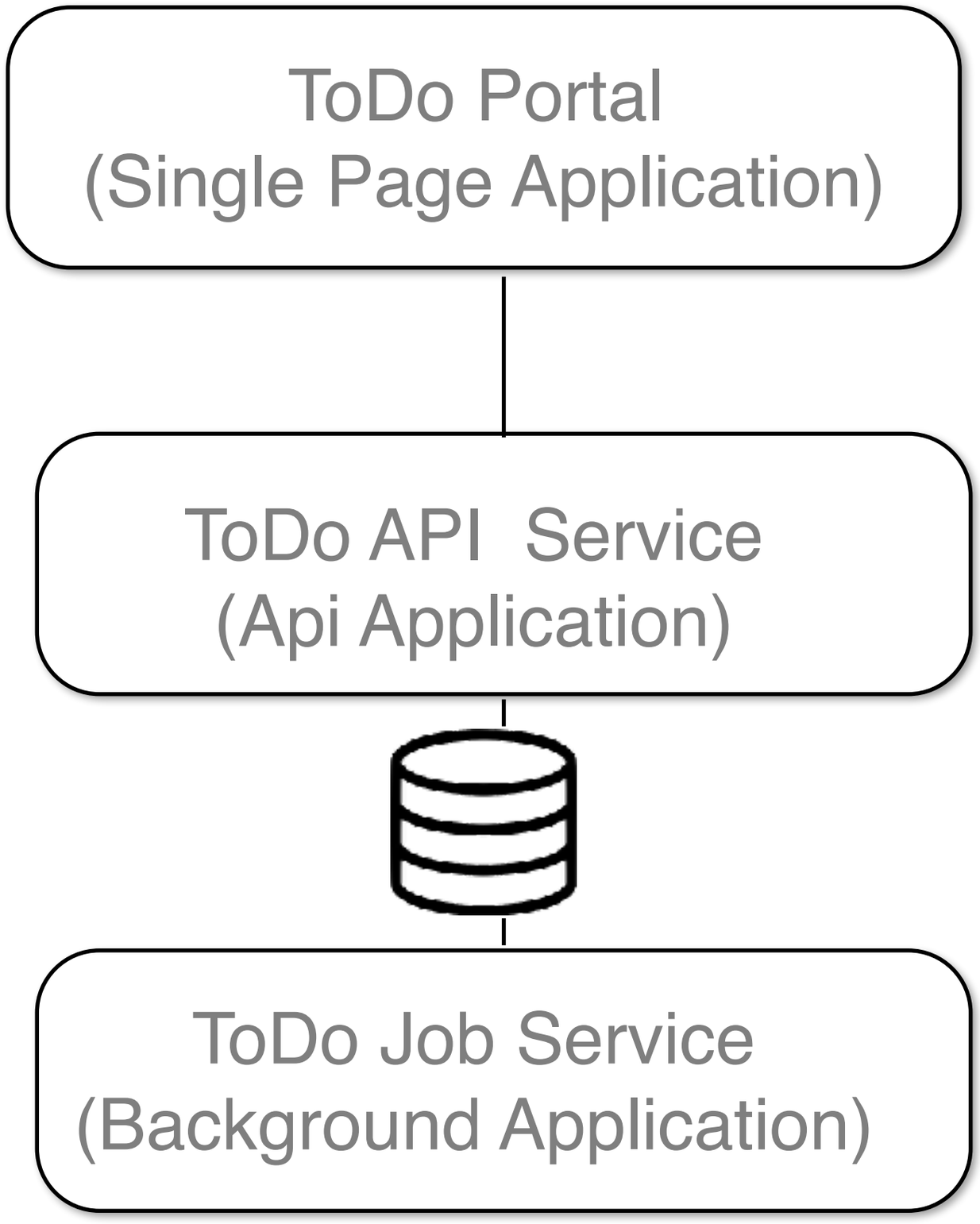


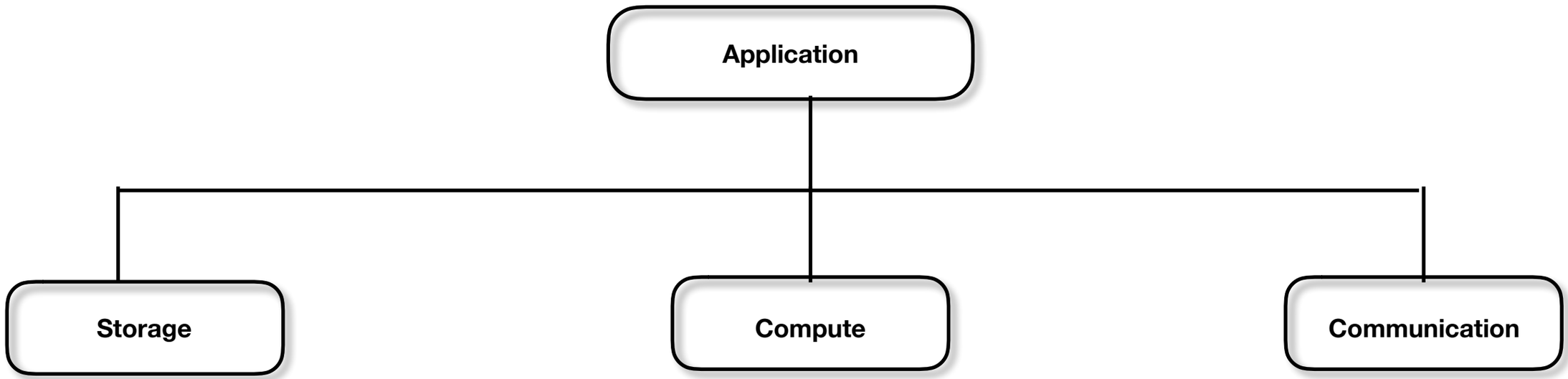




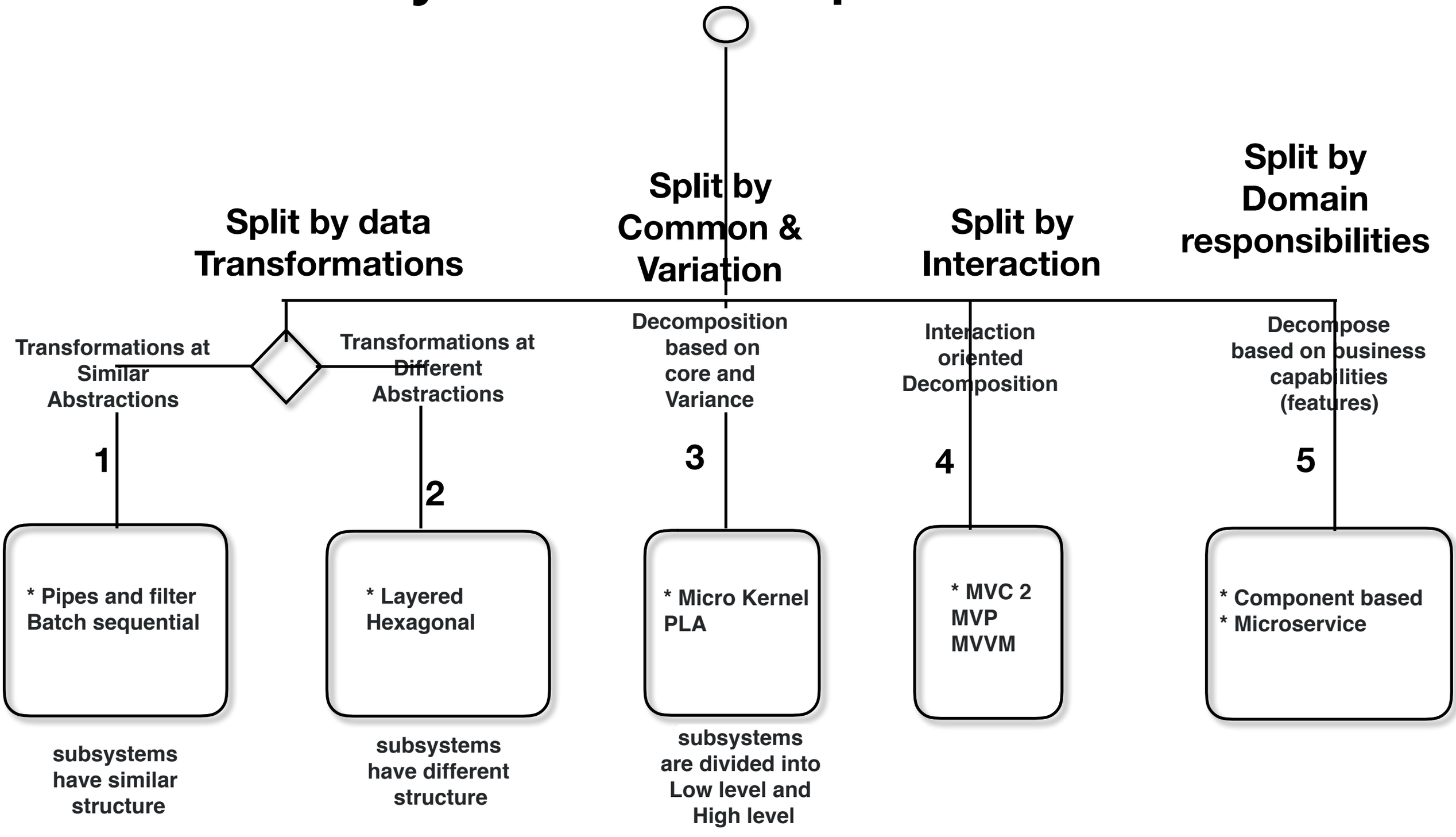
Step 1. Decompose the system into Applications/Tiers



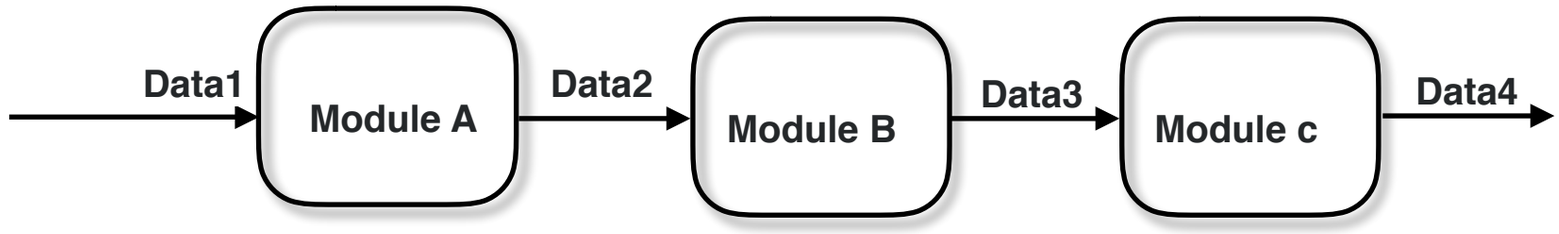

ToDo Portal
(Single Page Application)
ToDo API Service
(Api Application)
ToDo Job Service
(Background Application)


Application
Storage
Compute
Communication

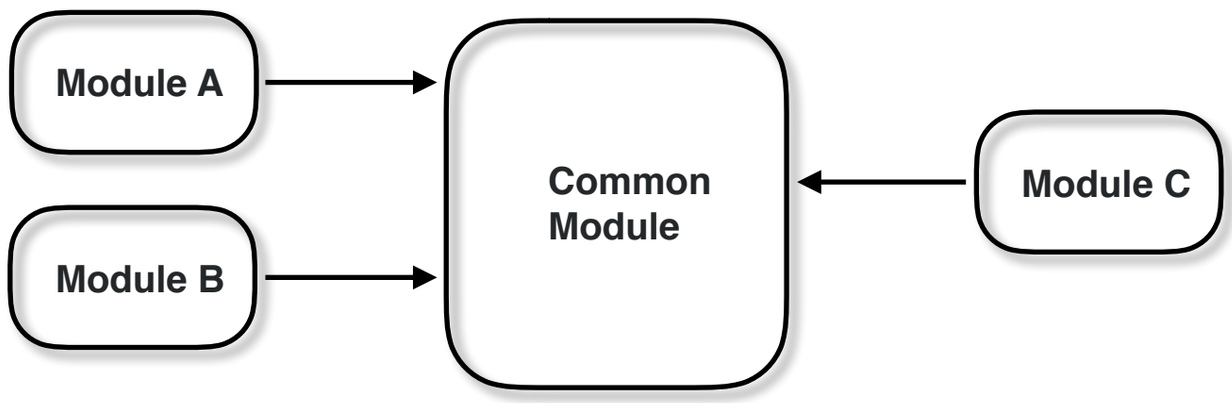
Choose System Decomposition Patterns



Actor *



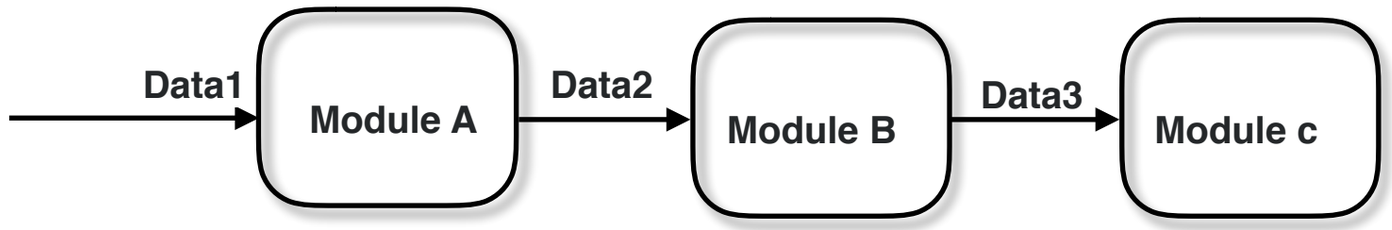
Split by data Transformations



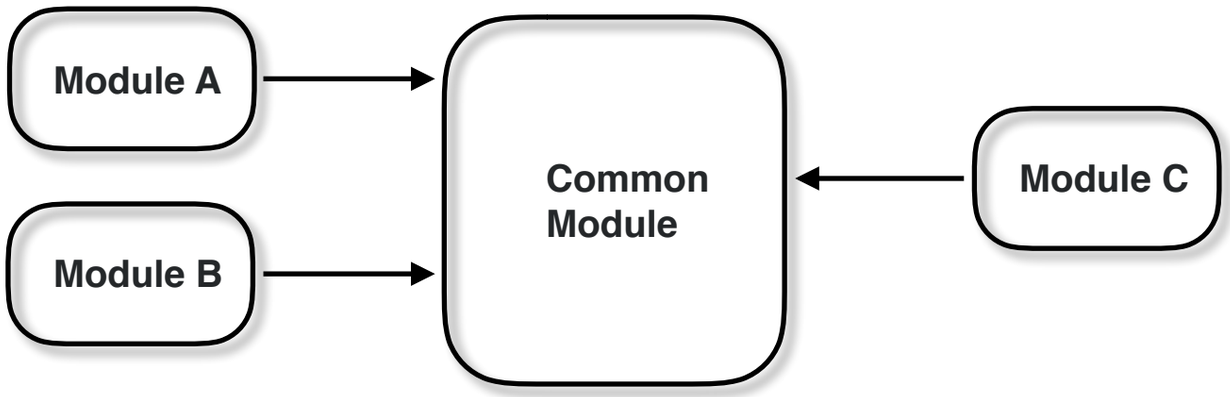
Split by Common & Variation

Split by Interaction

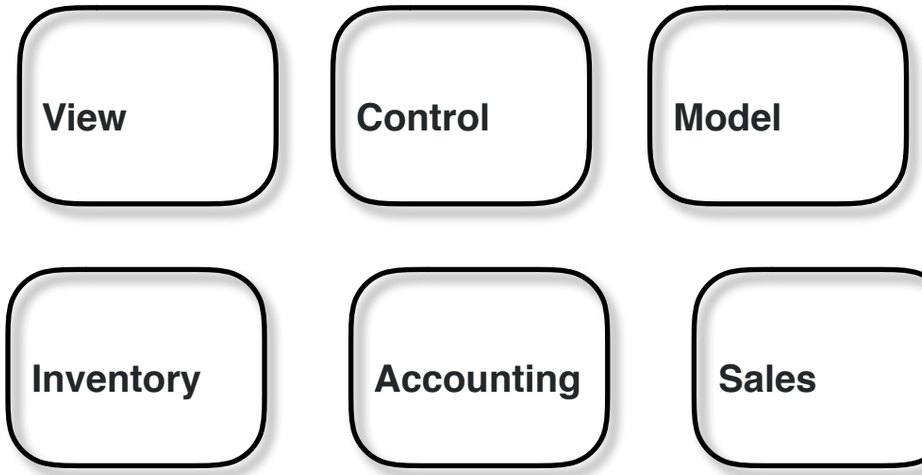
Split by Domain responsibilities



Split by data Transformations



Split by Common & Variation

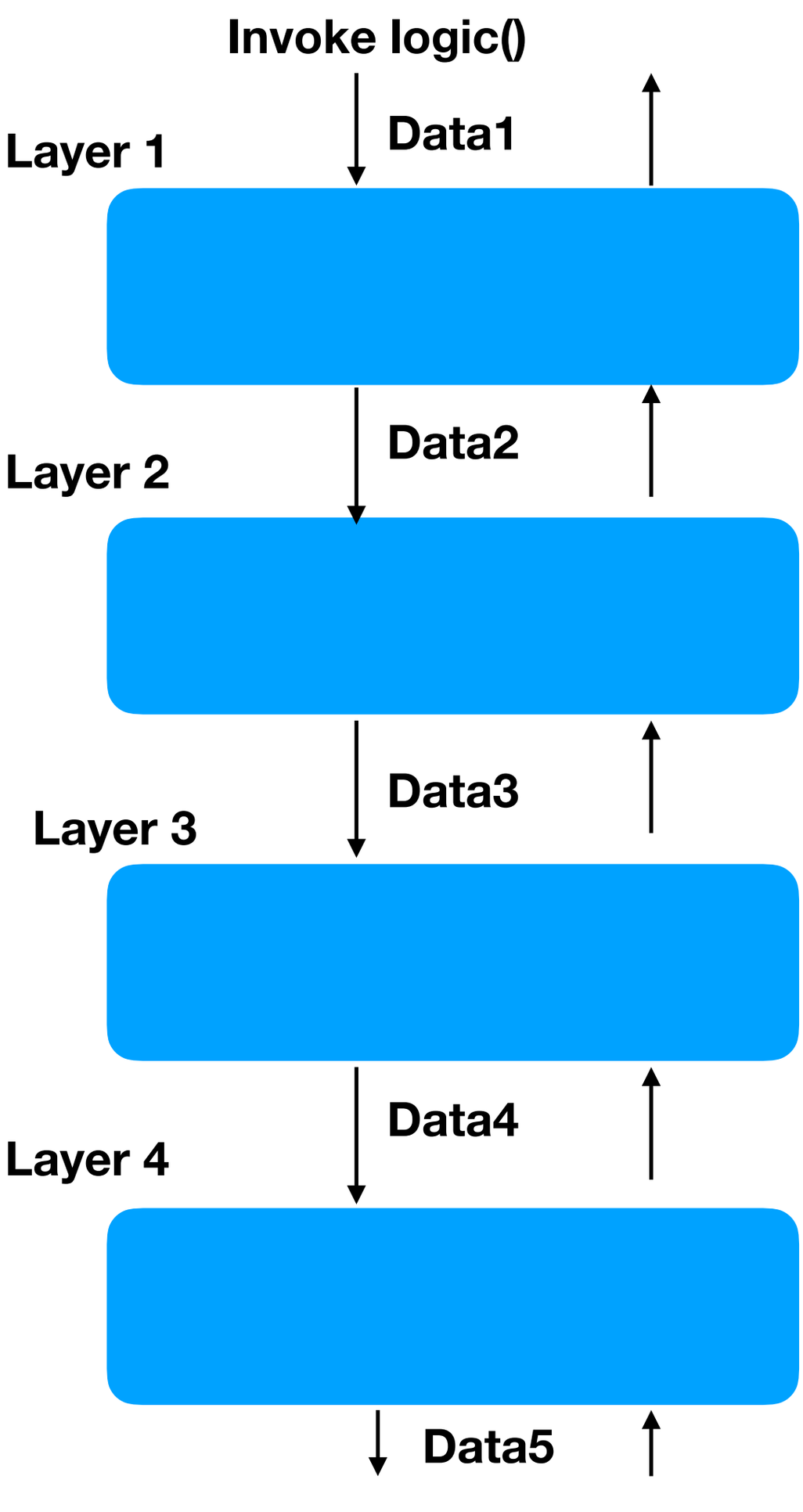


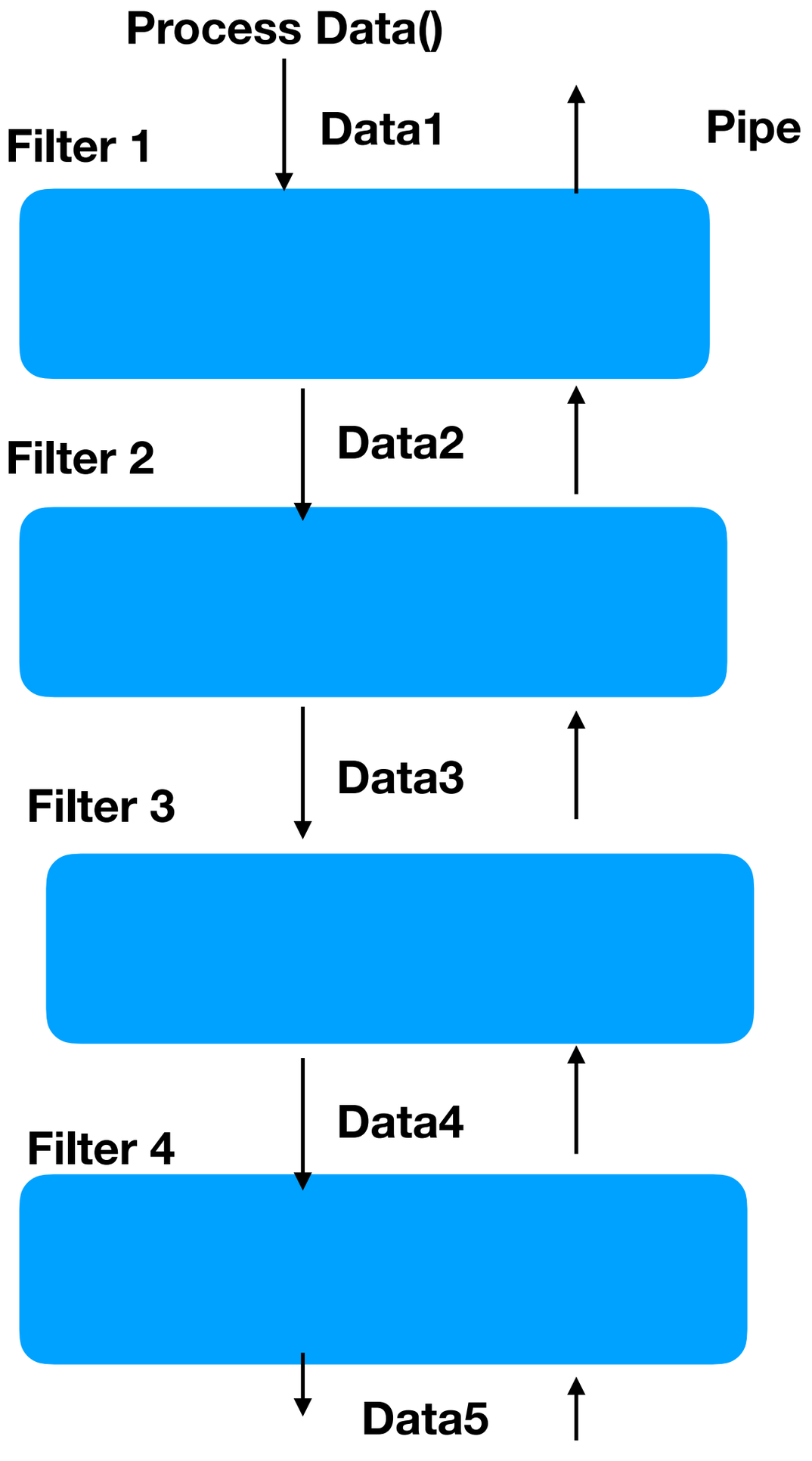
Split by Interaction

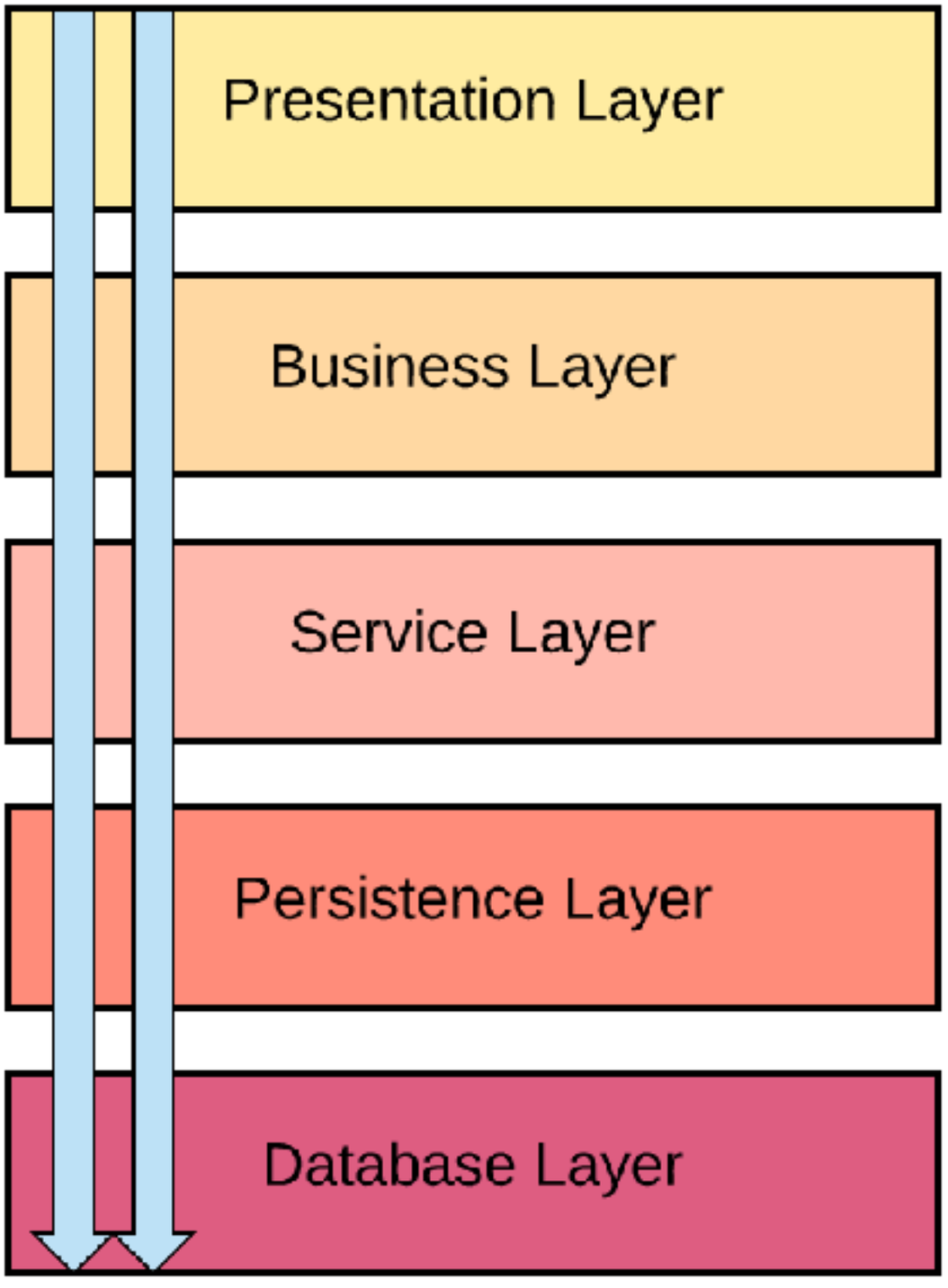
Split by Domain responsibilities

An application is required to perform a variety of tasks of varying complexity on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, **could change** as business requirements are updated. Also, **additional processing** might be required in the future, or the **order** in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

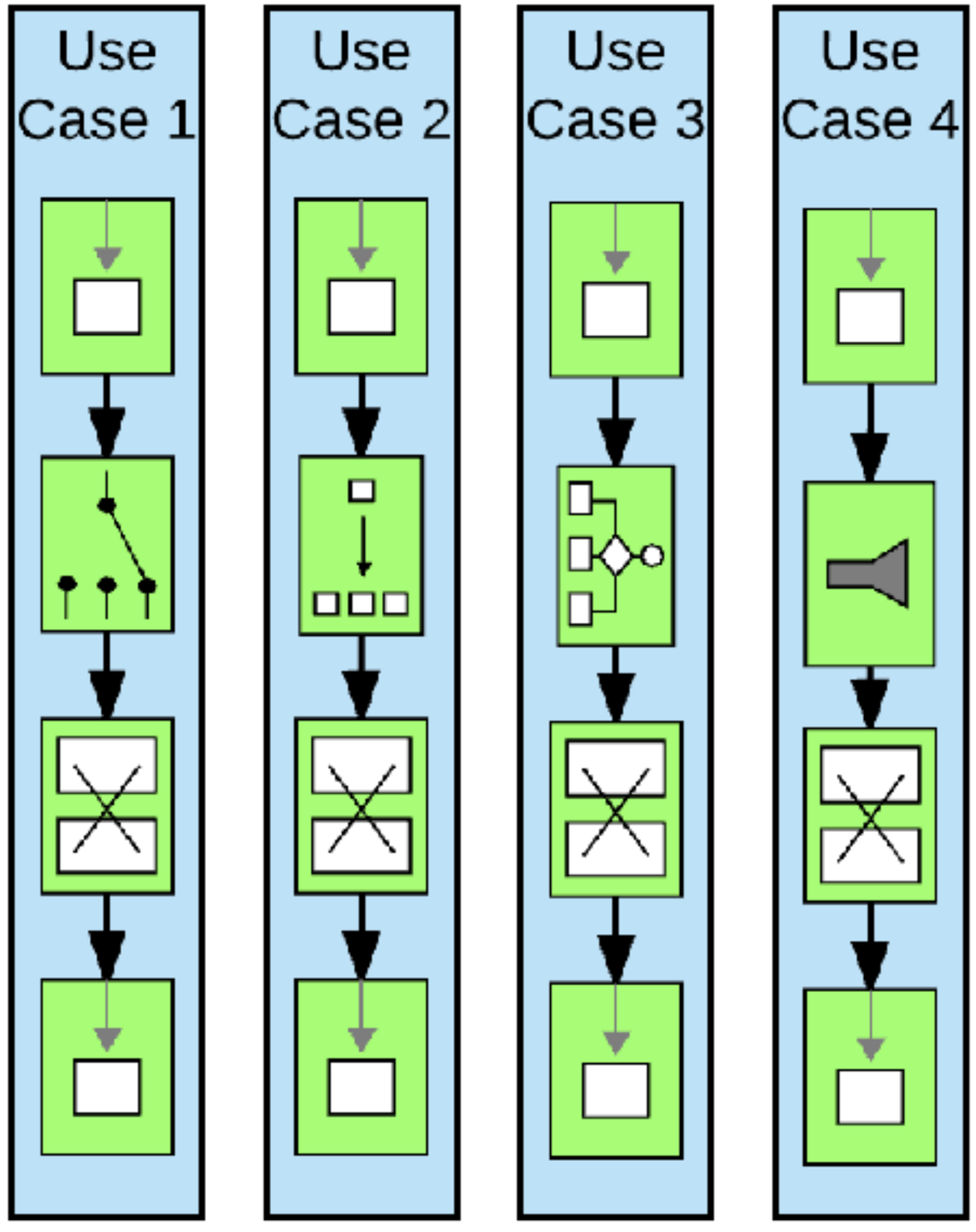
Eclipse IDE. Downloading the basic Eclipse product provides you little more than a fancy editor. However, once you start adding plug-ins, it becomes a highly customizable and useful product.


Invoke logic()
Layer 1
Data1
Data2
Layer 2
Data3
Layer 3
Data4
Layer 4
Data5


Process Data()
Filter 1
Data1
Pipe
Filter 2
Data2
Filter 3
Data3
Filter 4
Data4
Data5



Layered Architecture



Pipes and Filters

Invoke logic()

Data1

Layer 1

f1(int) f2() f3(int,int)

Data2

Layer 2

f3(int,int)

f1(bool) f4()

f2(double)

Data3

Layer 3

f1() f4()

f2(int) f3() f5(int)

Data4

Layer 4

f1(int) f2()

Data5

Process Data()

Data1

Pipe

Filter 1

f1(int)

Data2

Filter 2

f2(int)

Data3

Filter 3

f3(int)

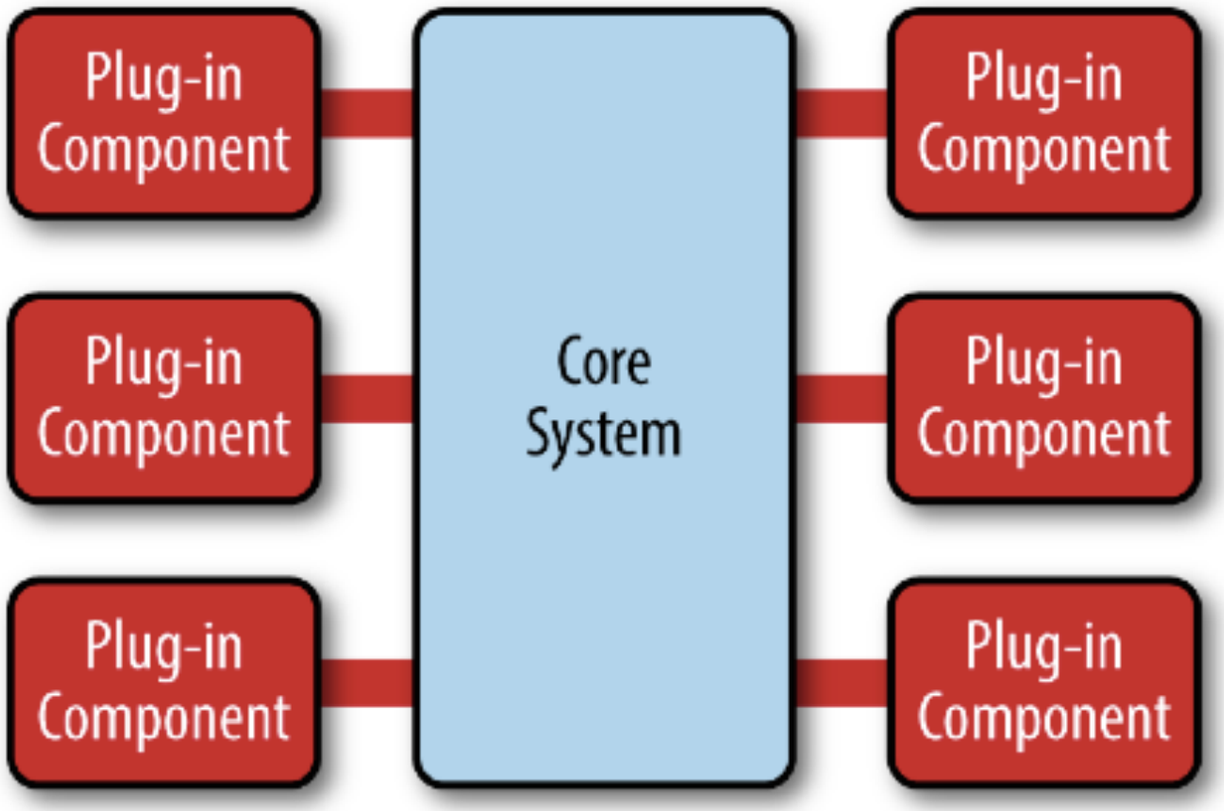
Data4

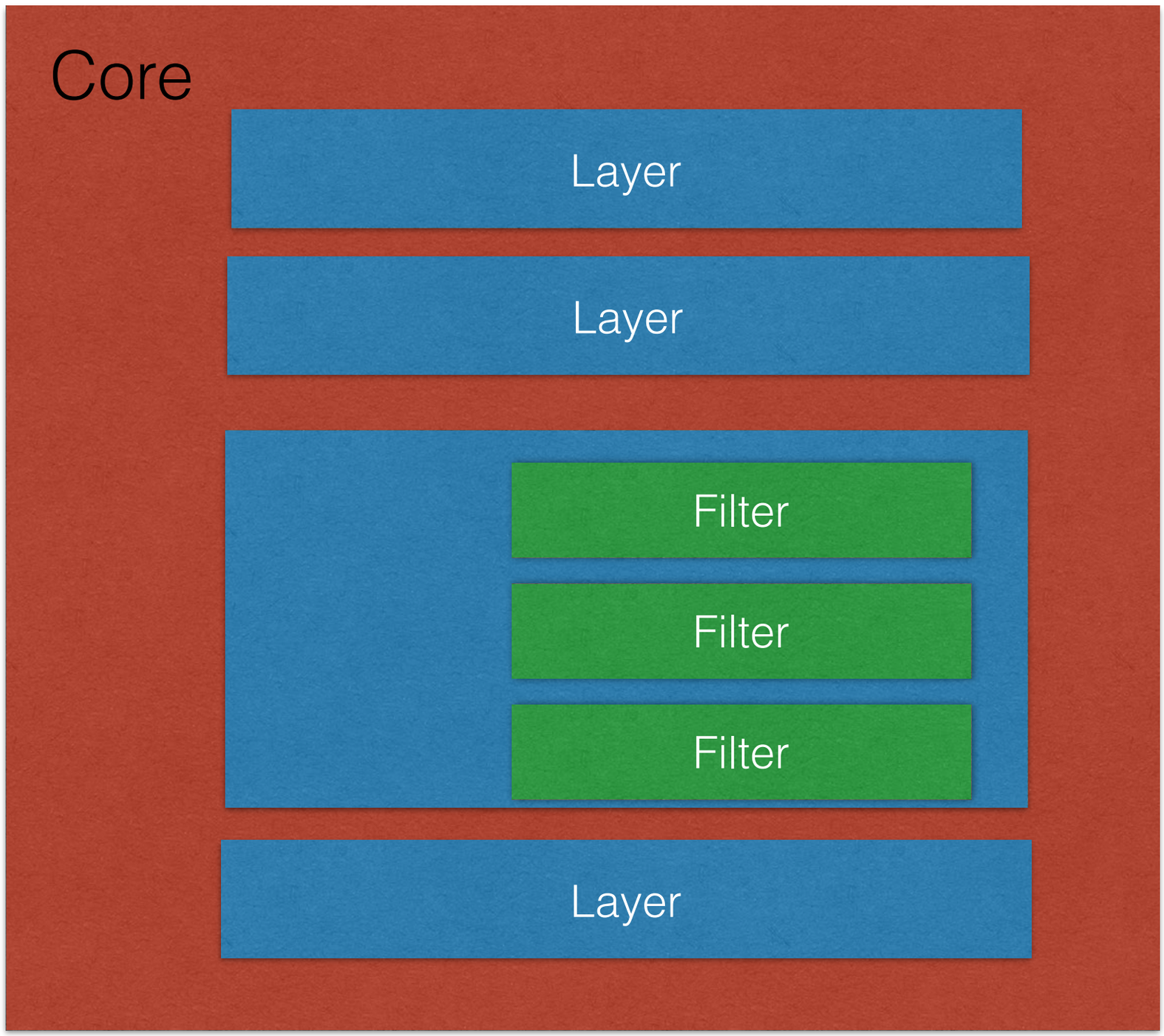
Filter 4

f4(int)

Data5

Microkernel Architecture

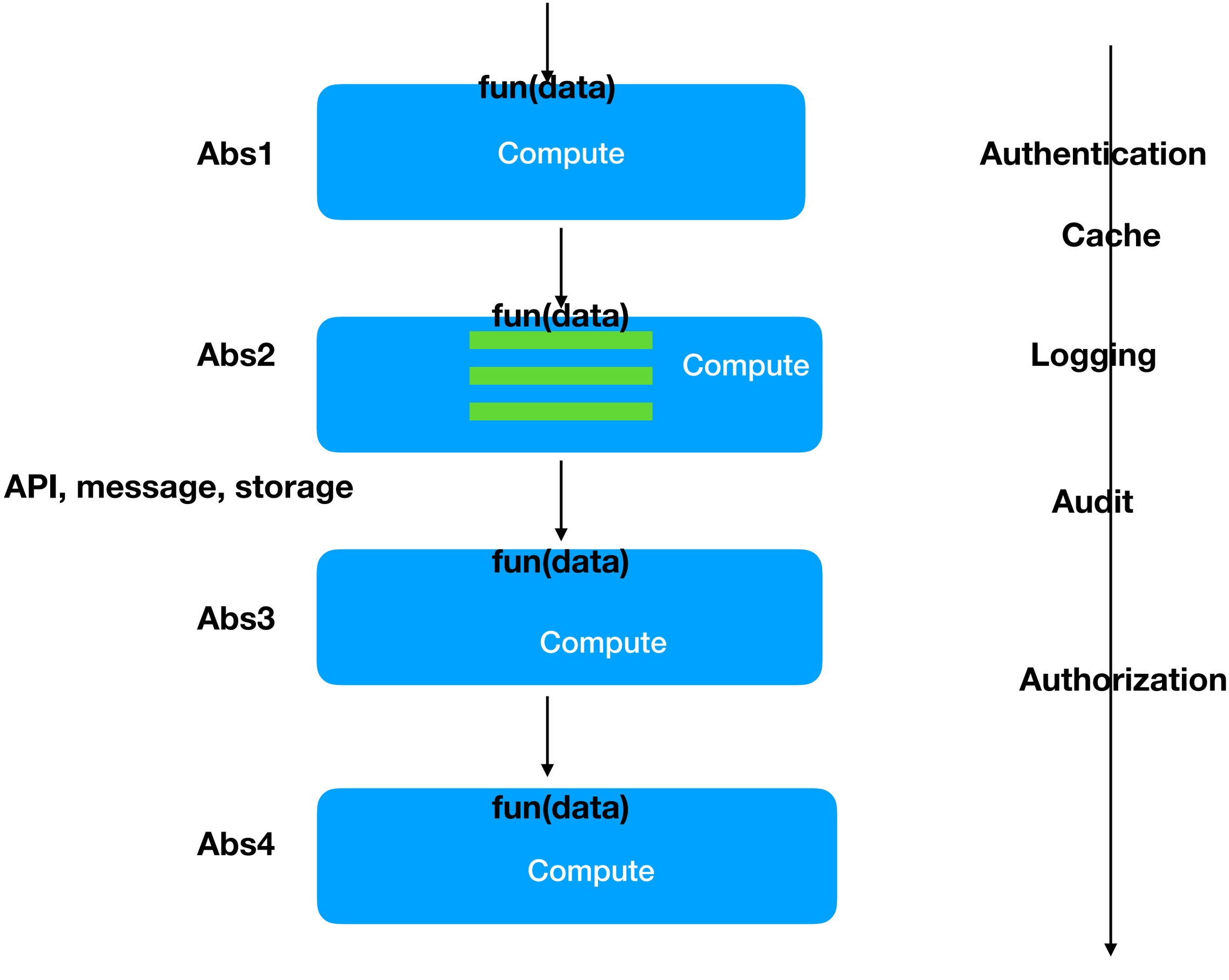


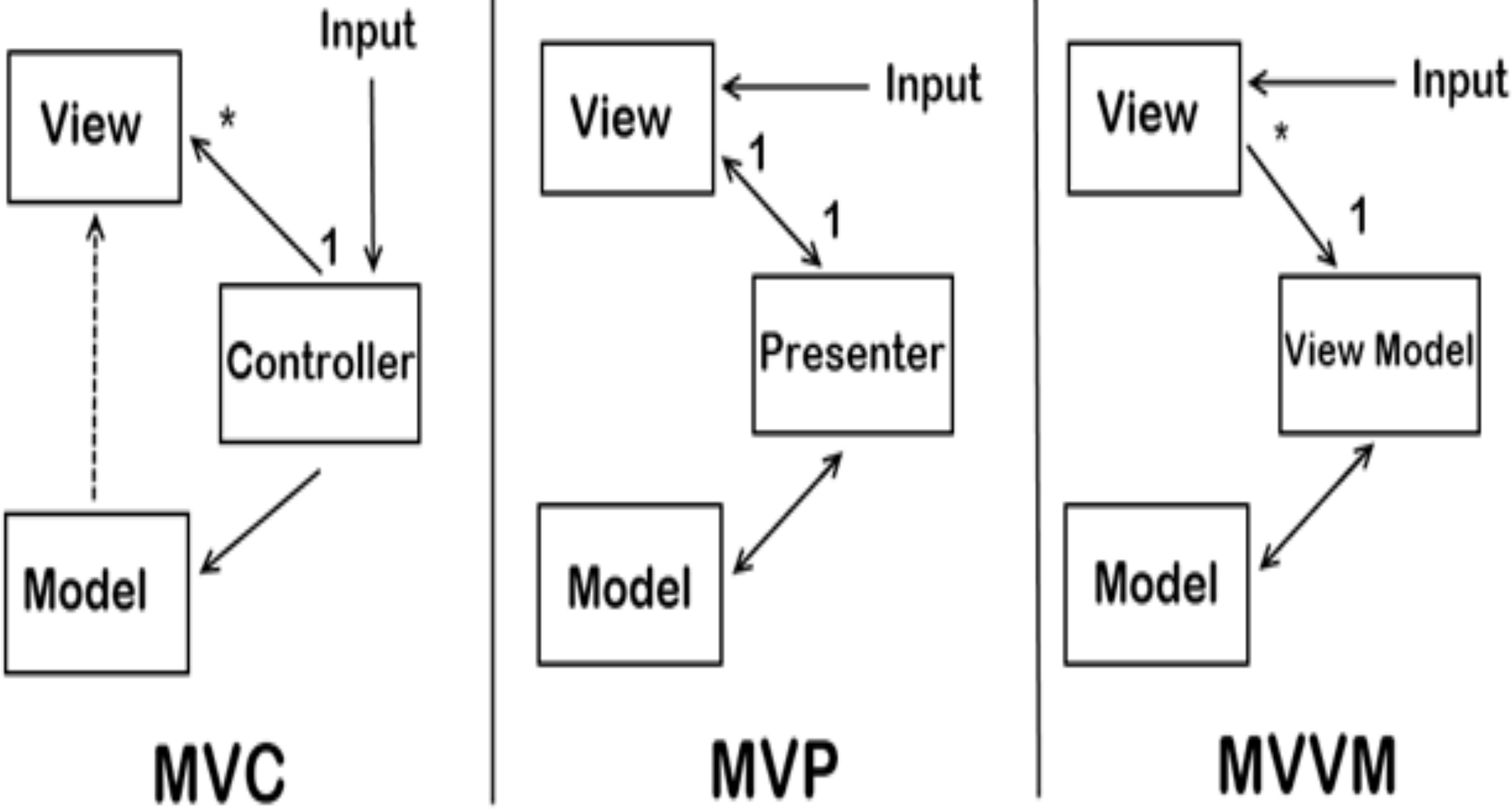


Plugins

Plugins

Plugins


fun(data)
Abs1
Compute
fun(data)
Abs2
Compute
API, message, storage
fun(data)
Abs3
Compute
fun(data)
Abs4
Compute
Authentication
Cache
Logging
Audit
Authorization


View
Input
*
1
Controller
Model
MVC
View
Input
1
1
Presenter
Model
MVP
View
Input
*
1
View Model
Model
MVVM

1

Ship one binary

Application
Exe

Source file

Source file

Compile time
monolithic

2

Ship one binary

Application
Exe

Lib binaries

Lib binaries

Source file

Link time
monolithic

3

Ship multiple binary

Process

Application
Exe

Dll/so/jar

Dll/so/jar

Dll/so/jar

Runtime
monolithic

In process
comp

Ship multiple binary & host in multiple process

4

Process

Exe

Dll/so/jar

Distributed
comp

Process

Exe

Dll/so/jar

Distributed
comp

Distributed Application

5

Process

Exe

Dll/so/jar

Application1

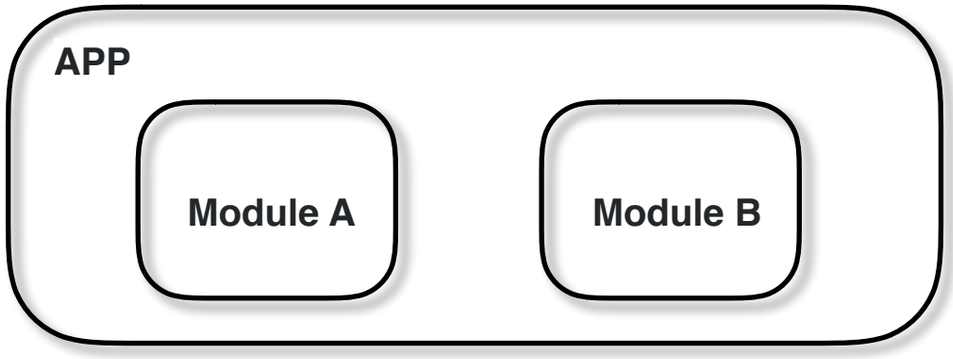
Process

Exe

Dll/so/jar

Application2

Micro Application



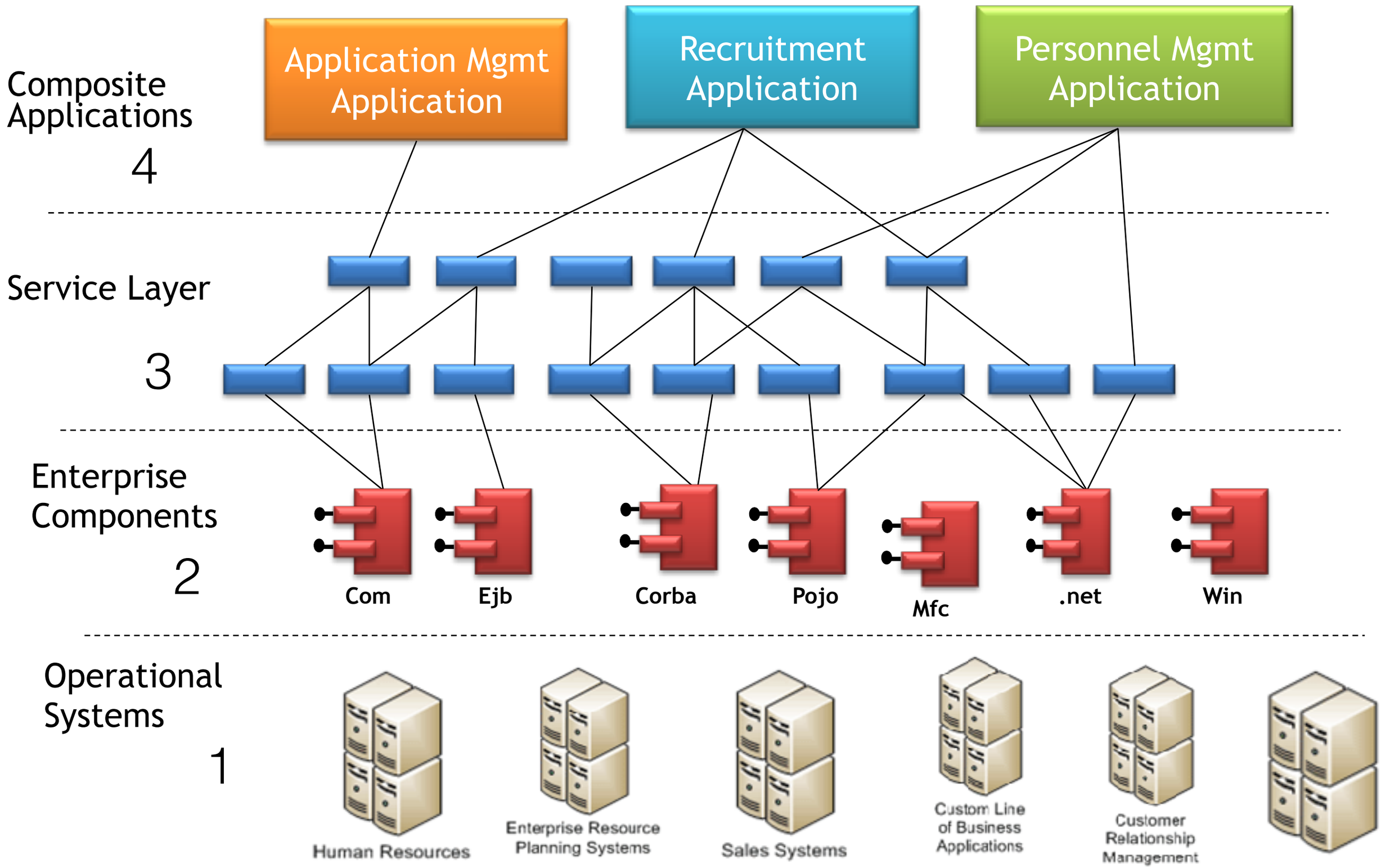
Can share persistence

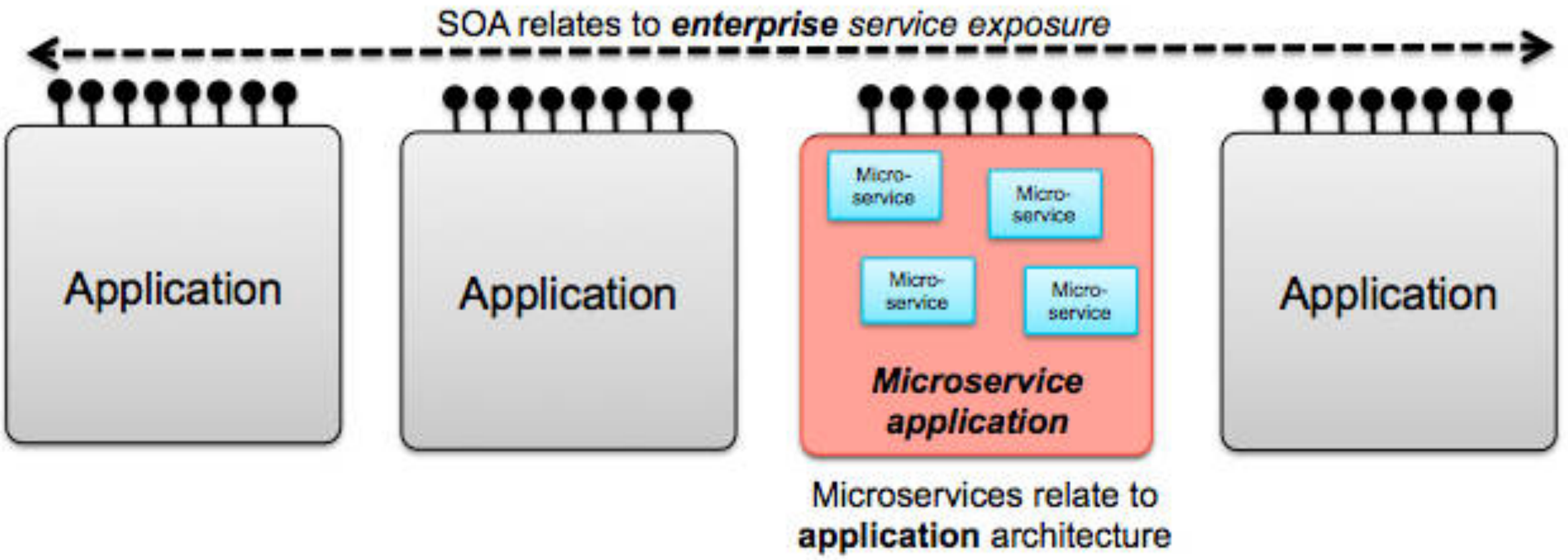
Should not share persistence

Can share Infra

Should not share Infra

Service Oriented Architecture

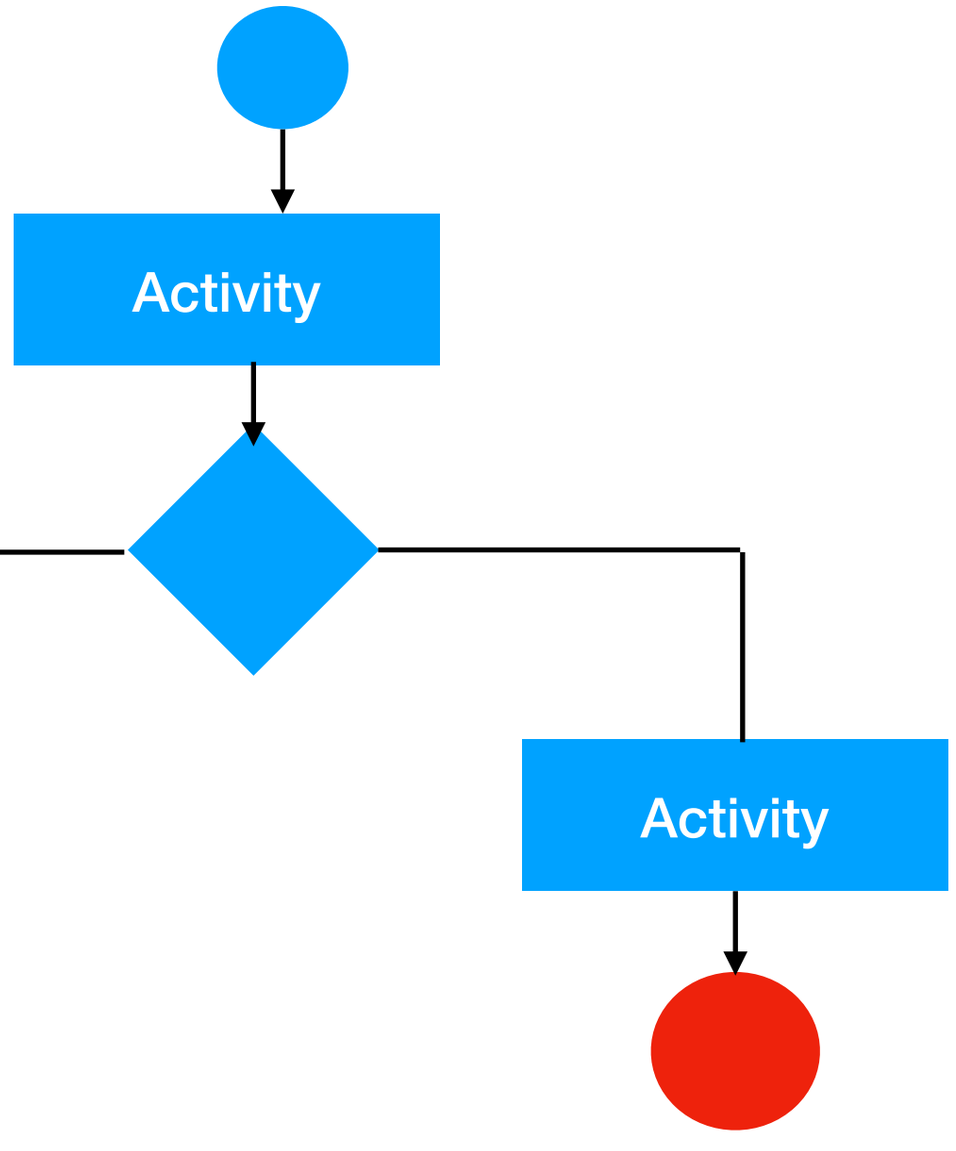


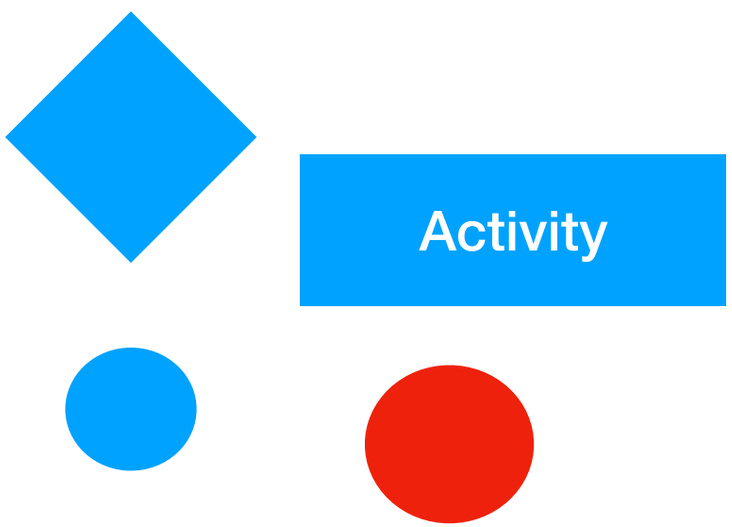


Microservices relate to **application** architecture

An application is required to perform a variety of tasks of varying complexity on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, could change as business requirements are updated. Also, additional processing might be required in the future, or the order in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

A workflow implementation. The implementation of a workflow contains concepts like the order of the different steps, evaluating the results of steps, deciding what the next step is, etc.





A task scheduler. A scheduler contains all the logic for scheduling and triggering tasks

Internet browses plug-ins add additional capabilities that are not otherwise found in the basic browser

Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is decomposed into more manageable parts.

Break an app into smaller manageable piece

	2 Modules	**2 Applications**	**W**	**Score**
Share Database / Storage	Yes	No	2	
Share Infra (Hosting)	Yes	No	3	
Share Sorce Control	Yes	No	2	
Share CI/CD (Build Server)	Yes or No	No	3	
Share functional logic (same feature)		No		
Fun Requirements	Shared	Its own	1	
SCRUM Team / Sprint	Shared	Its own	1	
Test Cases	Shared	Its own	1	
Architecture	Shared	Its own	1	
Technology Stack / Fwks	Shared	Its own	1	

Application

Modules

Modules

Modules

3 or 4

Application

Small App

Small App

Small App

Modules

Modules

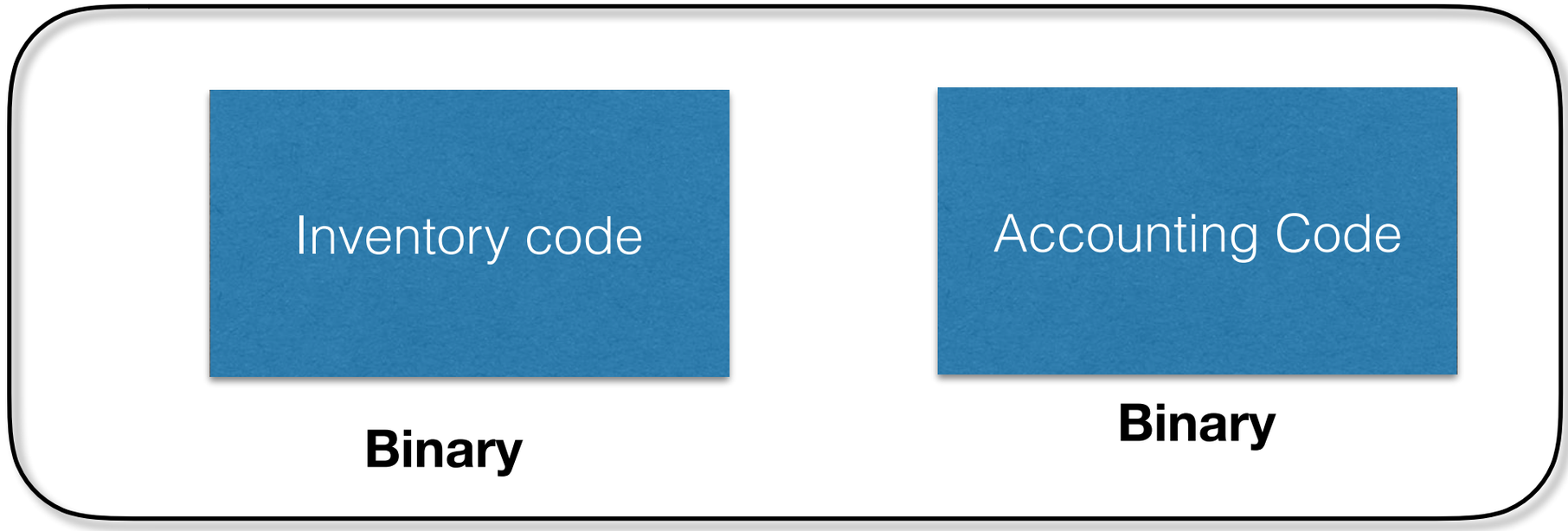
Modules

Modules

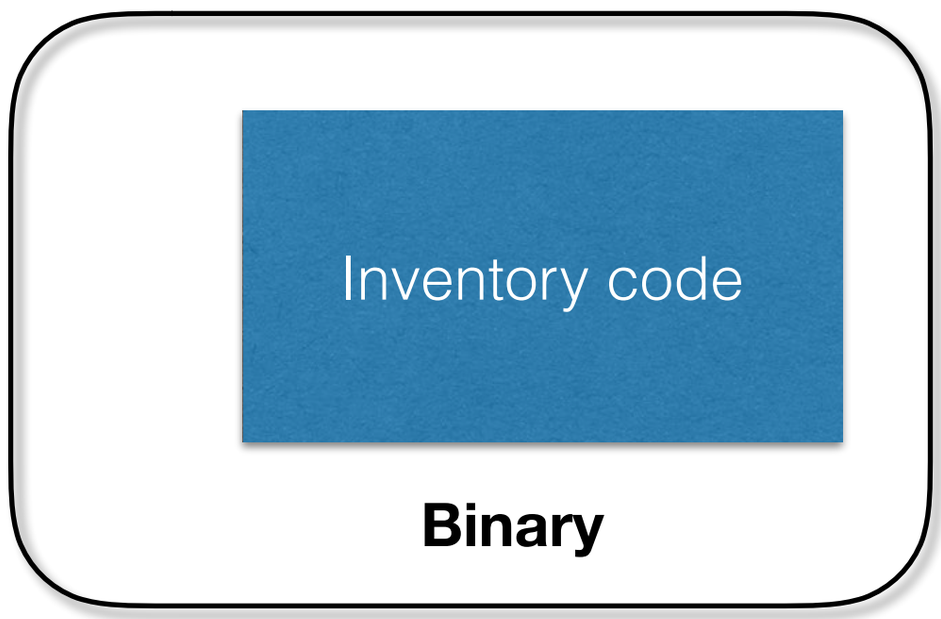
Modules

Modules

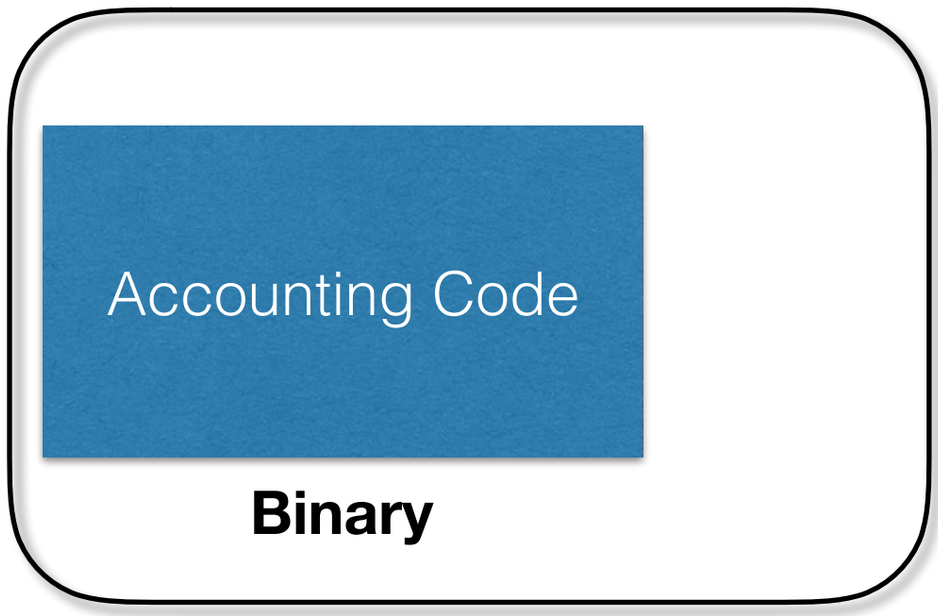
App

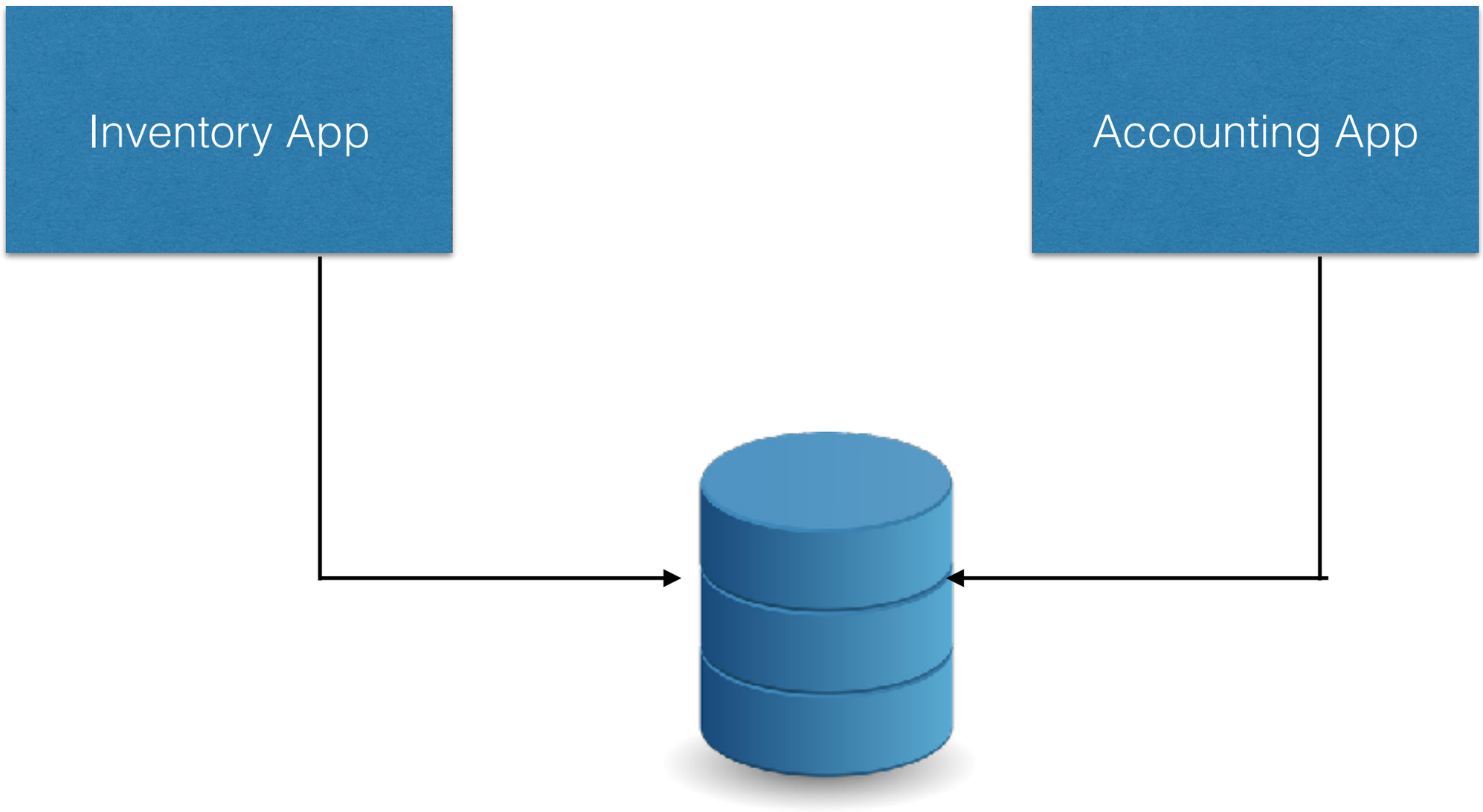

Inventory code
Accounting Code
Binary
Binary

App

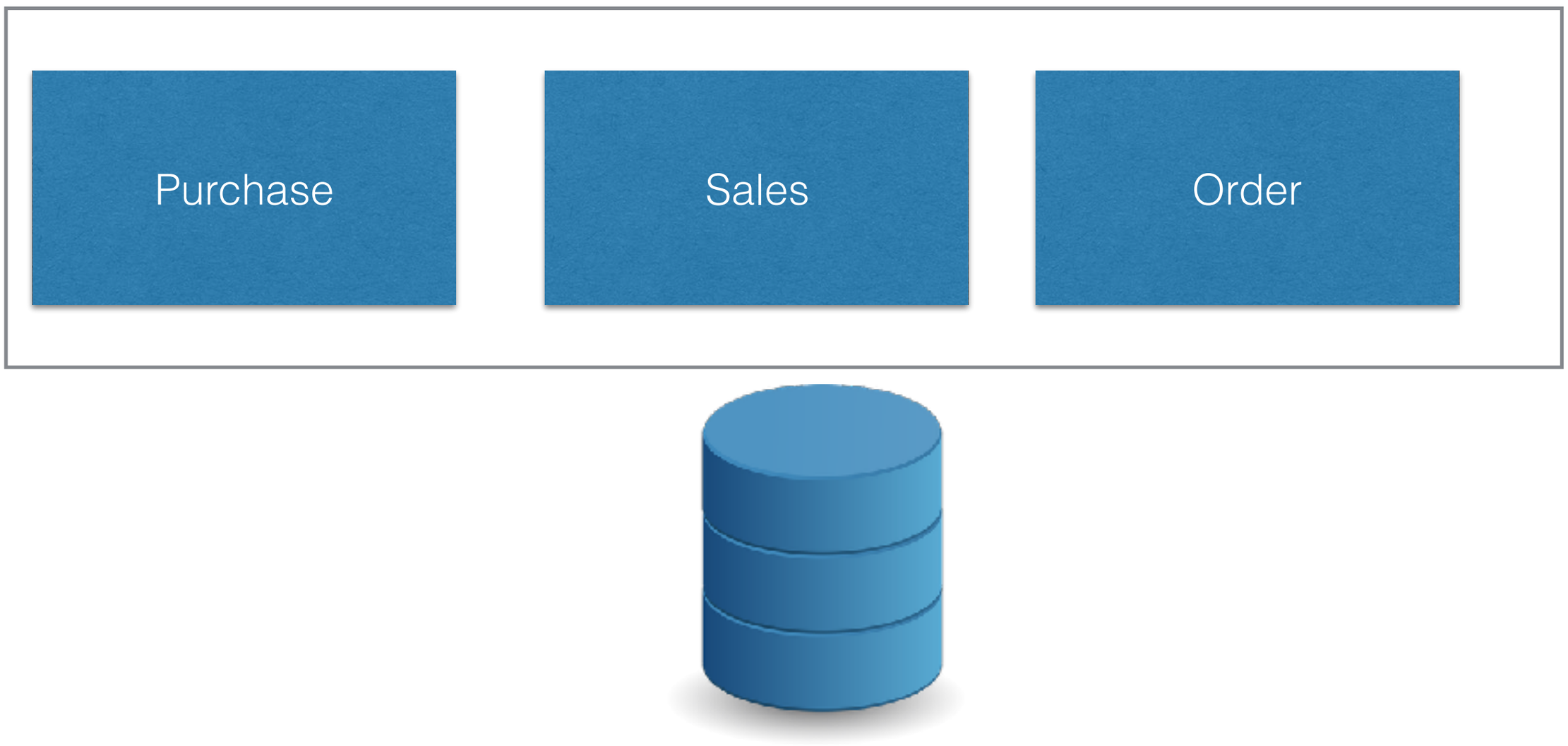

Inventory code
Binary

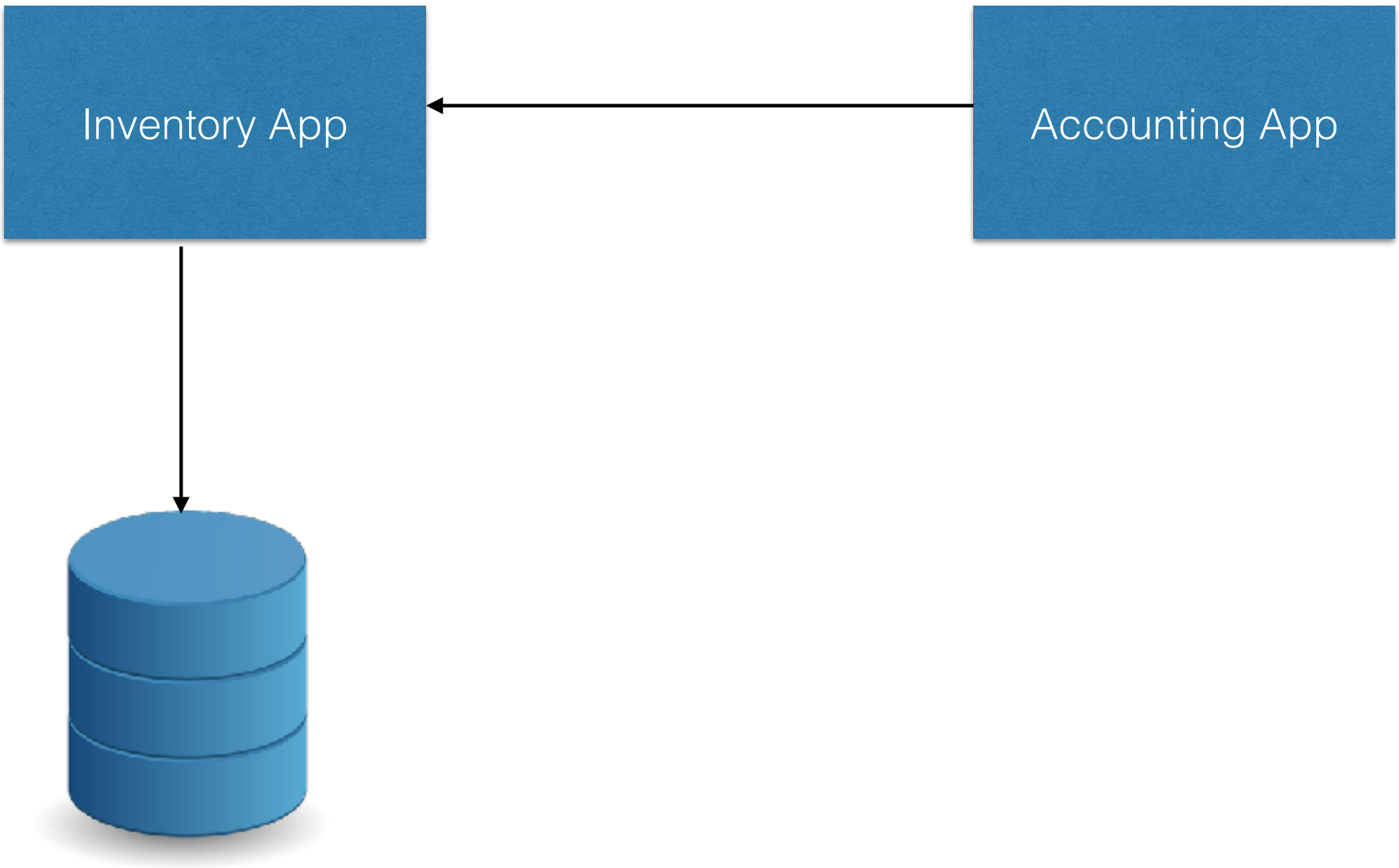
App


Accounting Code
Binary


Inventory App
Accounting App

Inventory Application




Inventory App
Accounting App

VM

Inventory App

Accounting App


Inventory App


Accounting App

APP

APP

APP

APP

APP

Types of Microservice

1
Web Server

UI App (SPA)

API Layer

Domain Layer
<<feature1>>

Data Layer

Angular (SPA)

2
Web Server

Browser

UI Layer
(server page)

Domain Layer
<<feature2>>

Data Layer

JSP, Servlet, JSF (Server pages)

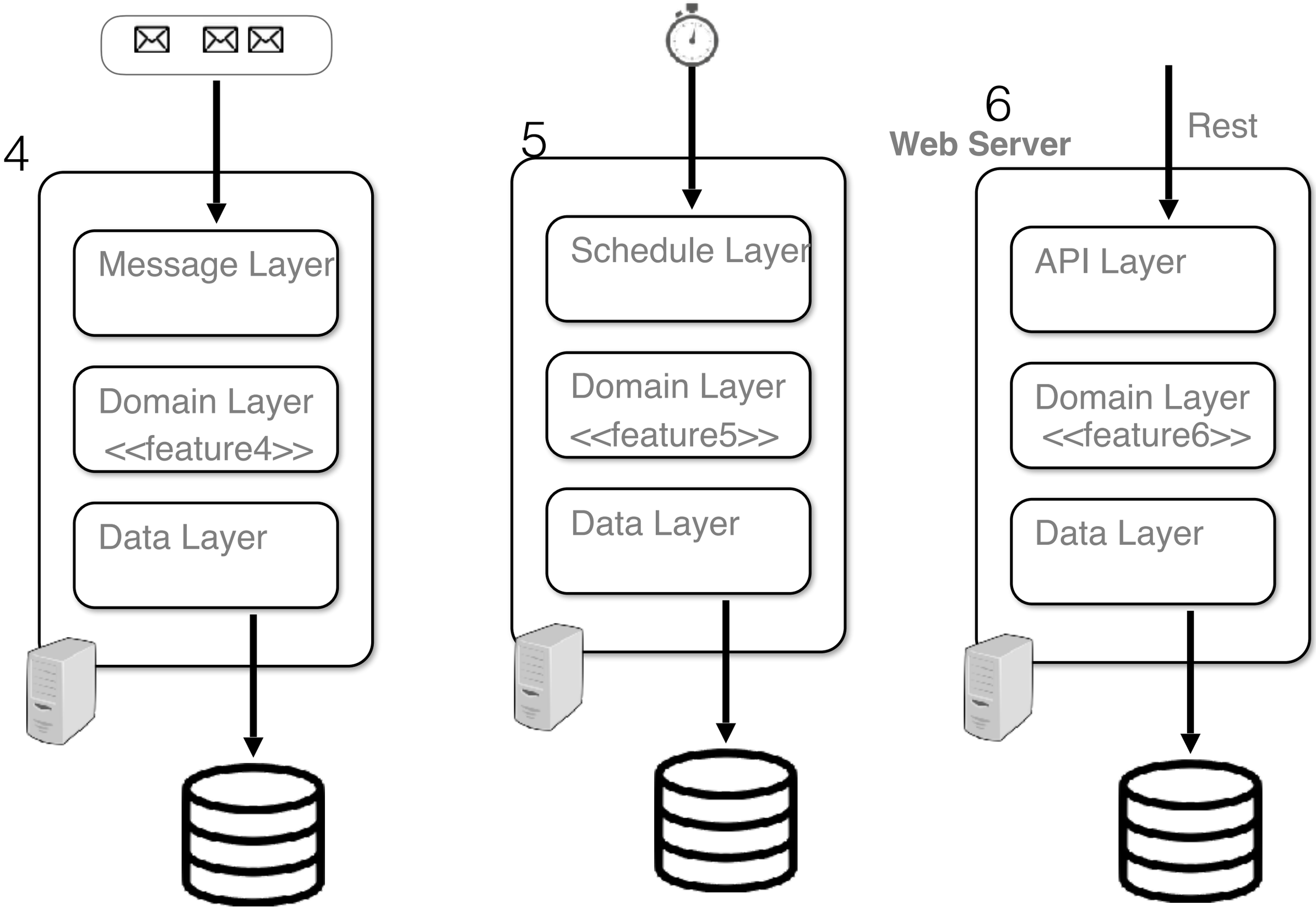
3
Desktop

UI Layer
(native)

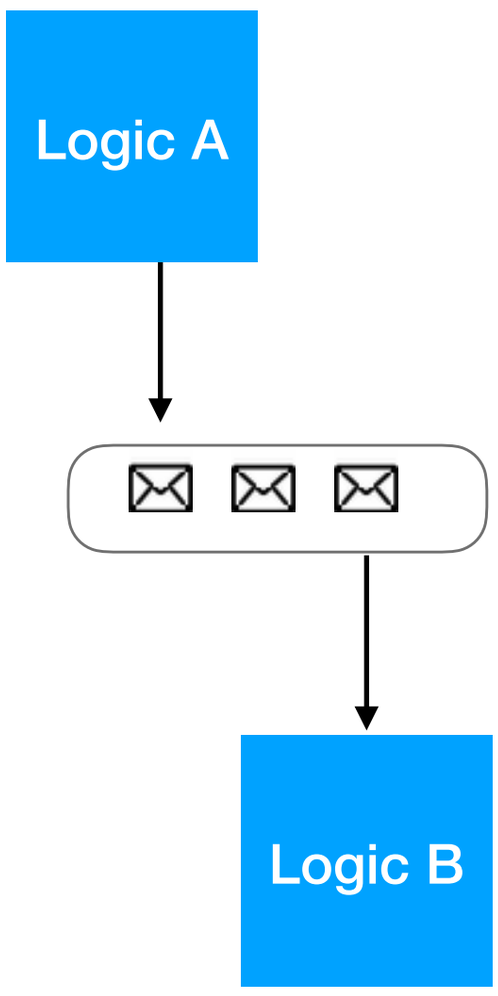
Domain Layer
<<feature3>>

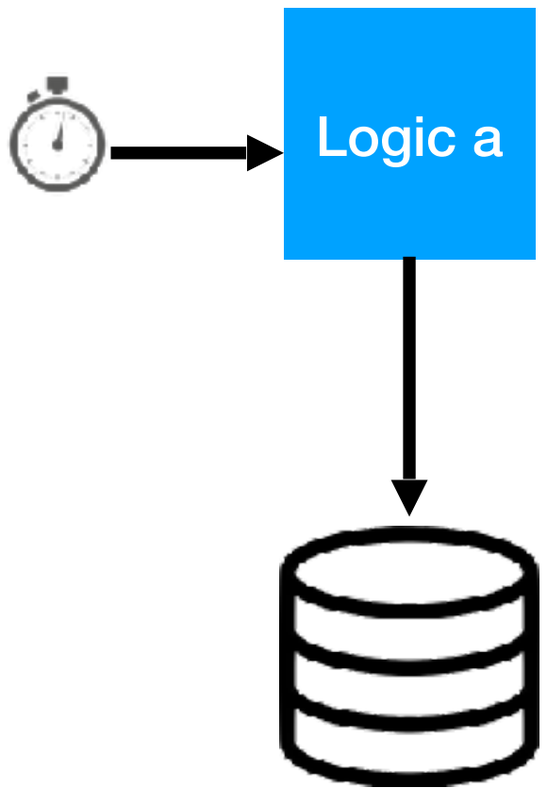
Data Layer

Swing, Android, Ios (Native)


4
Message Layer
Domain Layer
<<feature4>>
Data Layer
5
Schedule Layer
Domain Layer
<<feature5>>
Data Layer
6
Web Server
Rest
API Layer
Domain Layer
<<feature6>>
Data Layer

Stream vs Batch Processing



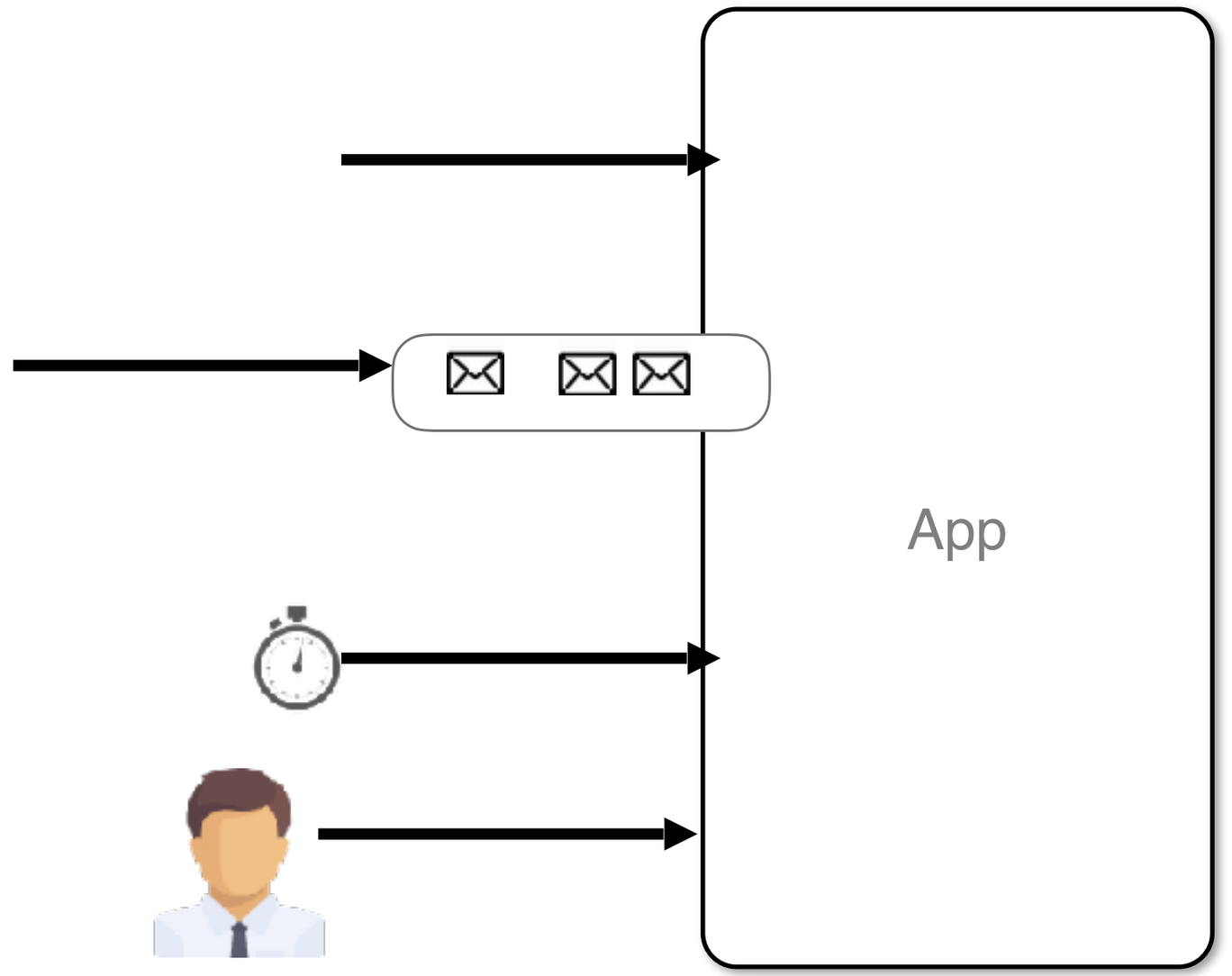


API

* Message

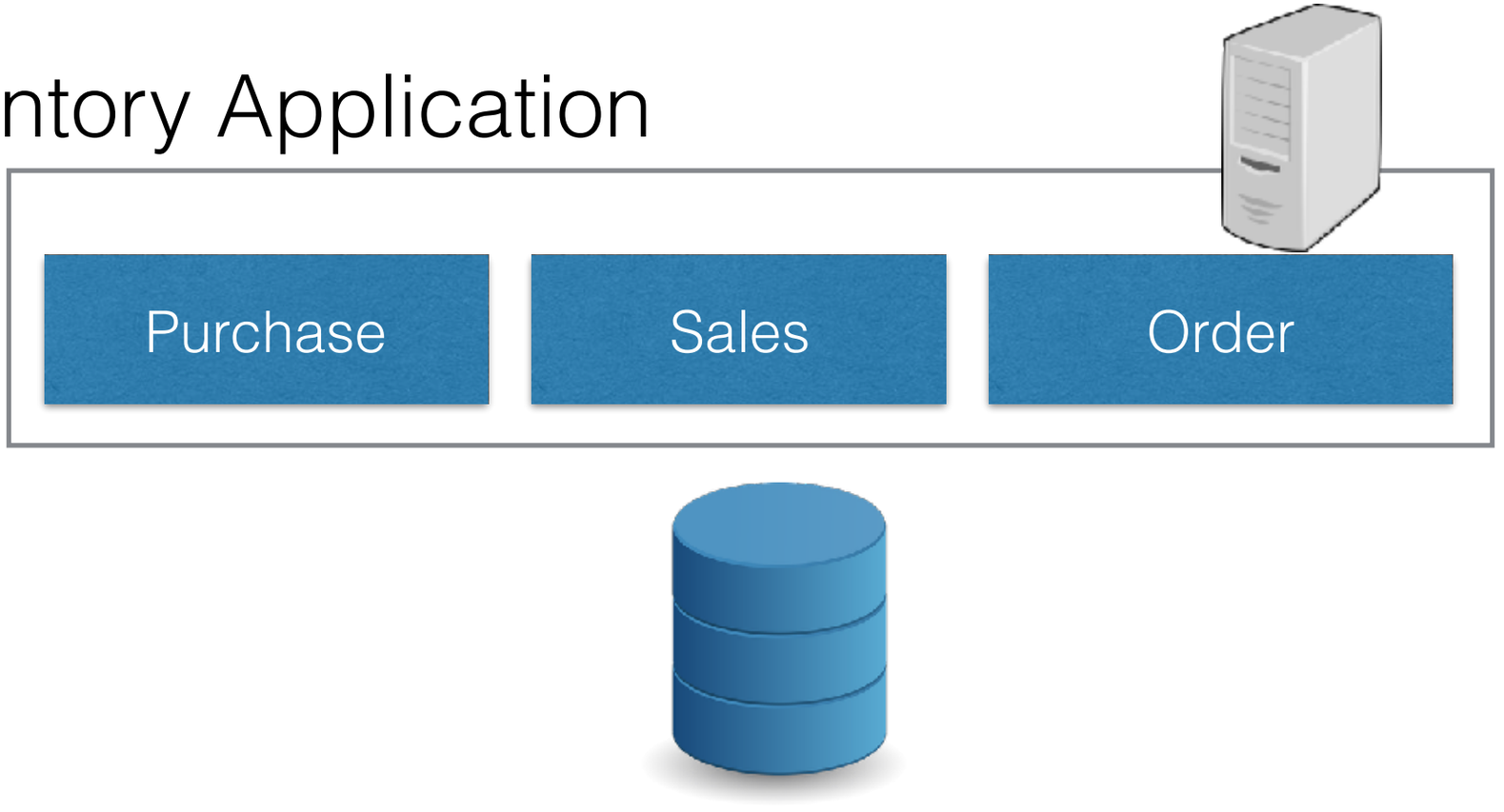
Schedule

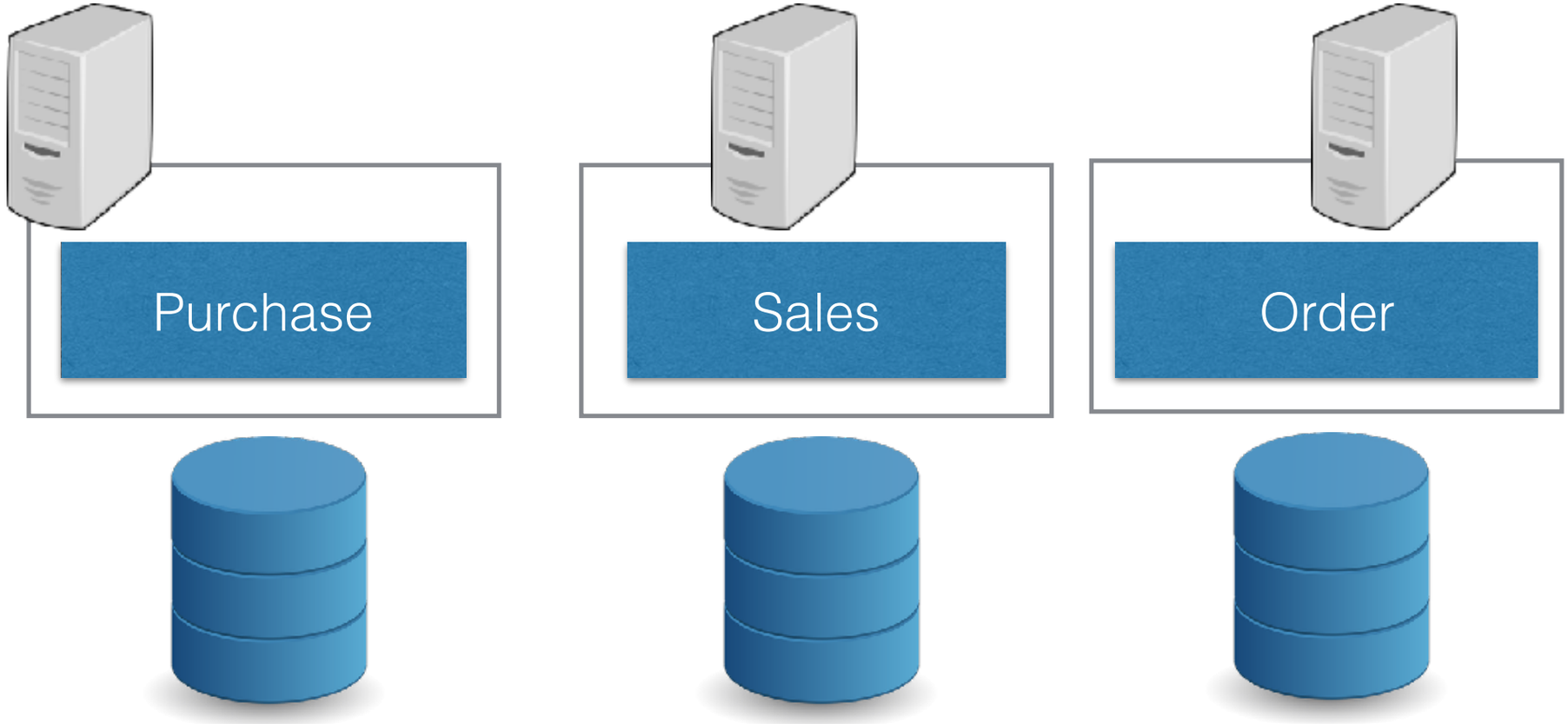
UI



	Pros/ Cons	Solution
Development time	- -	
Micro-service practices Learning Curve	- -	
Resource Performance (CPU, Memory, I/O)	- -	
Db Transaction Management	- -	SAGA
Views / Report / Dash board/ join	- -	
Infra Cost	- -	
First time Deployment effort	- -	
Debugging, Error Handling (End to End)	- -	
Integration Test	- -	
Log Mgmt (debug/ error)	- -	CENTRALIZE (EFK)
Config Mgmt	- -	CENTRALIZE
Authentication (who)	- -	CENTRALIZE
Authorization (what can they do)	- -	CENTRALIZE
Audit Log mgmt (who accessed what)	- -	CENTRALIZE
Monitoring / Alerting	- -	CENTRALIZE
Data Security and Privacy (transit, storage)	- -	
Build Pipeline (CI)	- -	Automation
Agile Architecture (Agility to change)	+ + +	
Feature Shipping (Agility to ship)/ CD	+ +	
Scalability (volume - request, data,	+	
Availability	+	
Ability to do Polygot	+	

Inventory Application

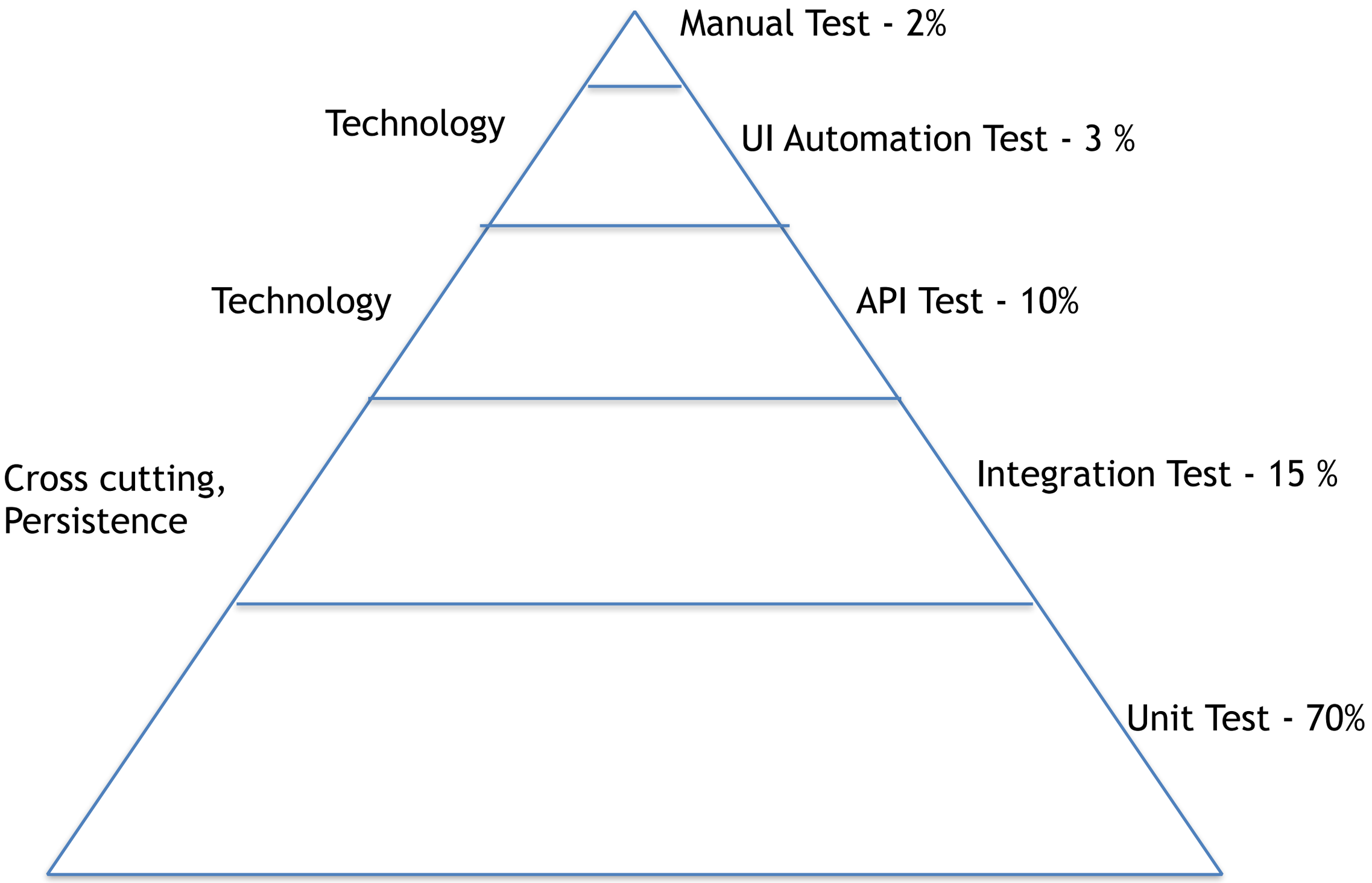




Decentrailze Domain

Centrailze Tech

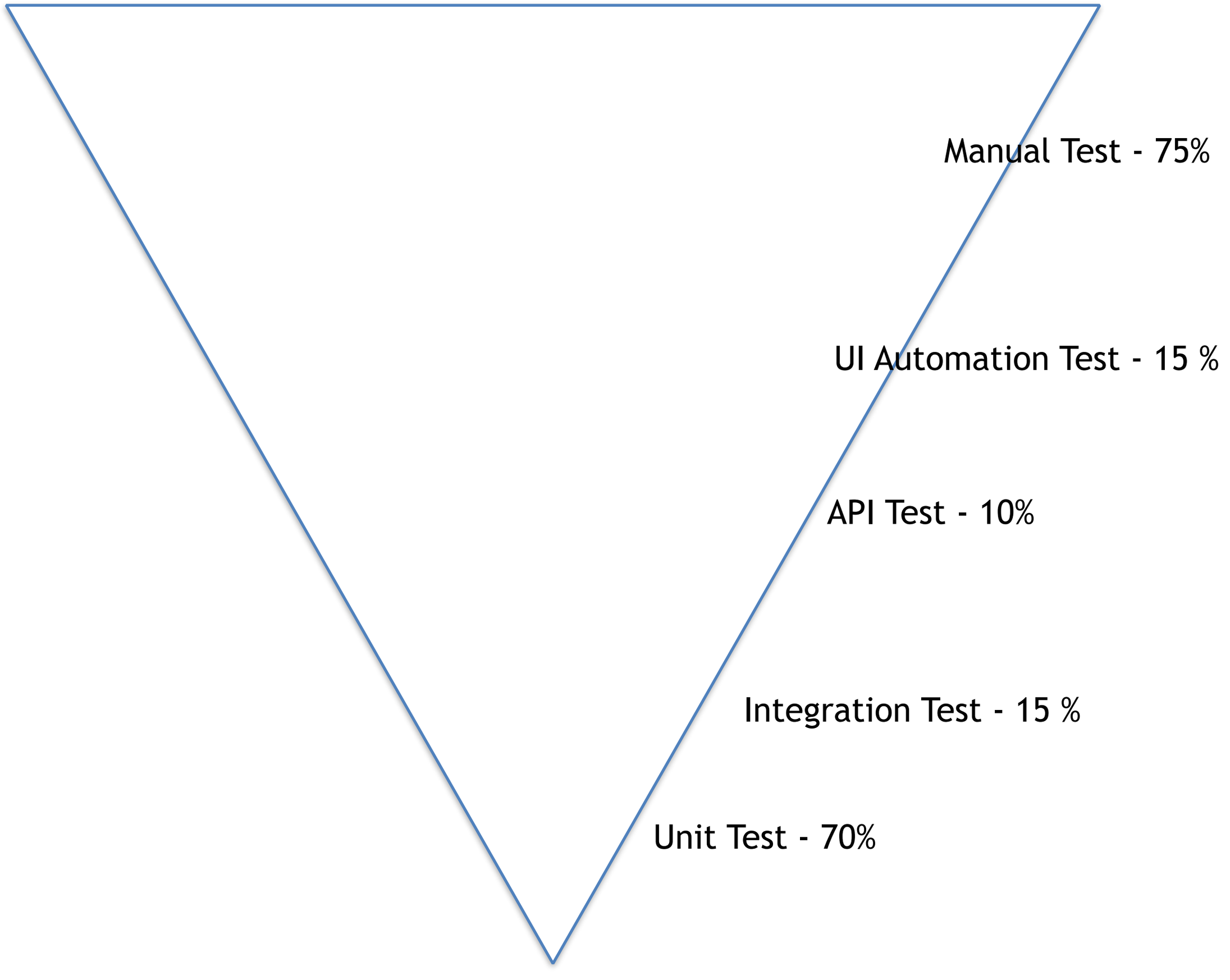
Pyramid test

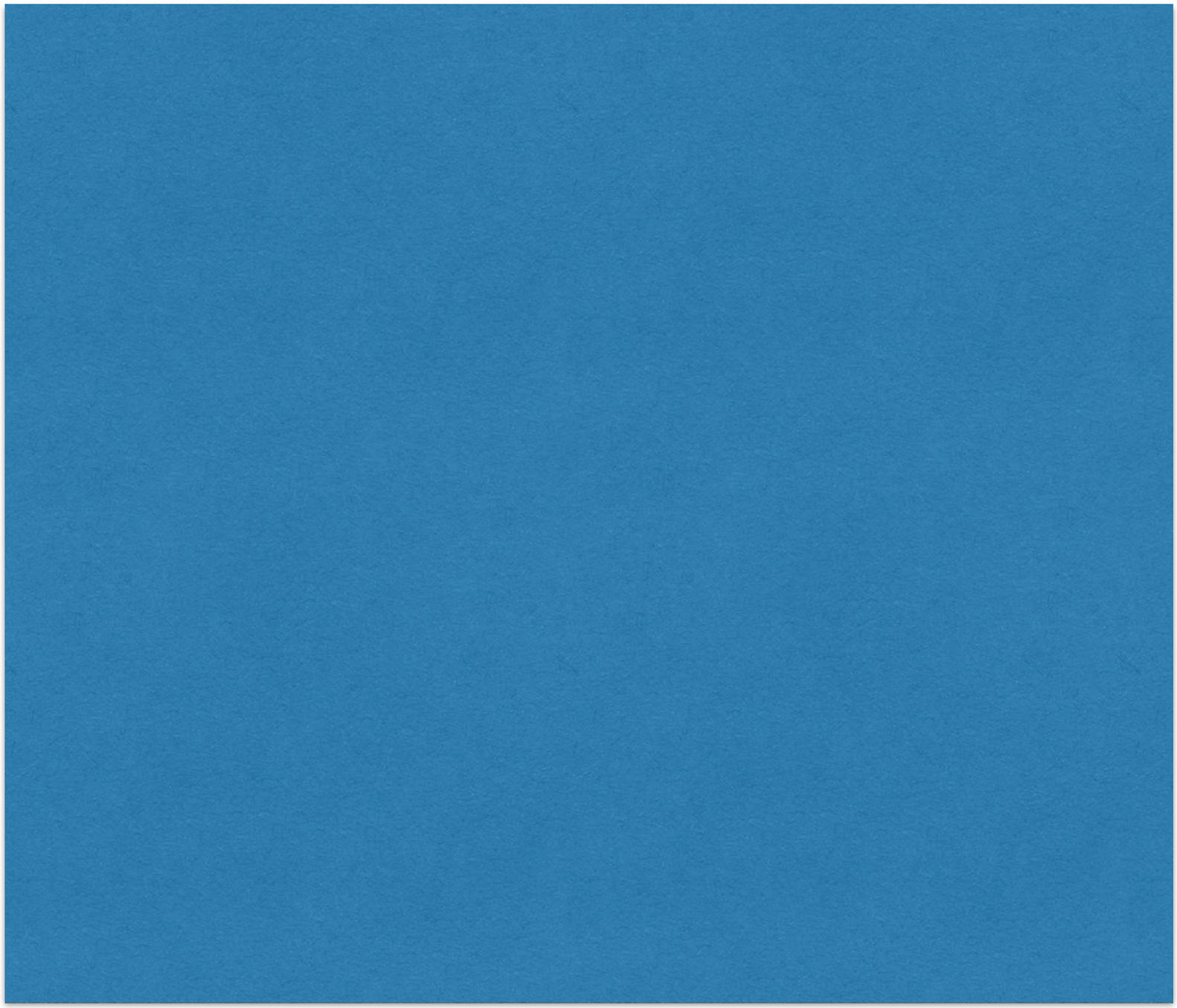


Unit Test

- Documentation for Code
- Design Code
- Regression
- Find Bugs

Its working don’t touch it


Manual Test - 75%
UI Automation Test - 15 %
API Test - 10%
Integration Test - 15 %
Unit Test - 70%



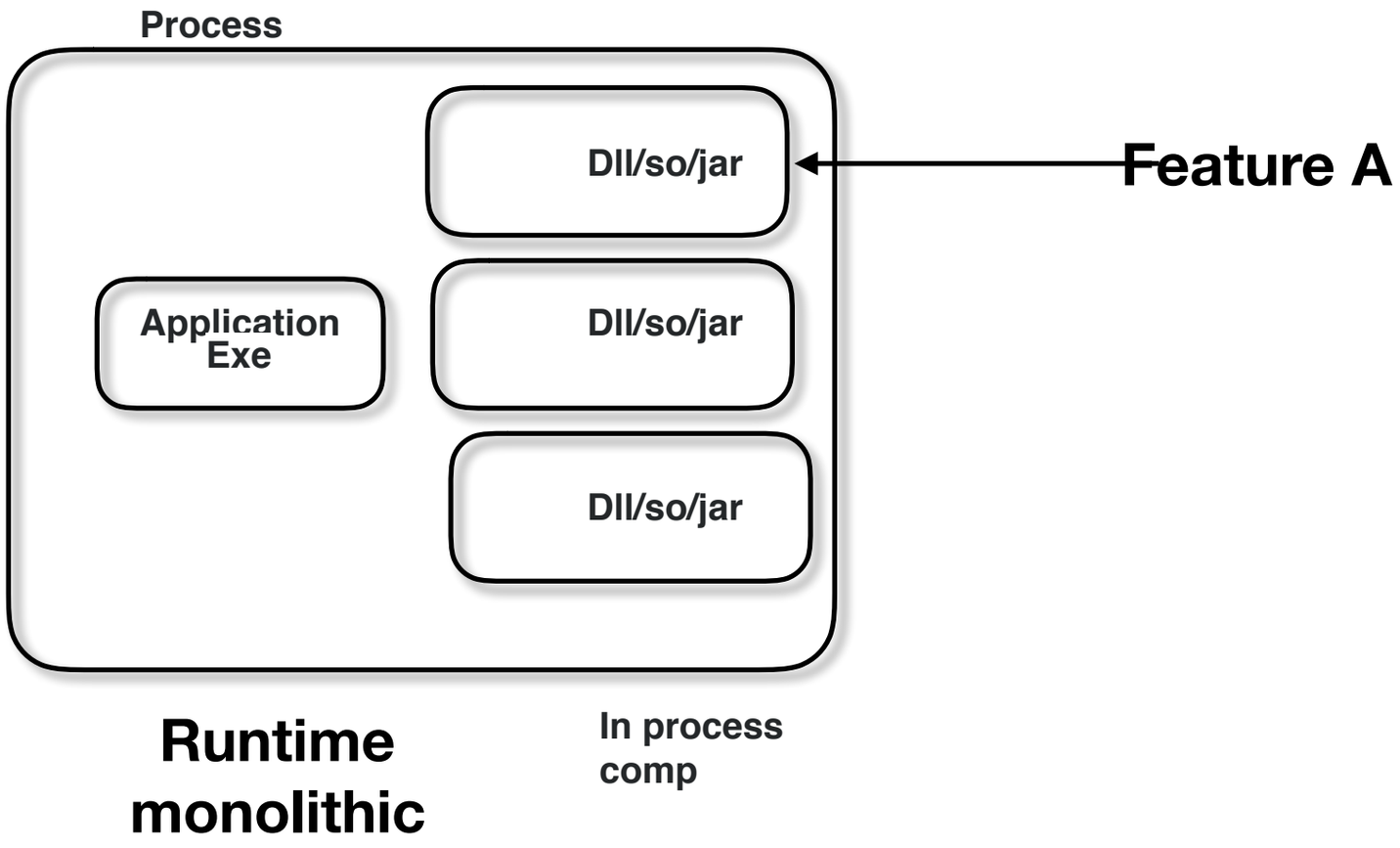
Manual Test - 2%

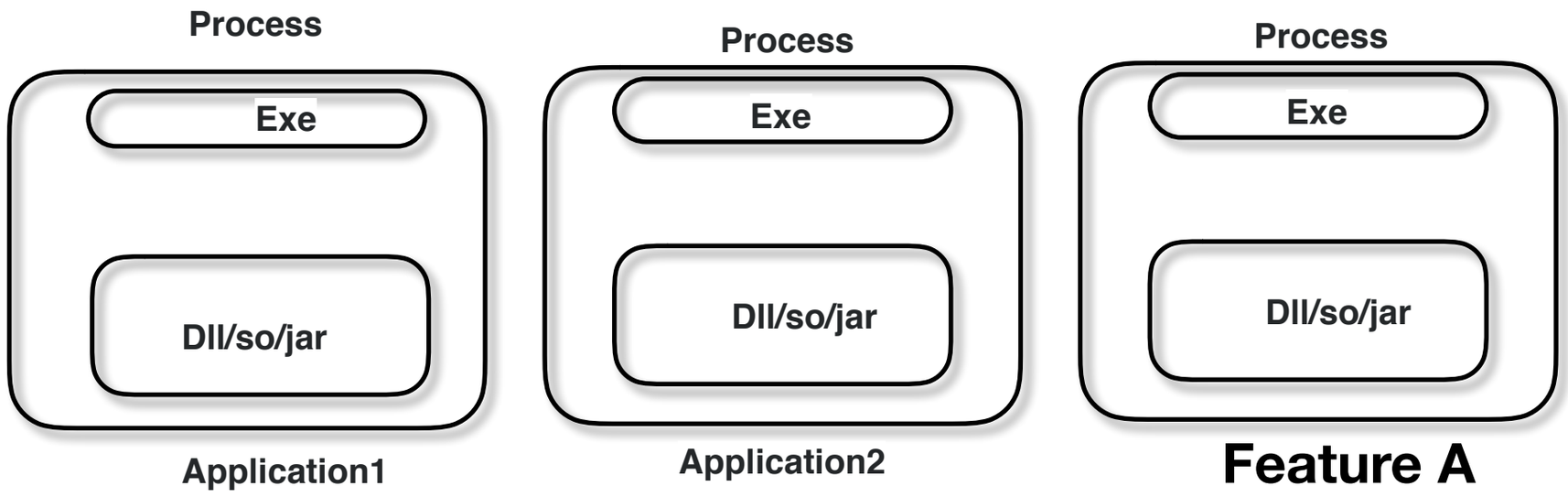
UI Automation Test - 3 %

API Test - 10%

Integration Test - 15 %

Unit Test - 70%

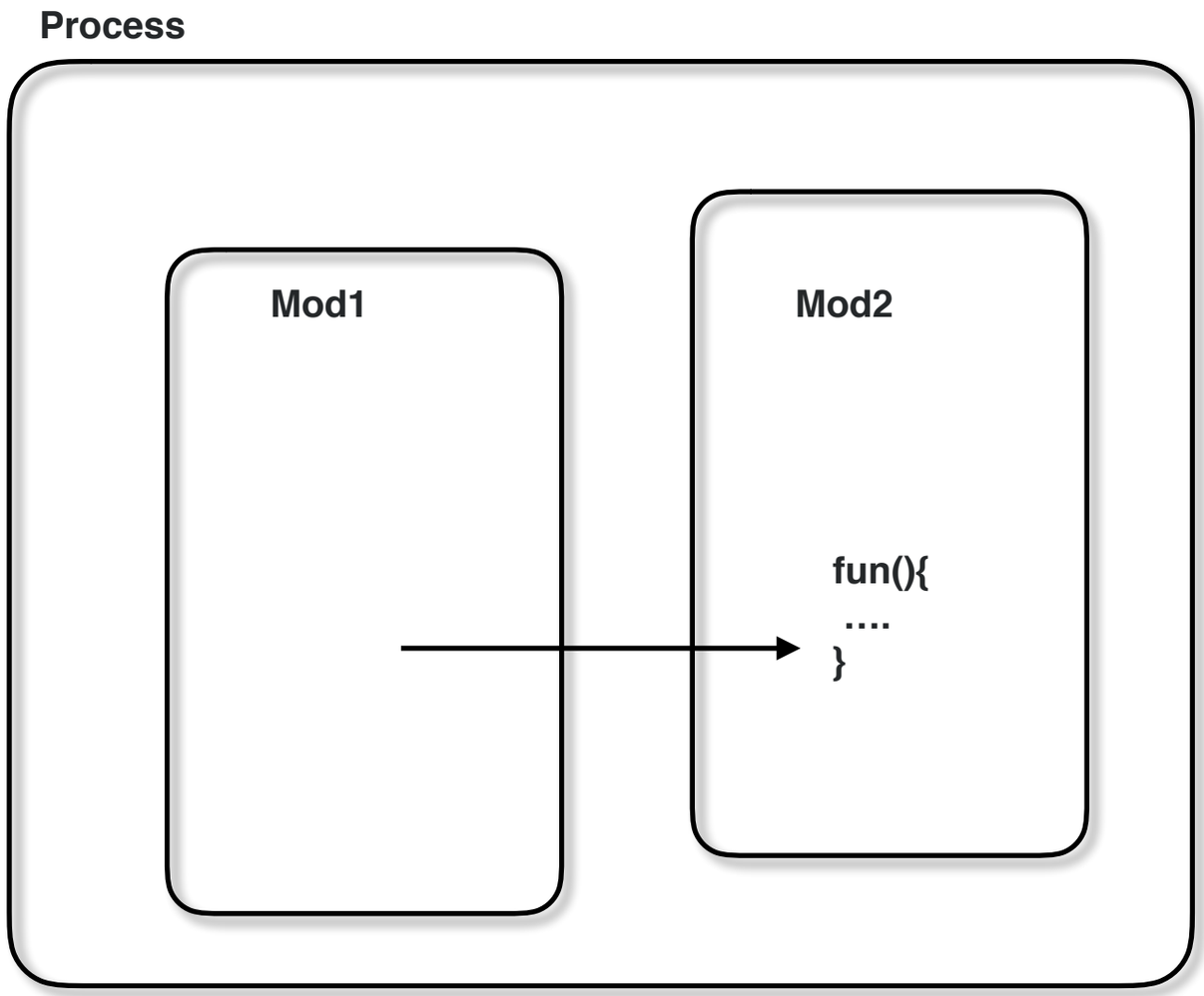




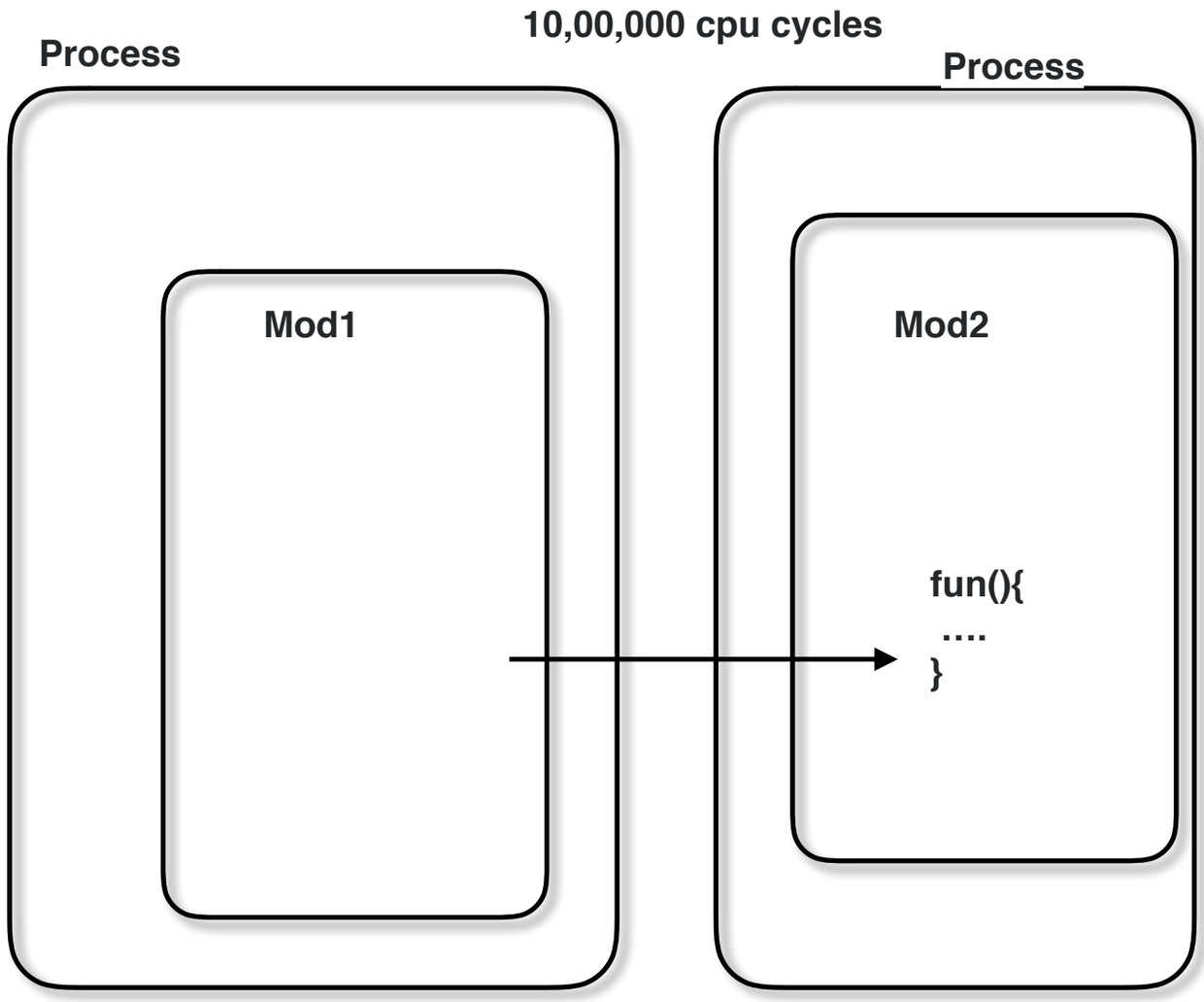
Micro Application

1. Performance

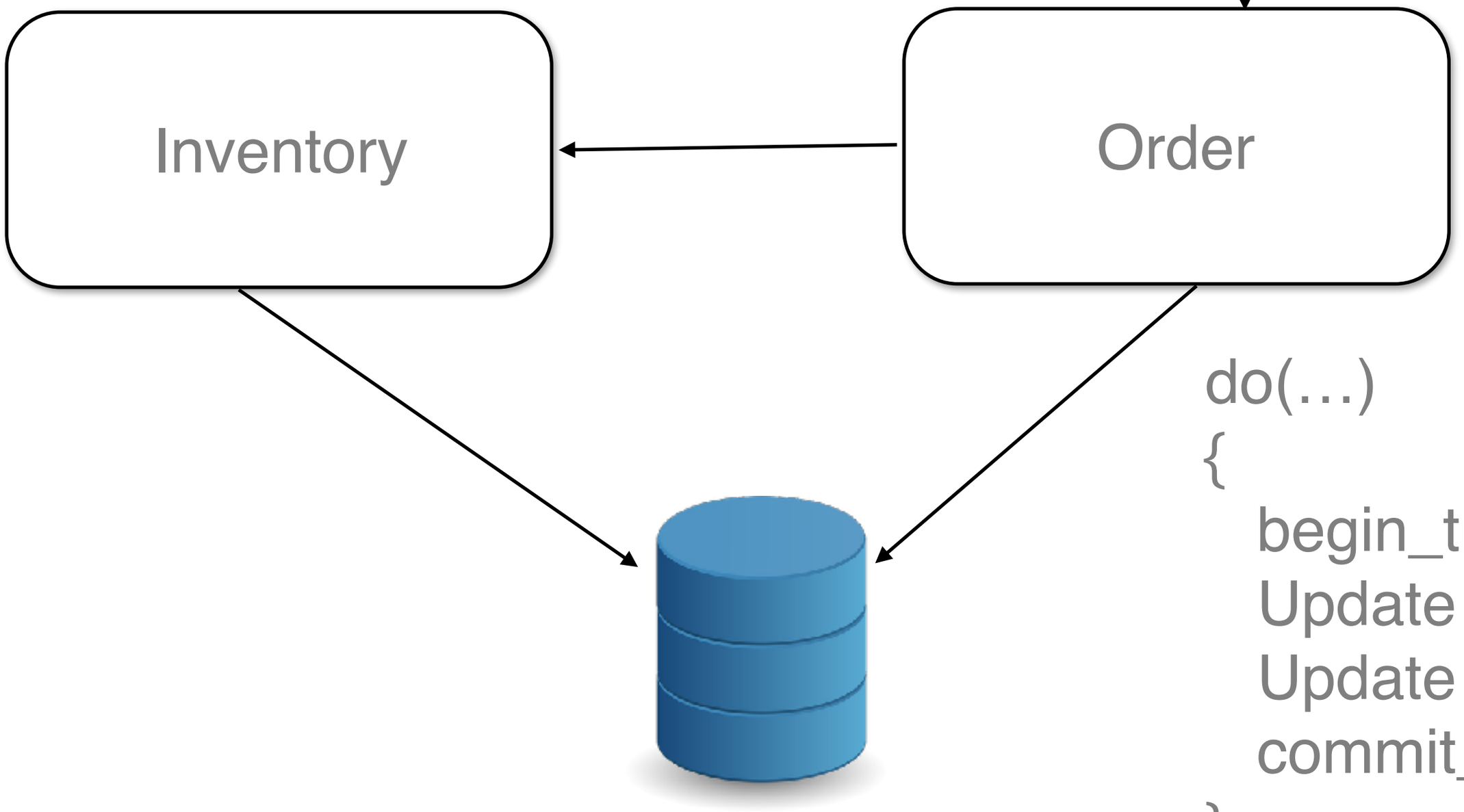
Operation	Cpu Cycles
• 10 + 12	3
• Calling a in-memory Method	10
• Create Thread	2,00,000
• Destroy Thread	1,00,000
• Database Call	40,00,000
• Distributed Fun Call	20,00,000
Write to disk	10,00,000



10 cpu cycles

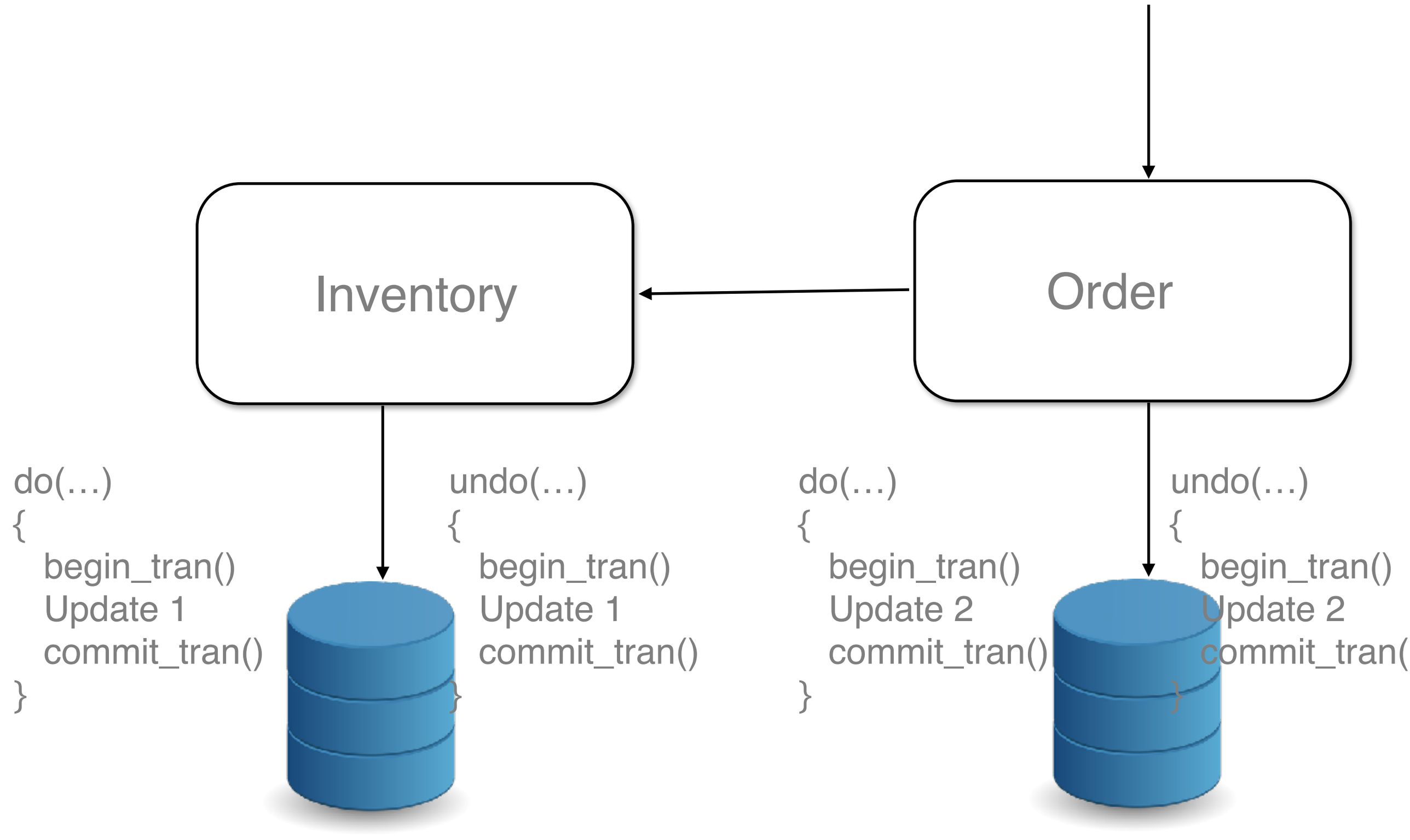


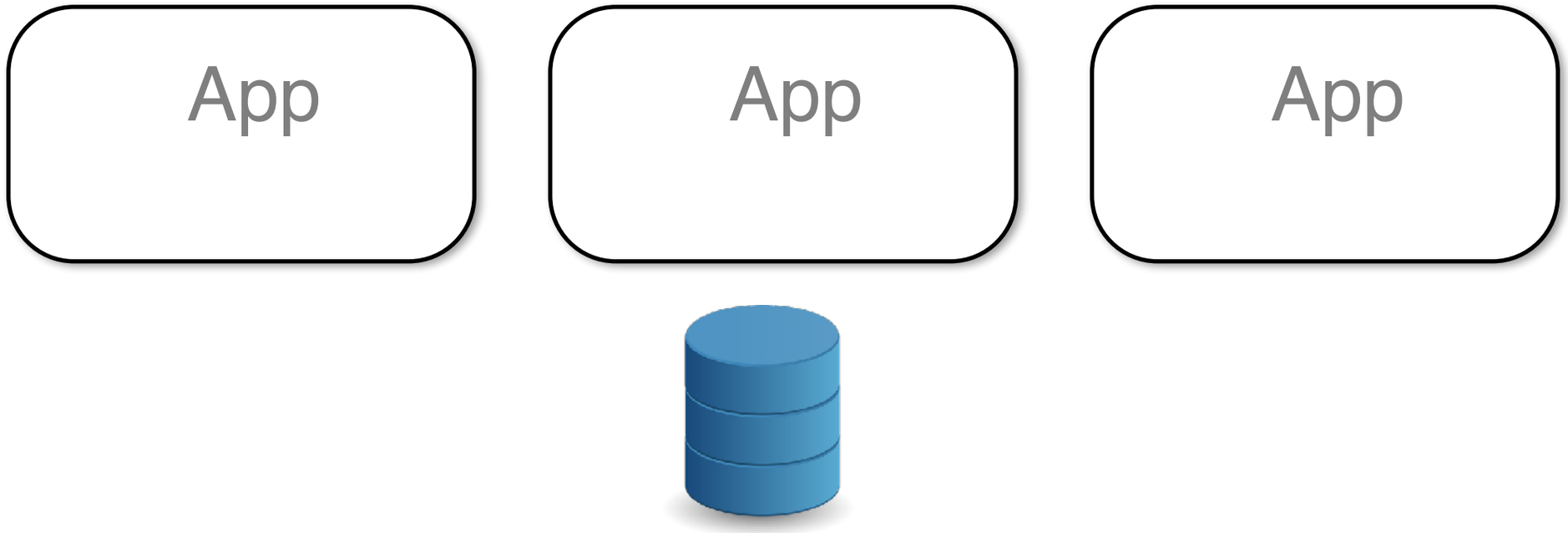
2. Db transaction



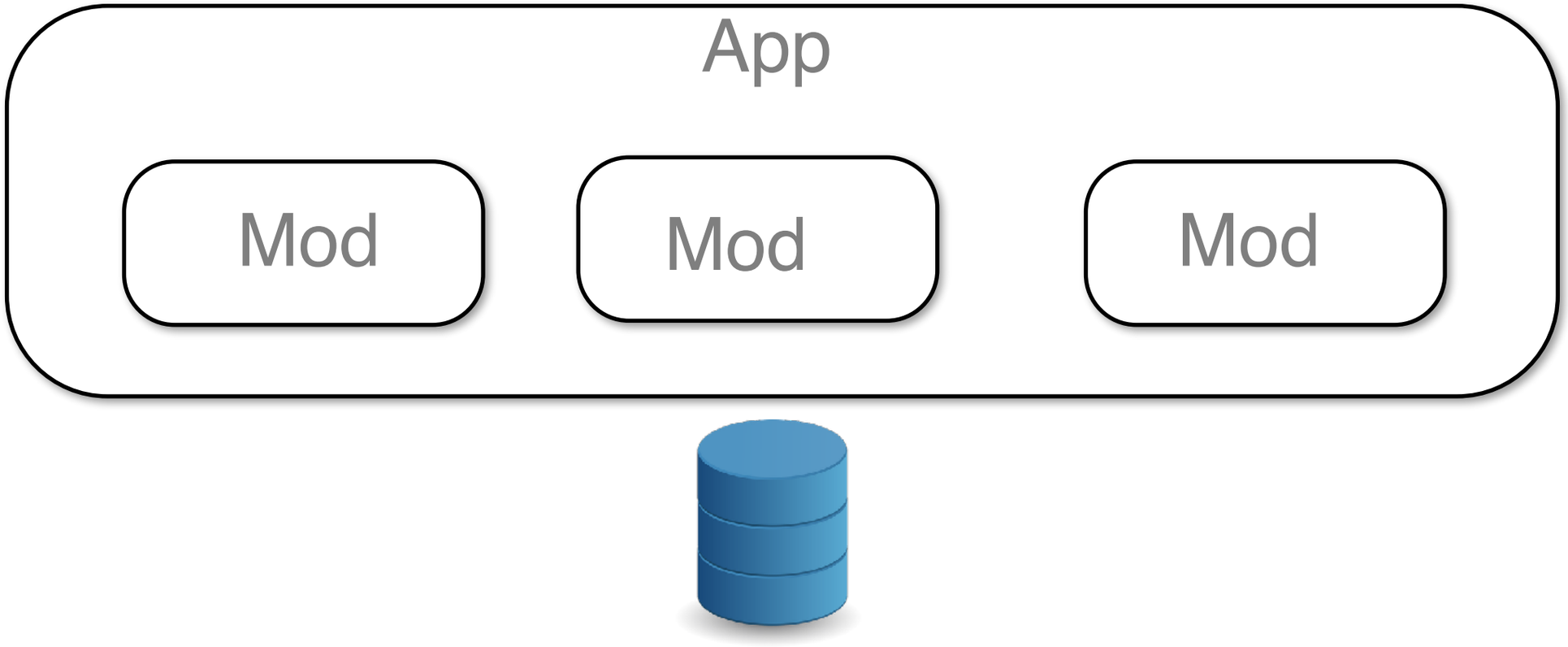

```
do(…)
{
   begin_tran()
   Update 1
   Update 2
   commit_tran()
}
```

2. Db transaction



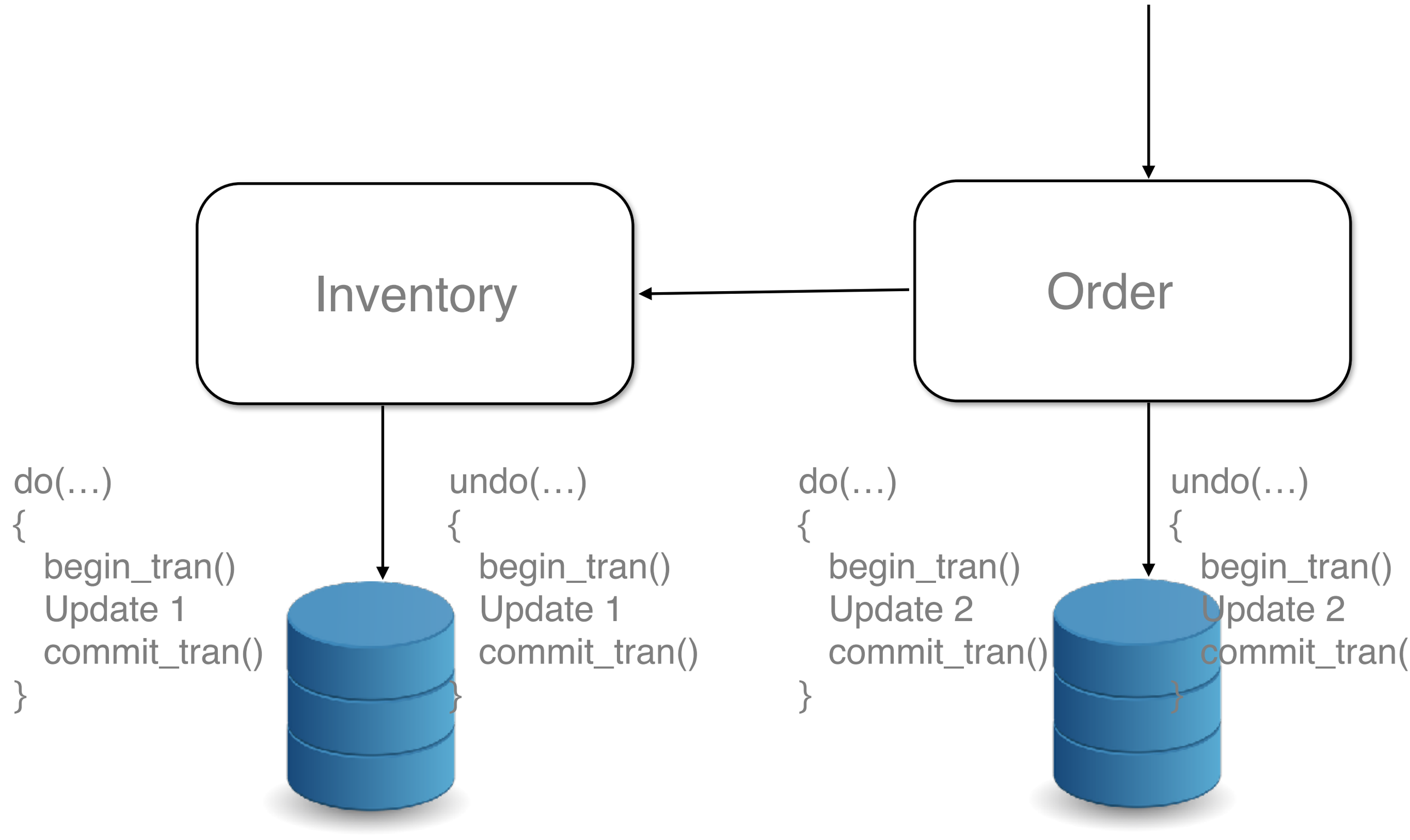

App
App
App

App
App
App


App
Mod
Mod
Mod

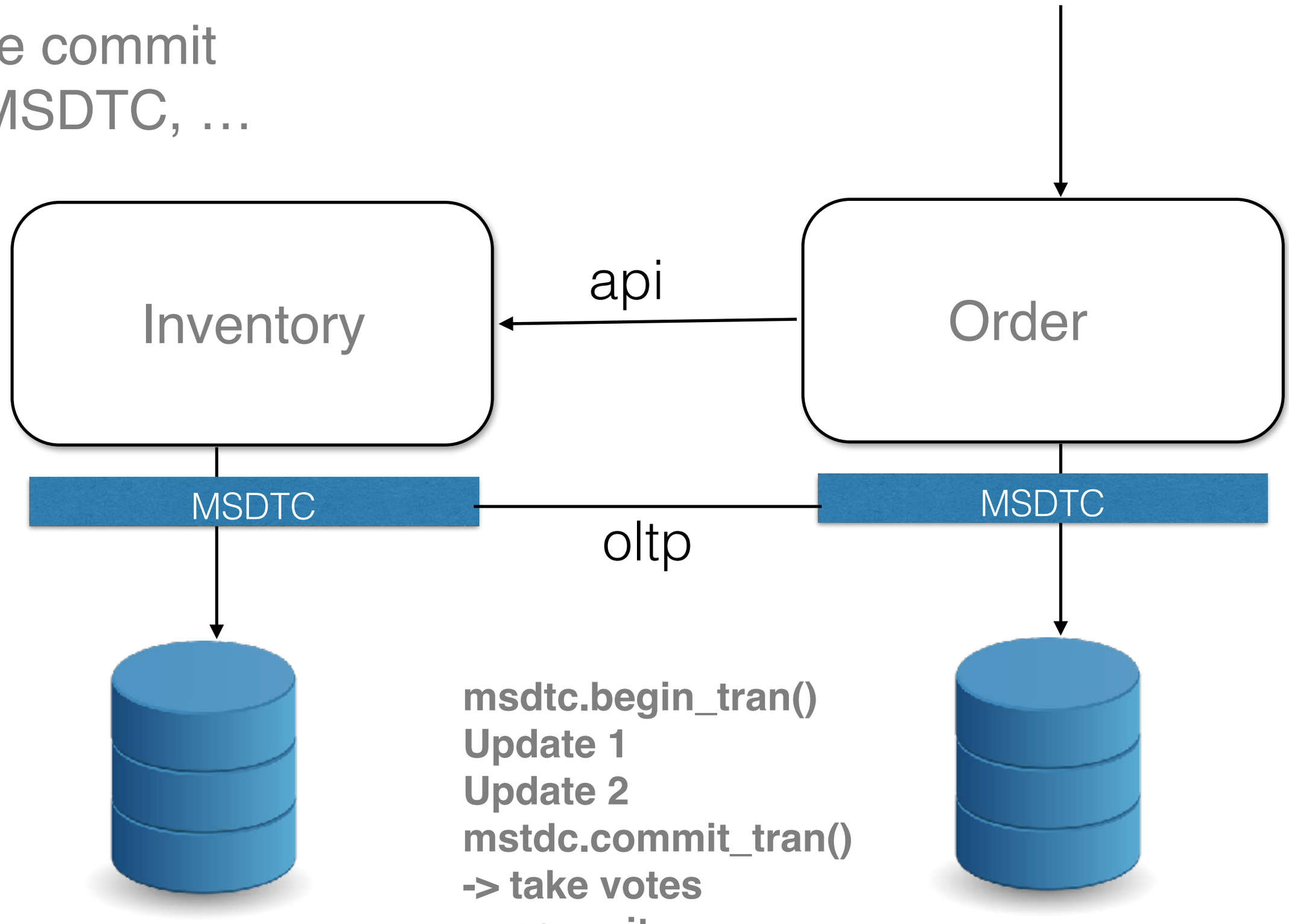

App
App
App

2. Db transaction



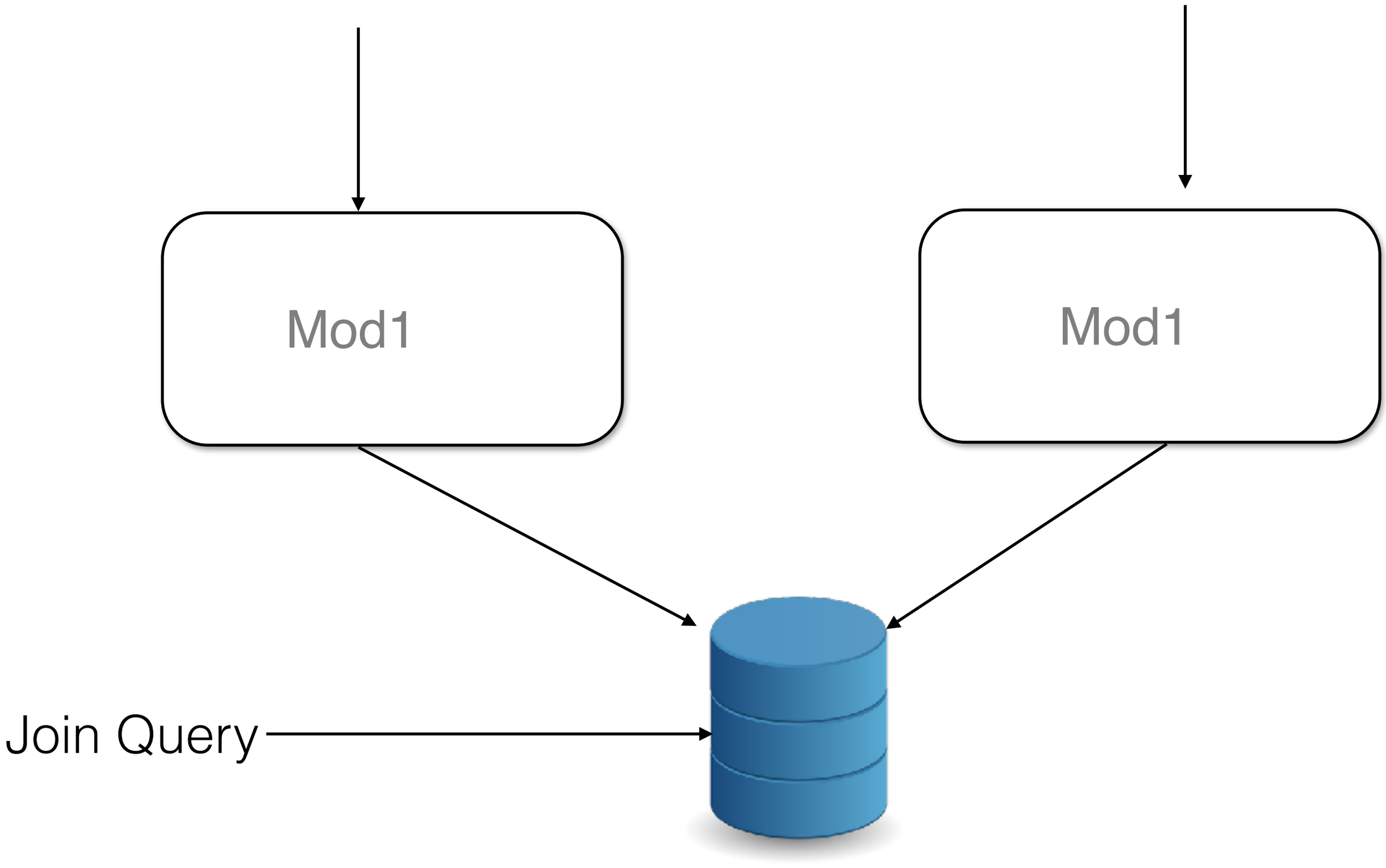
2. Db transaction

2 phase commit
JTX , MSDTC, …

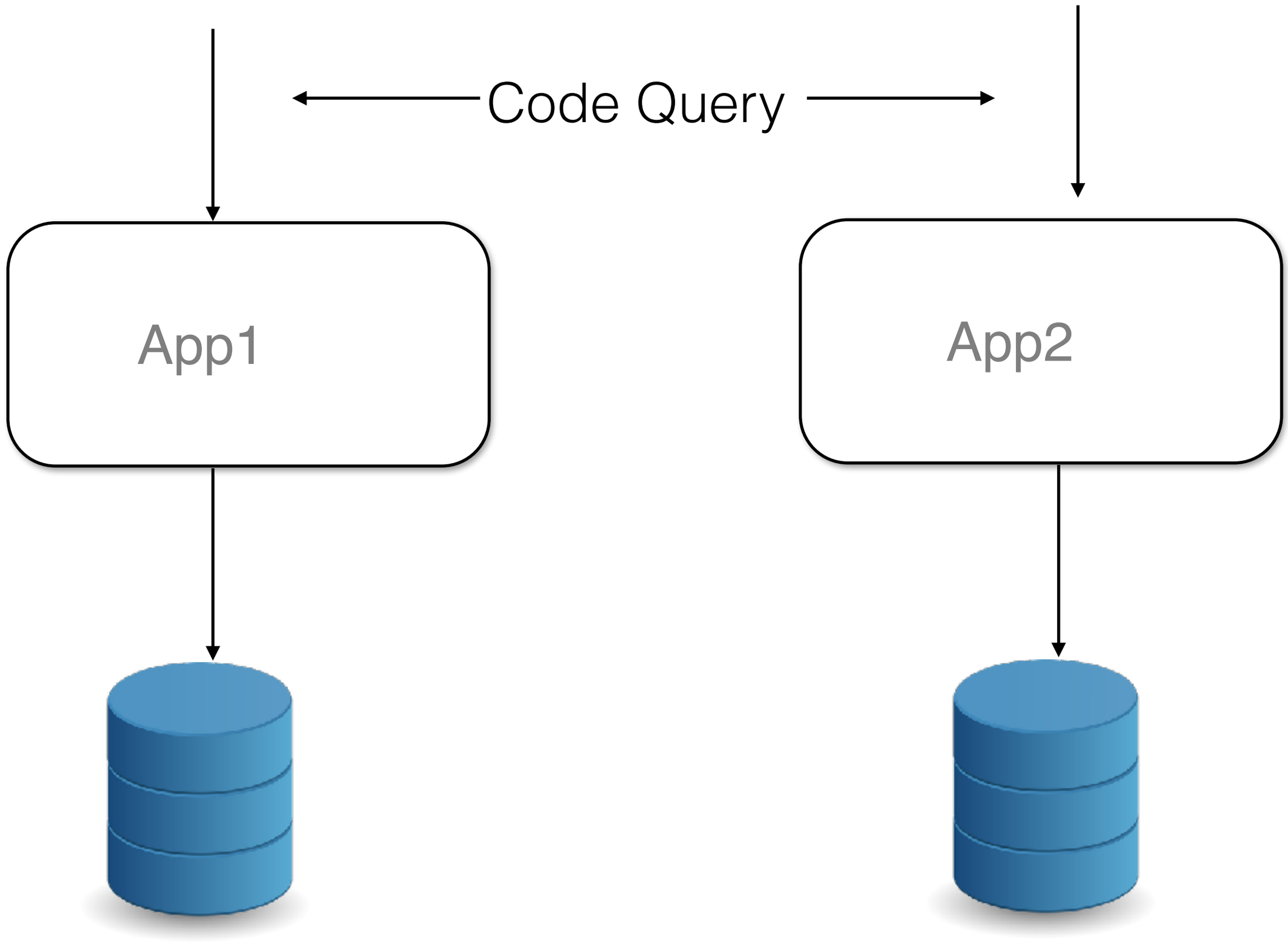


msdtc.begin_tran()
Update 1
Update 2
mstdc.commit_tran()
-> take votes
-> commit

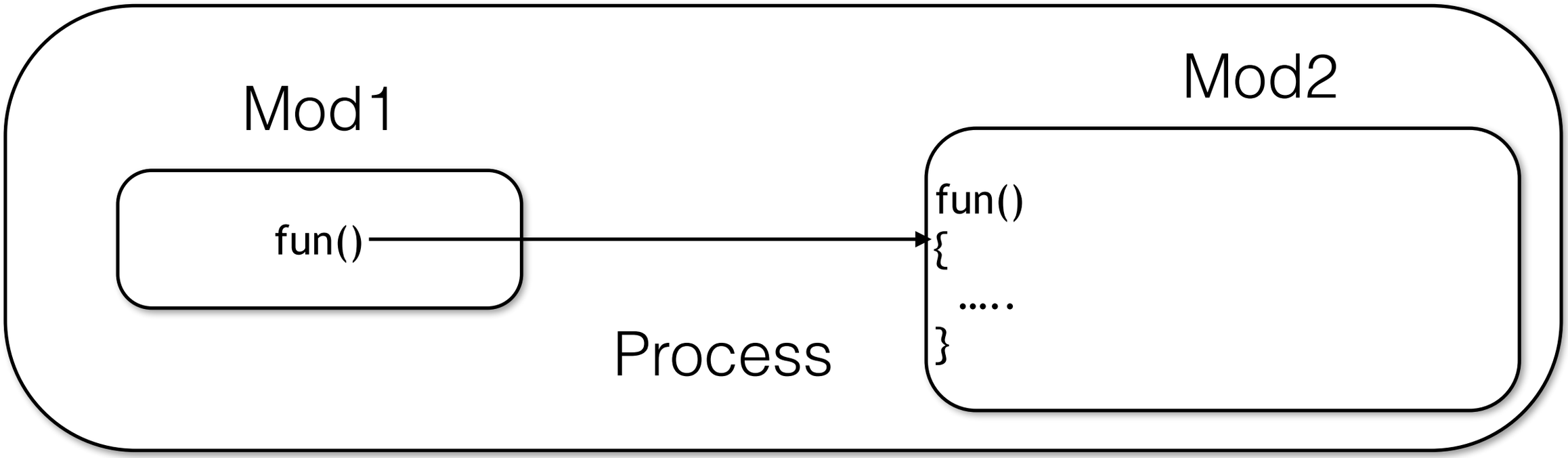
3. Db query

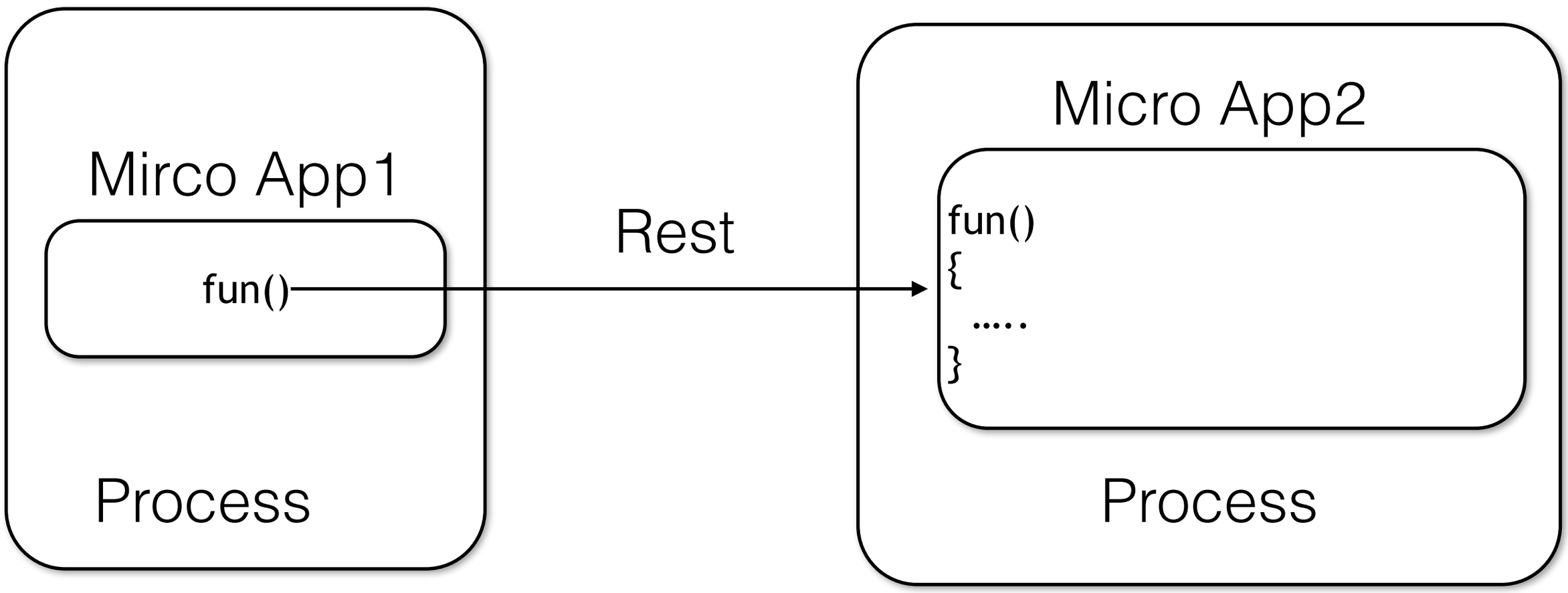


3. Query

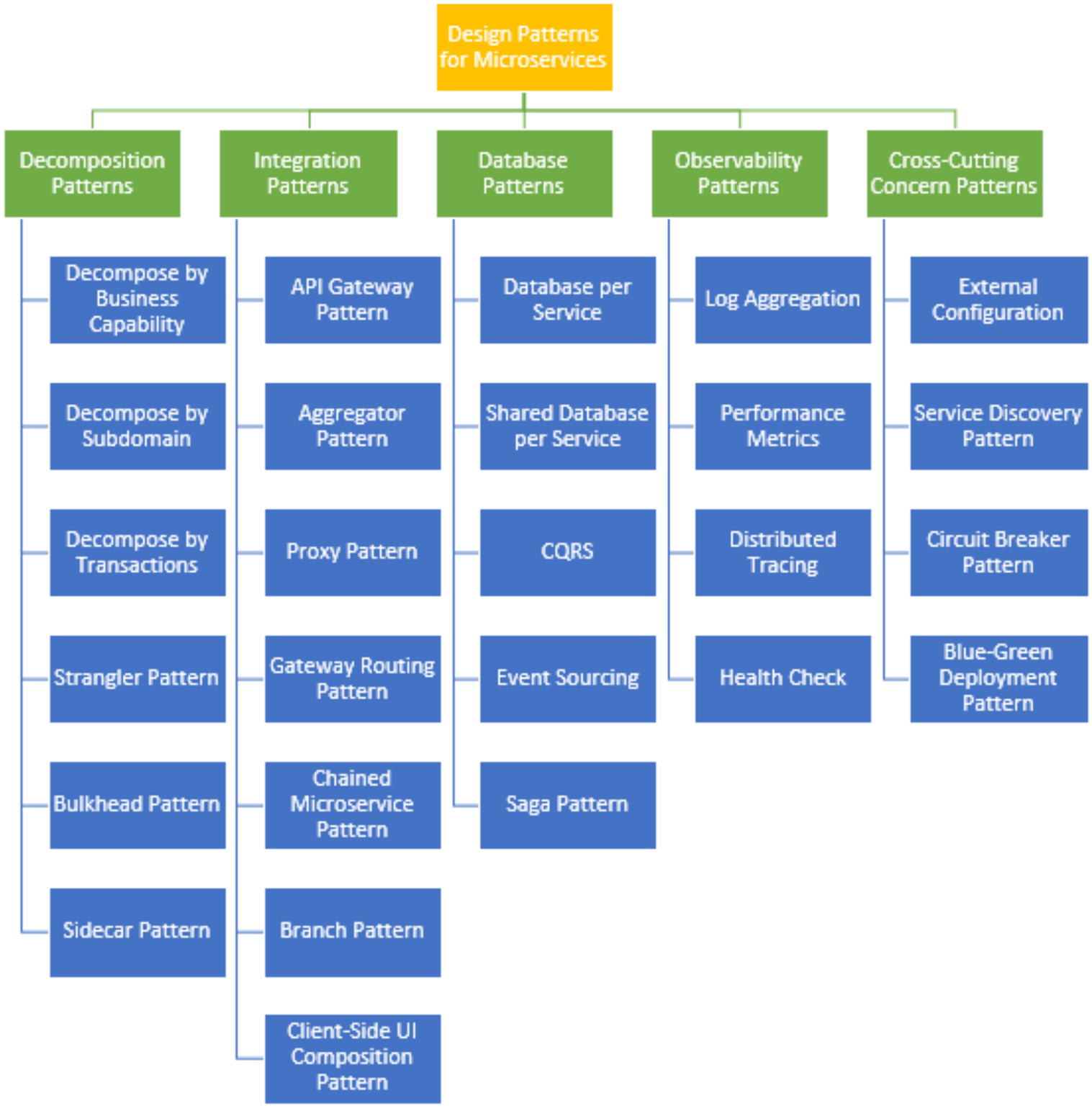


4. Development time

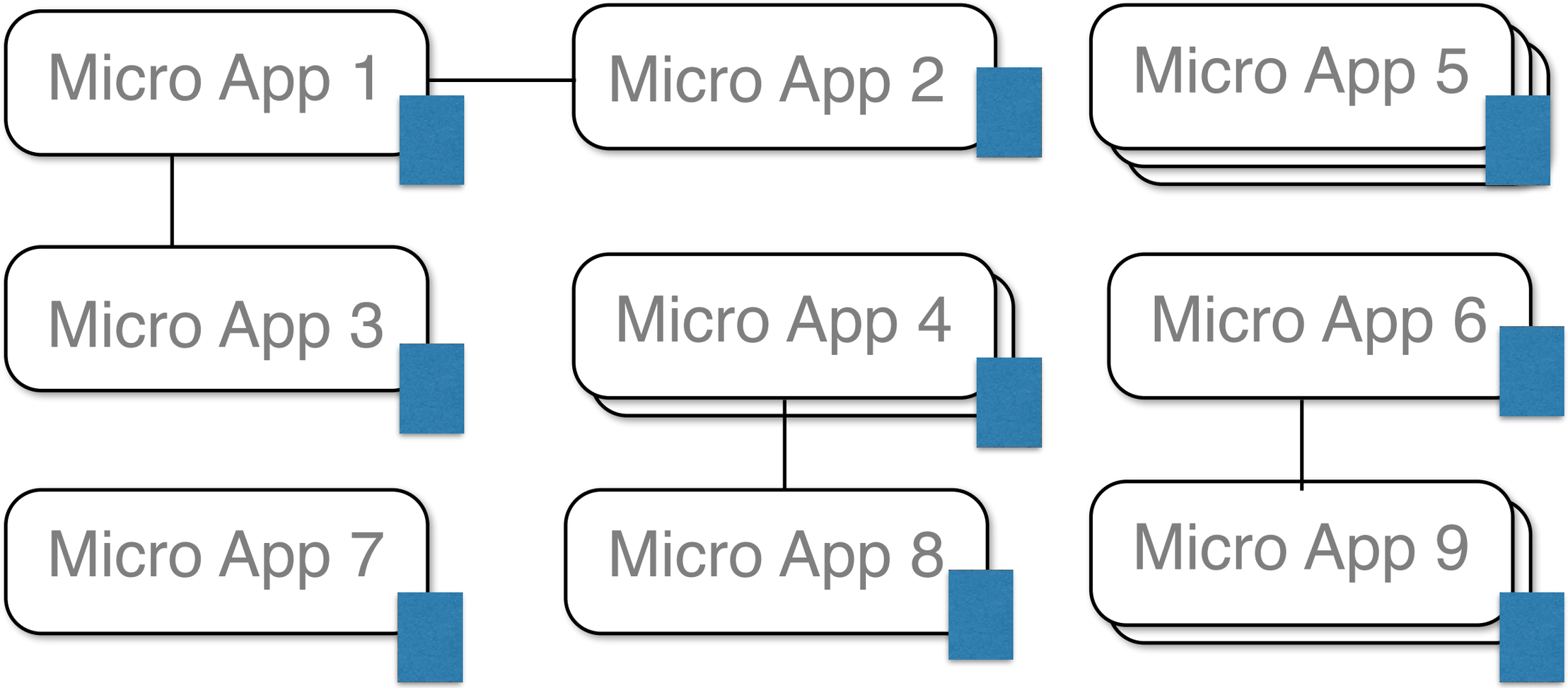




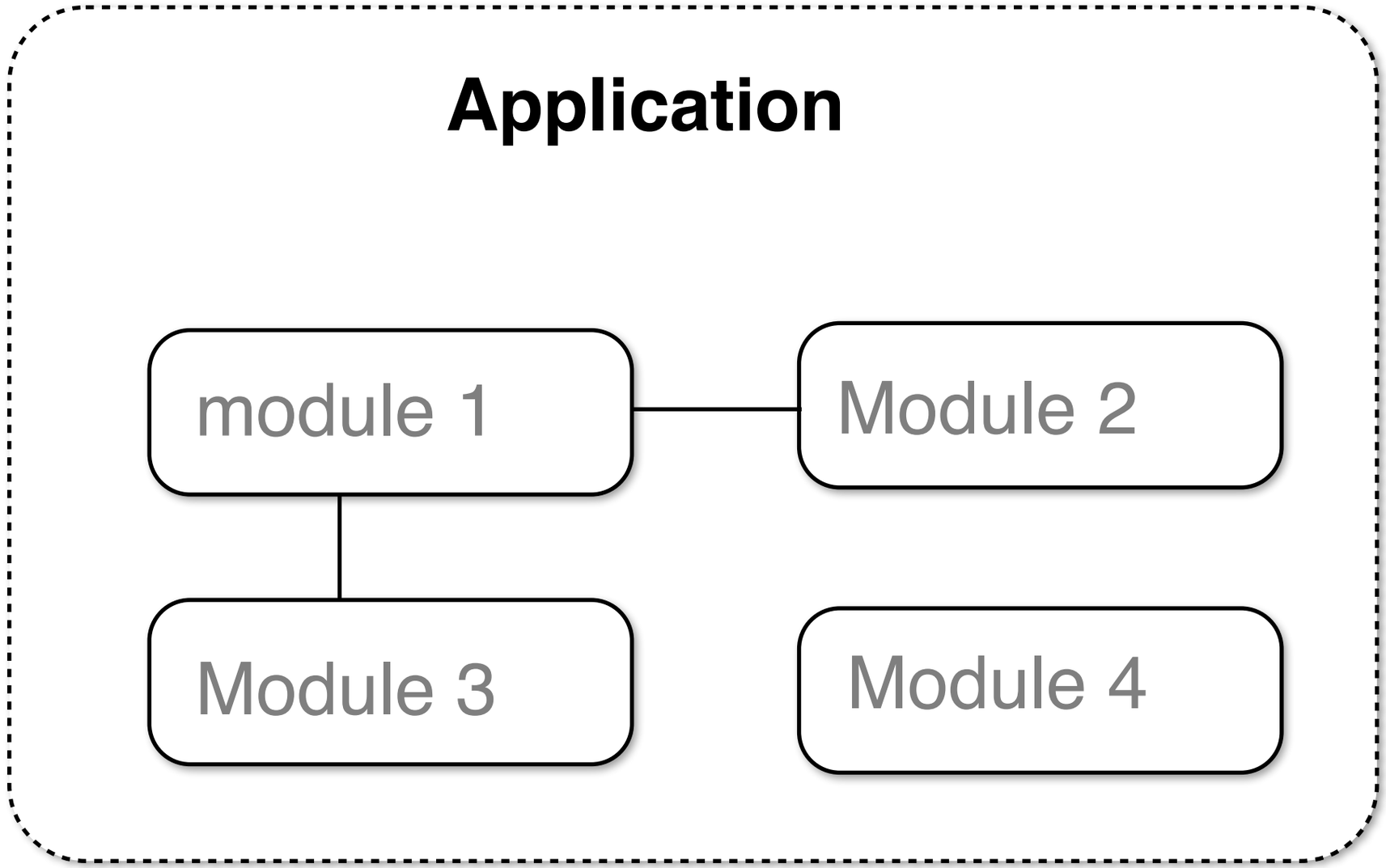
5. Learning Curve

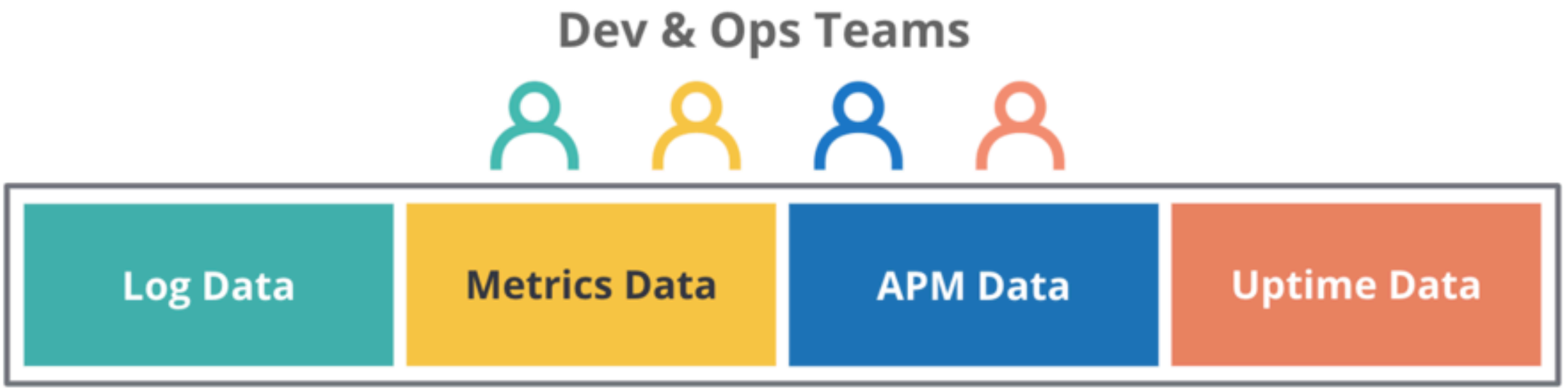


6. Infra Cost

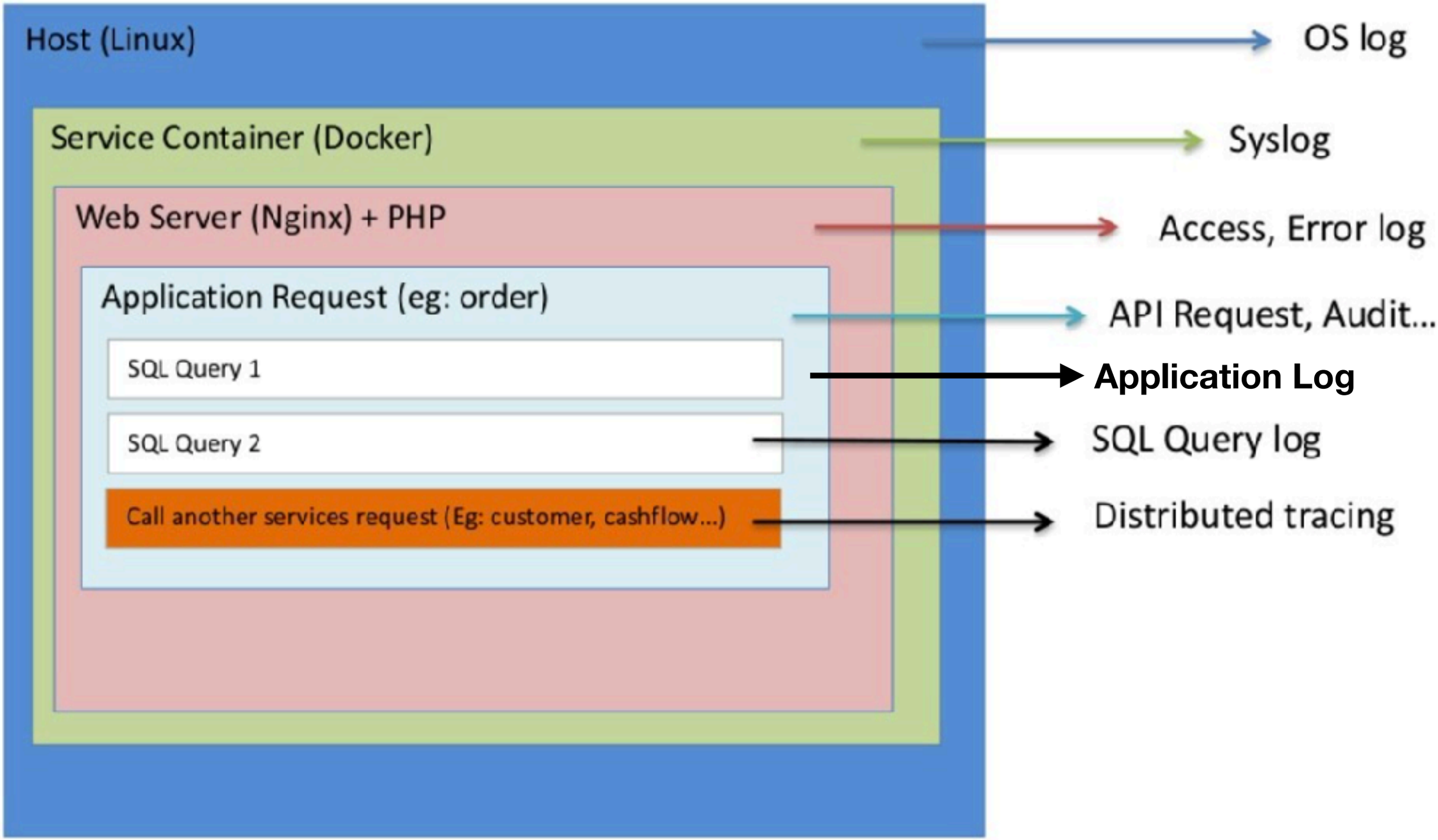

Micro App 1
Micro App 2
Micro App 5
Micro App 3
Micro App 4
Micro App 6
Micro App 7
Micro App 8
Micro App 9

7. Debugging

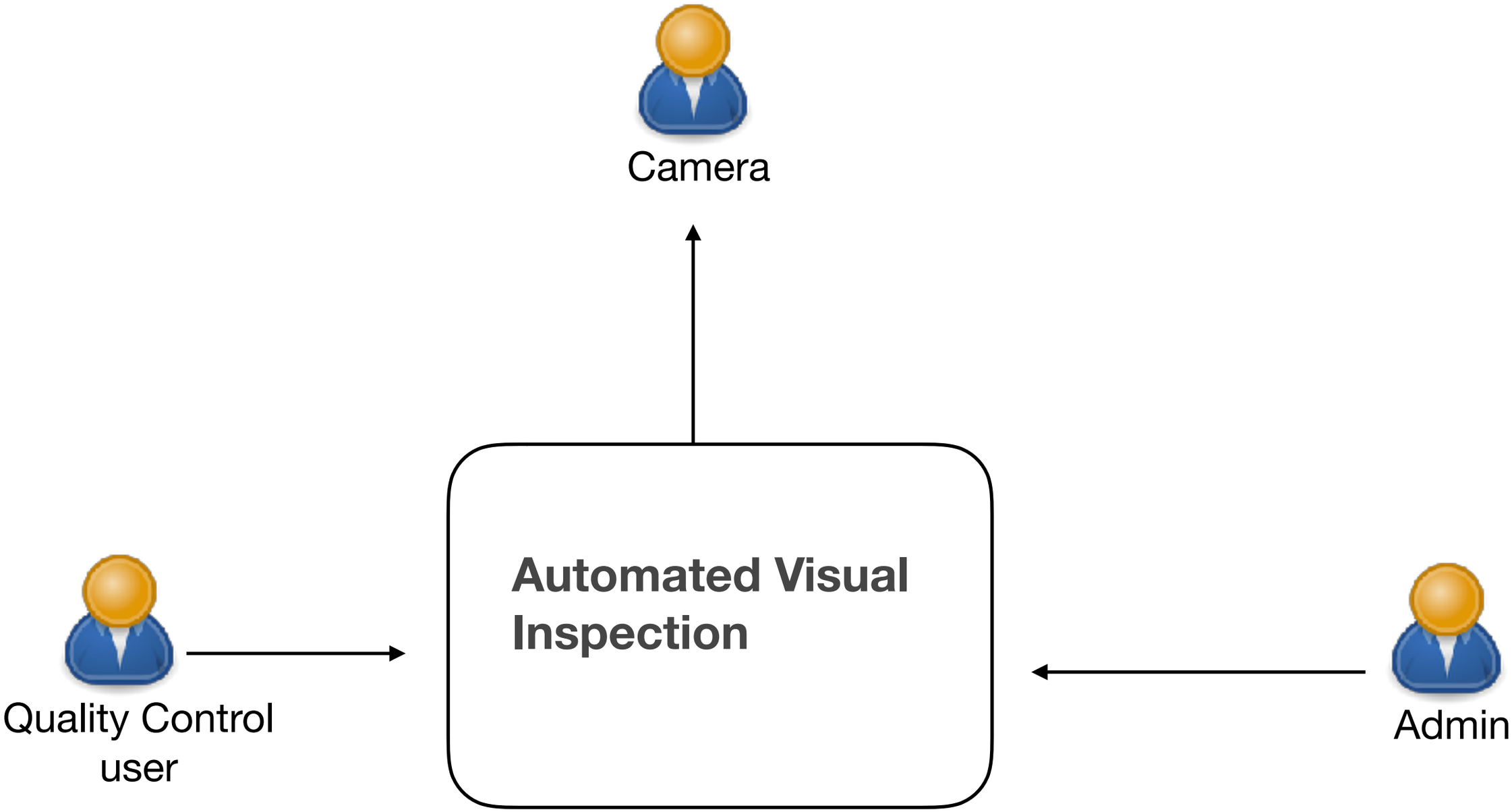




Web Logs	Container Metrics	Real User Monitoring	Uptime
App Logs	Host Metrics	Txn Perf Monitoring	Response Time
Database Logs	Database Metics	Distributed Tracing	
Container Logs	Network Metrics		
	Storage Metrics		


Host (Linux)
Service Container (Docker)
Web Server (Nginx) + PHP
Application Request (eg: order)
SQL Query 1
SQL Query 2
Call another services request (Eg: customer, cashflow...)
OS log
Syslog
Access, Error log
API Request, Audit...
Application Log
SQL Query log
Distributed tracing

Case study


Camera
Automated Visual Inspection
Quality Control user
Admin

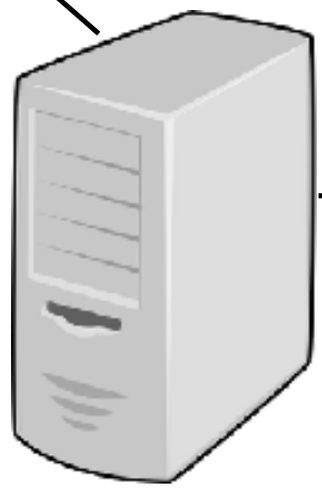
Approach 1

UI
API
Report generation
Defect identification

Data Compute Server

Capture
Send

Quality Control user

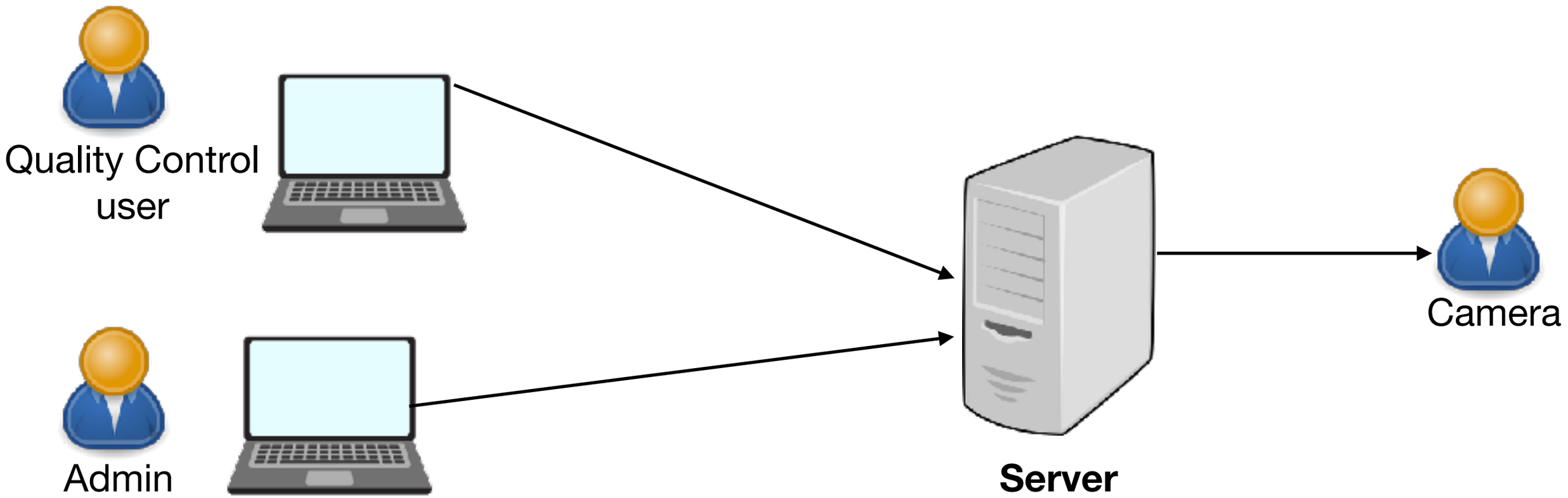


Camera

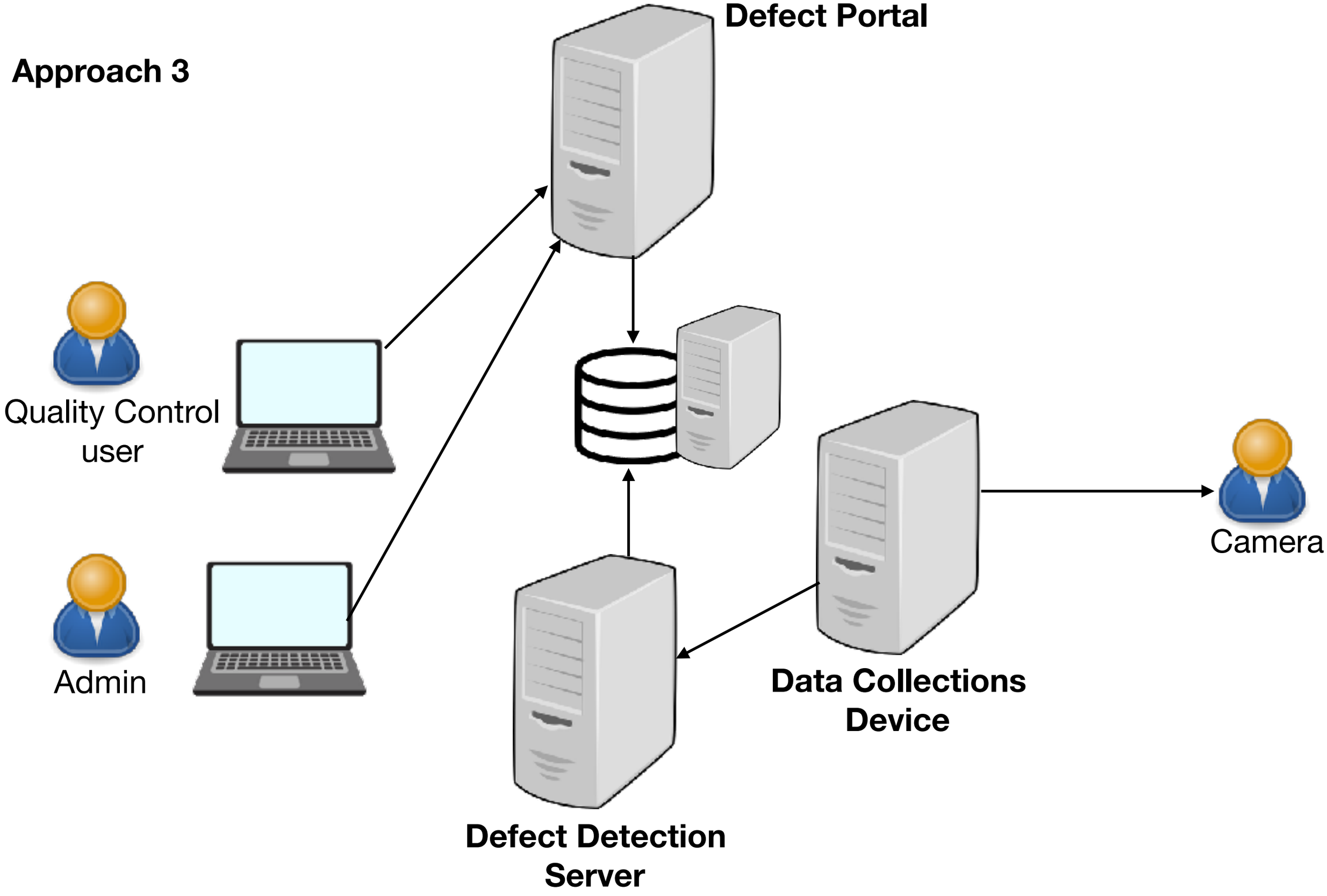
Admin

Data Collections Device

Approach 2



Approach 3

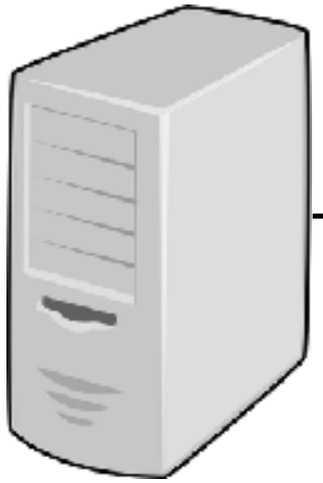

Defect Portal
Quality Control user
Admin
Camera
Data Collections Device
Defect Detection Server

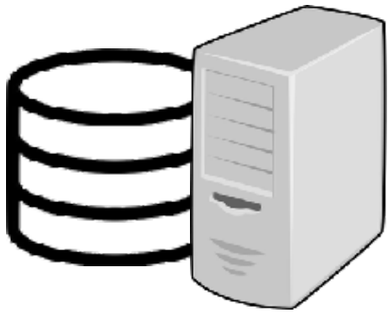
Approach 4

UI
API
Report generation

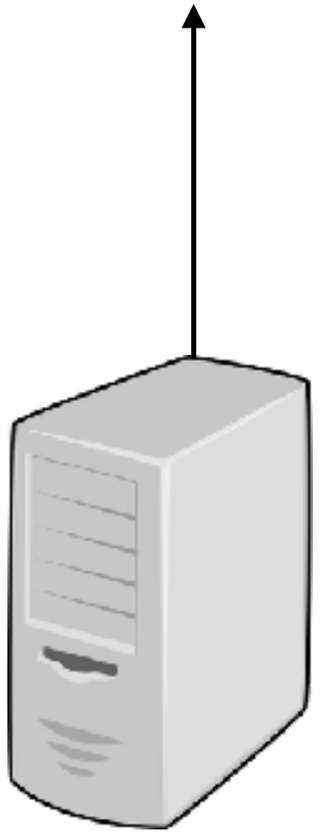
Web Portal

Quality Control user

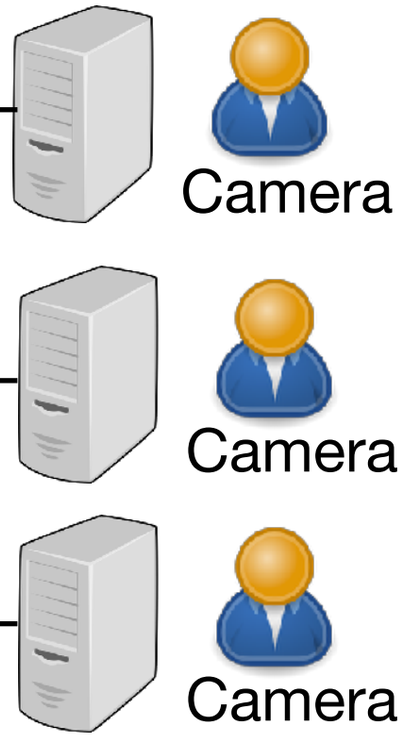




Admin



Defect Detection Server



Data Collections Device

Approach 4

UI Portal

API App

Report Generation App

Quality Control user

Admin

http: Rest

Getconfig

?

?

Quality Check results

?

Camera

Camera

Camera

Defect Detection App

Data Collection App

System

UI Portal

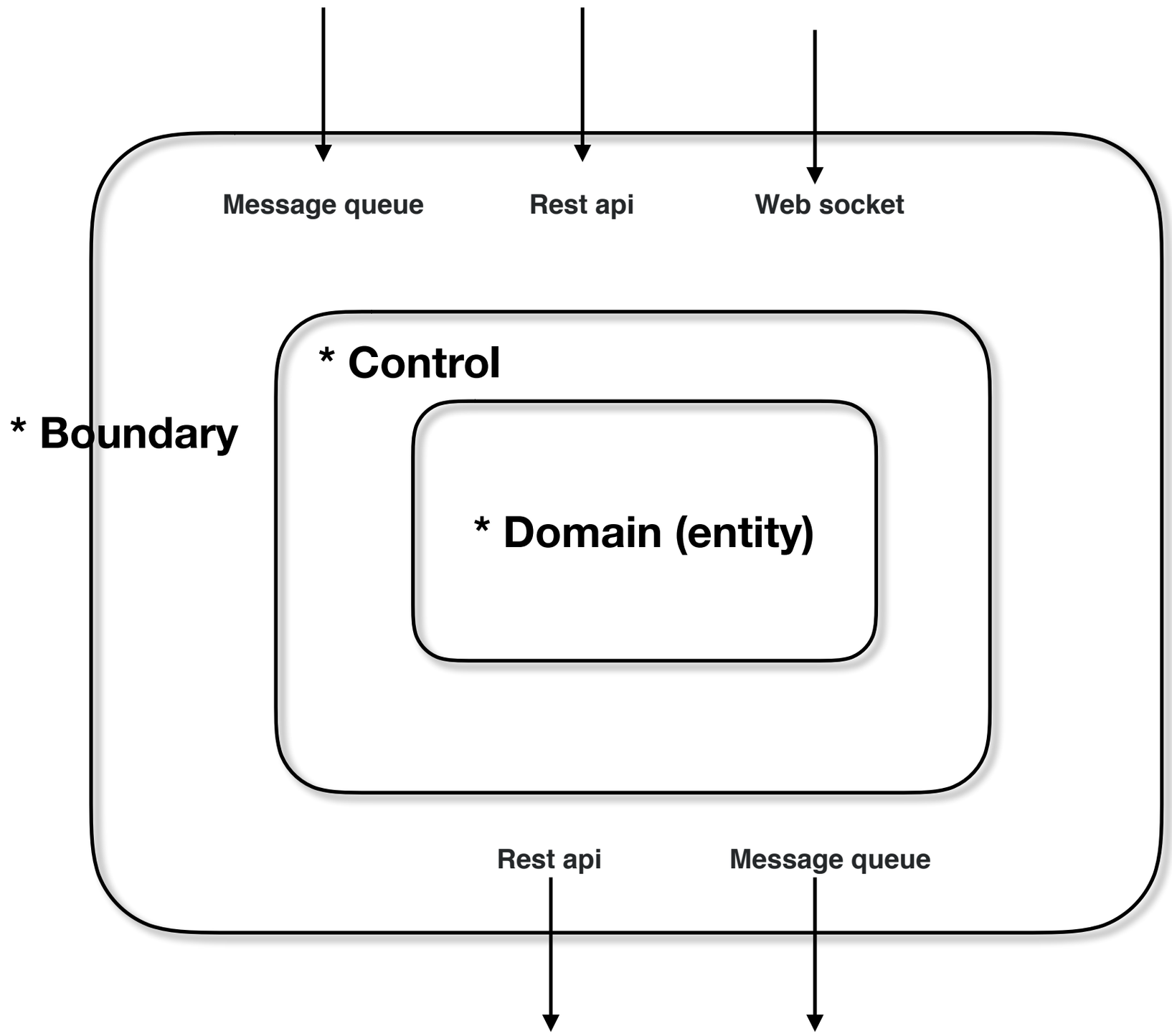
API App

Defect Detection App

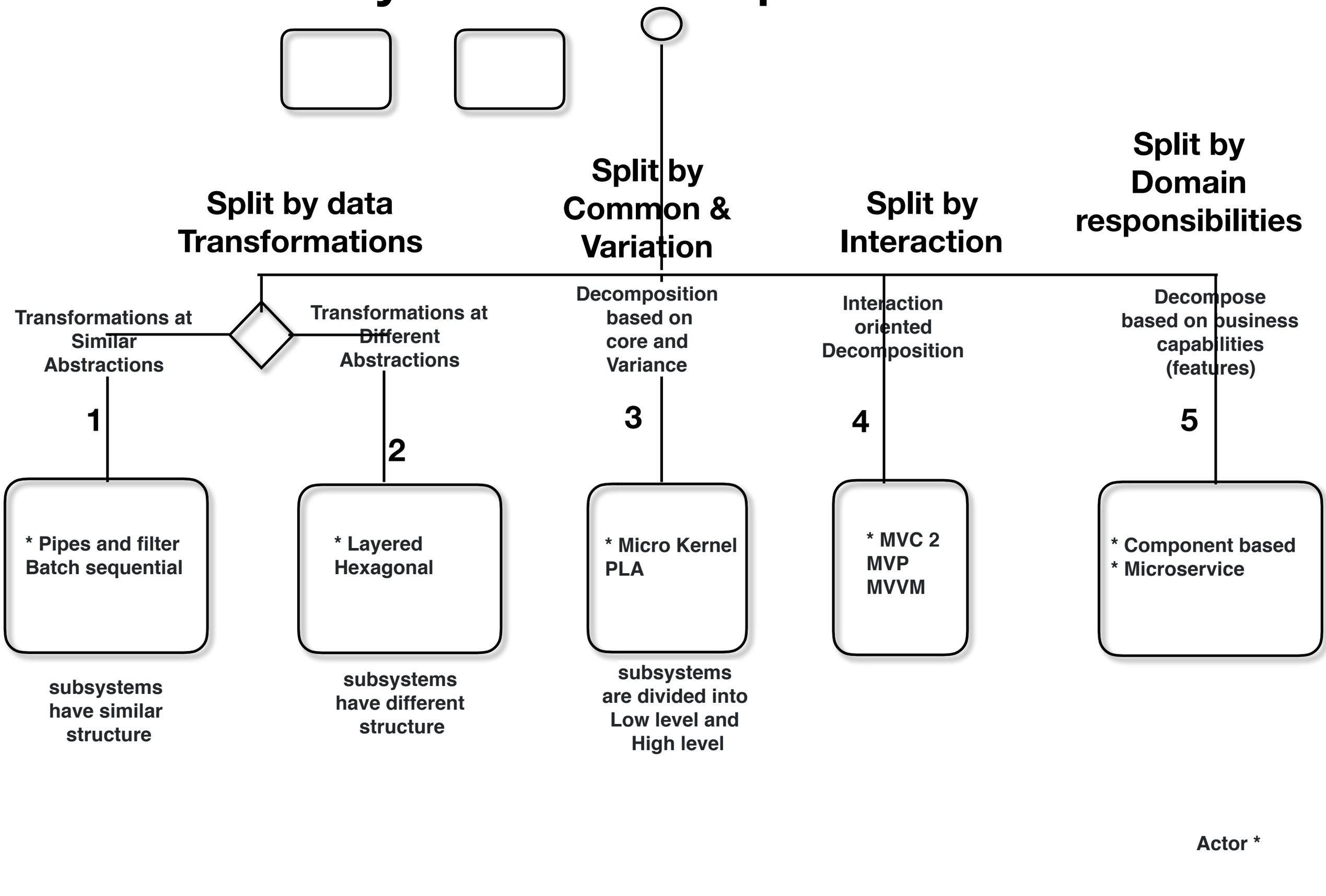
Report Generation App

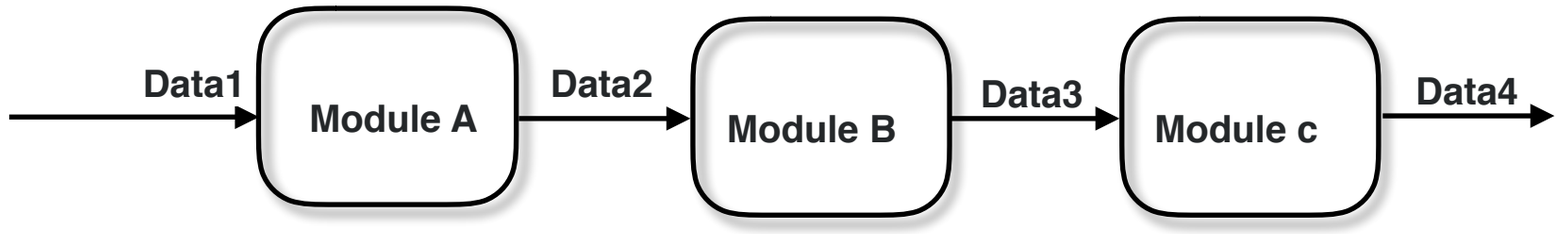
Data Collection App

* Boundary control entity

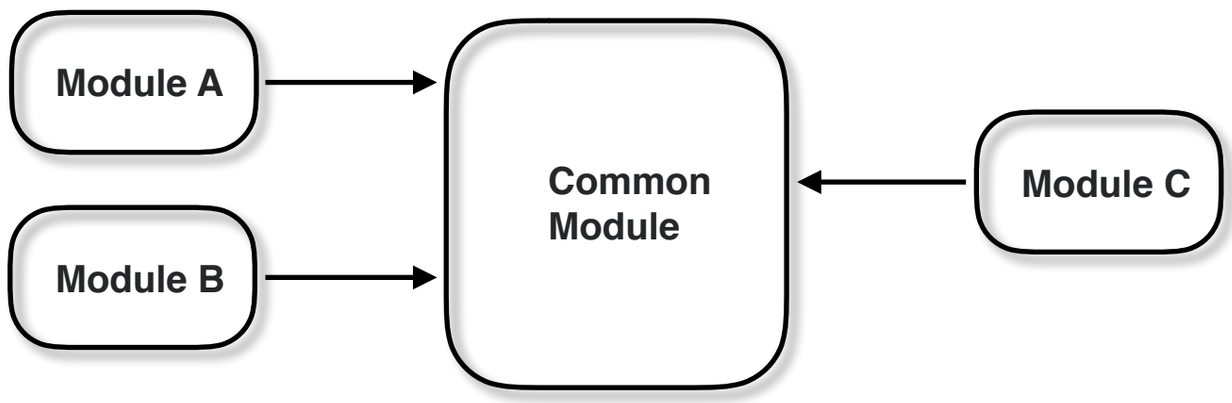


Choose System Decomposition Patterns





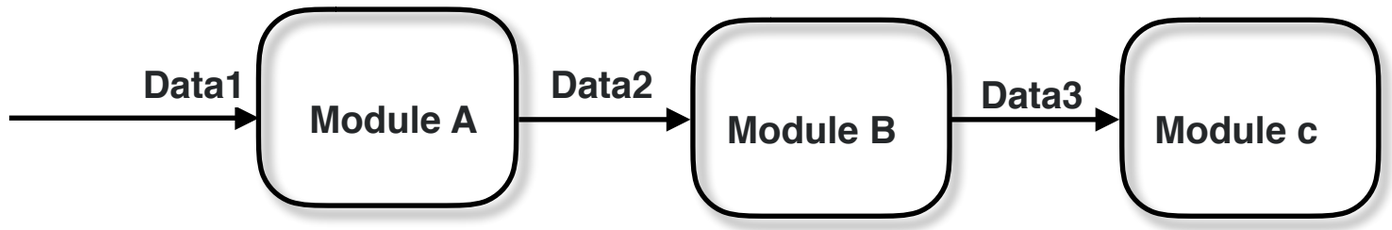
Split by data Transformations



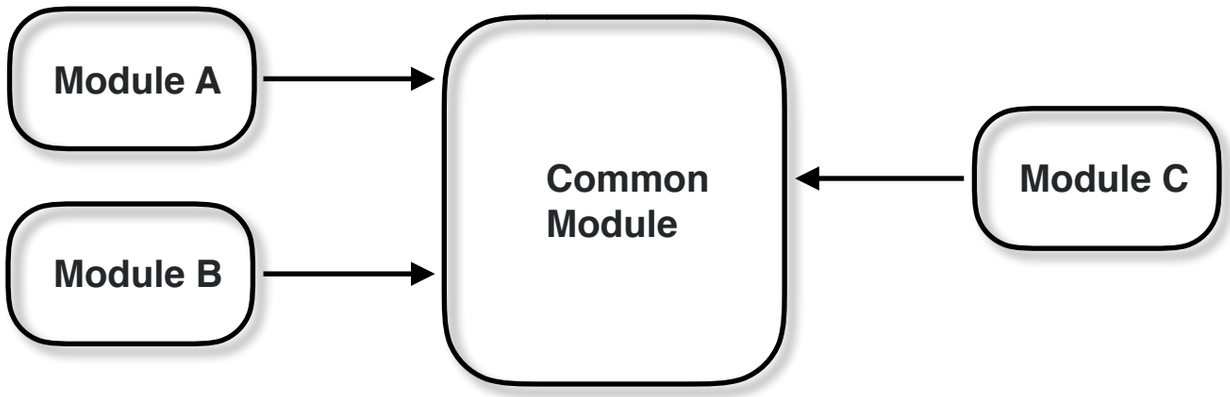
Split by Common & Variation

Split by Interaction

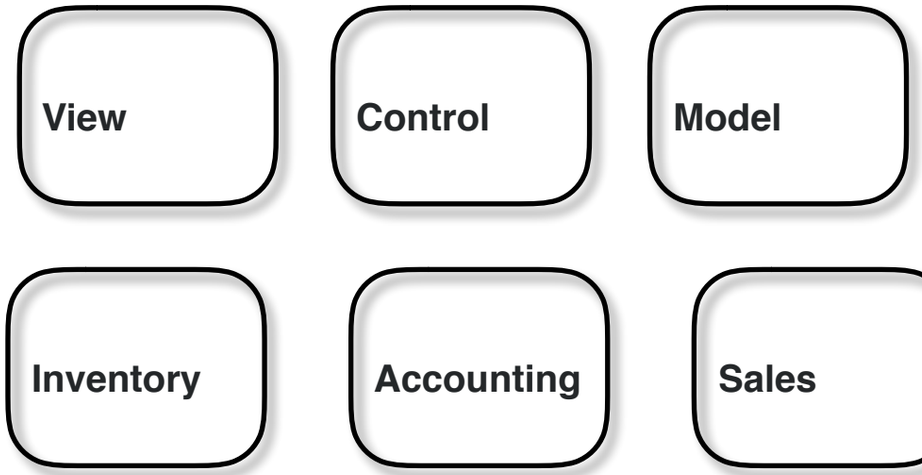
Split by Domain responsibilities



Split by data Transformations



Split by Common & Variation

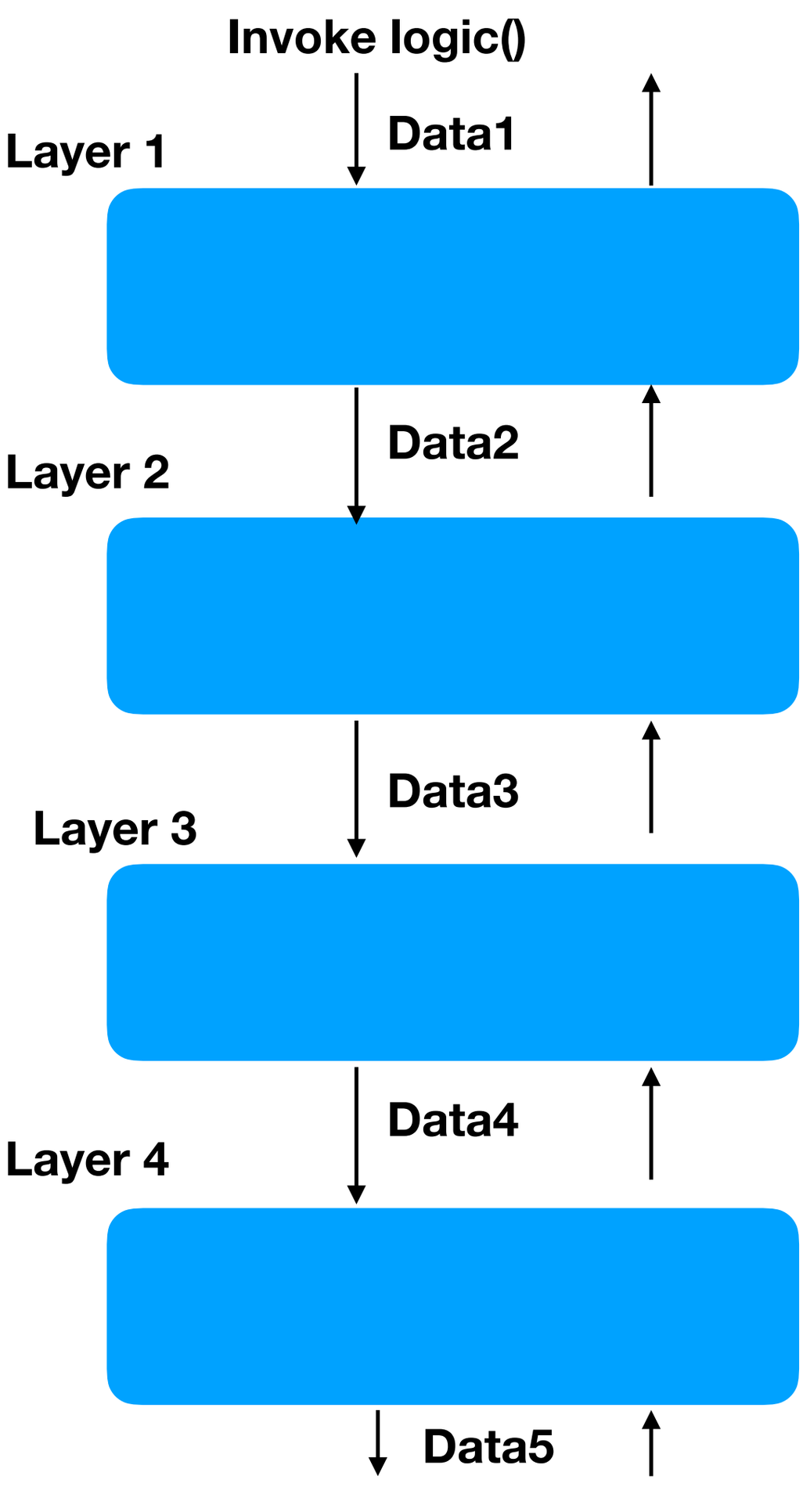


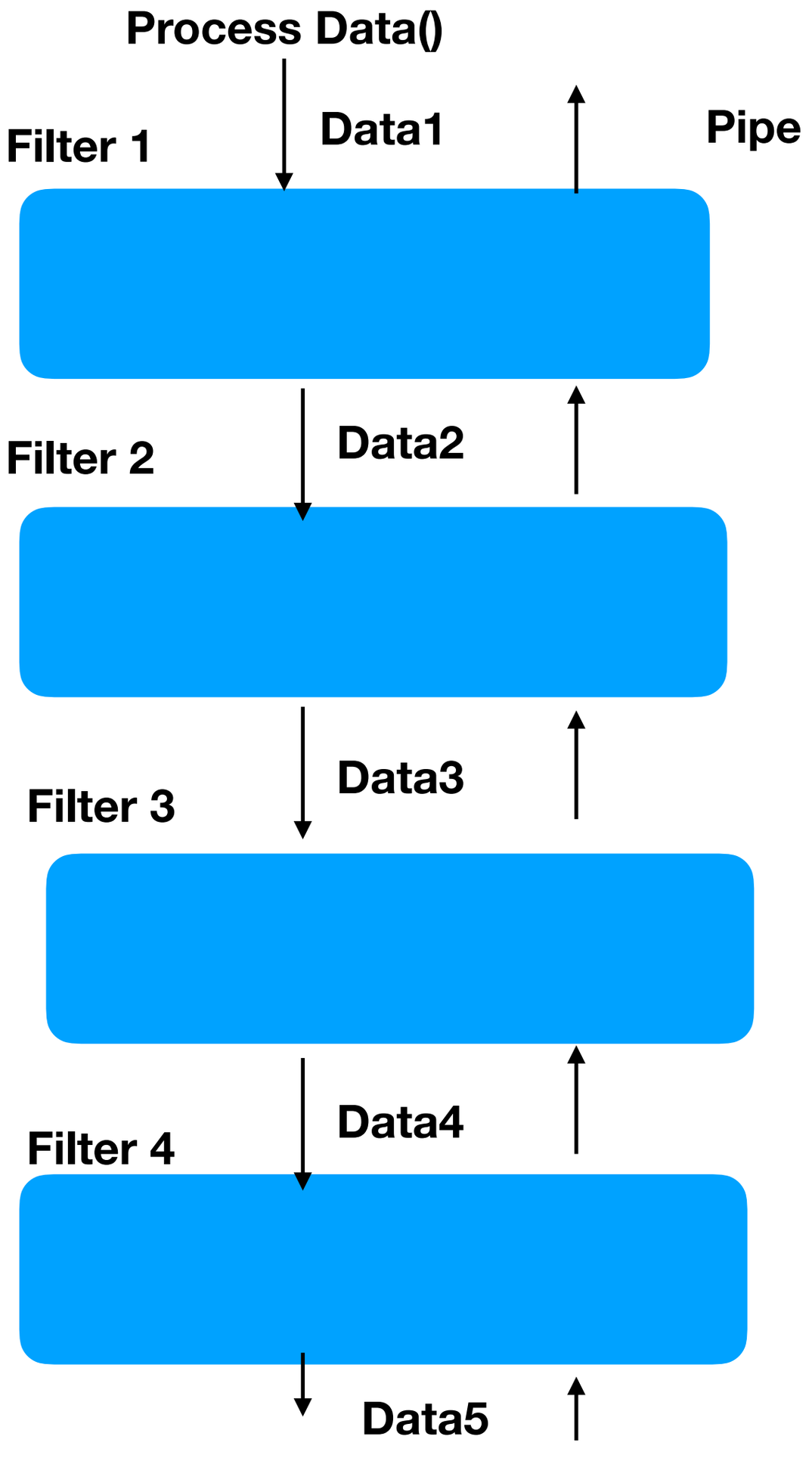
Split by Interaction

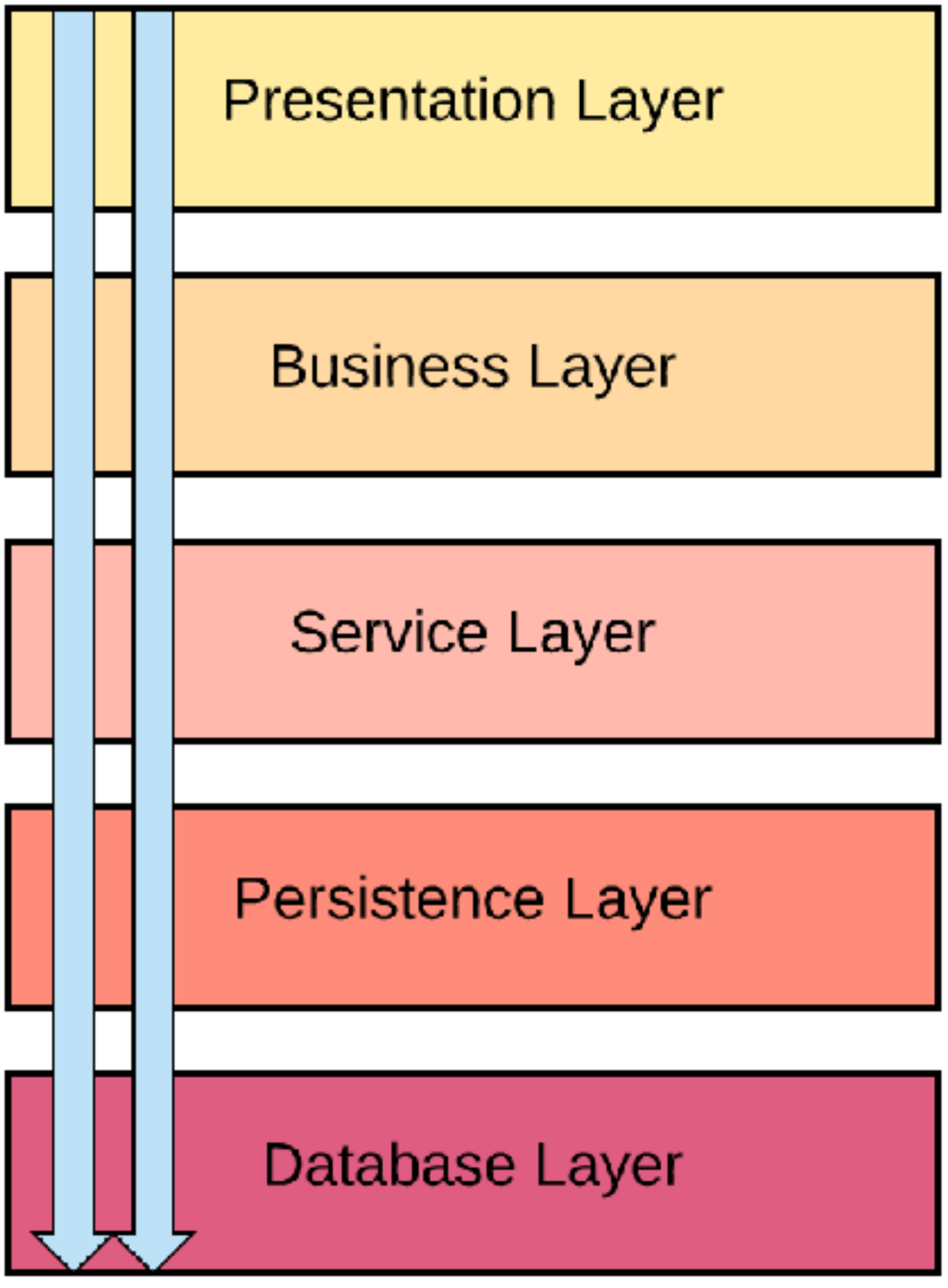
Split by Domain responsibilities

An application is required to perform a variety of tasks of varying complexity on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, **could change** as business requirements are updated. Also, **additional processing** might be required in the future, or the **order** in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

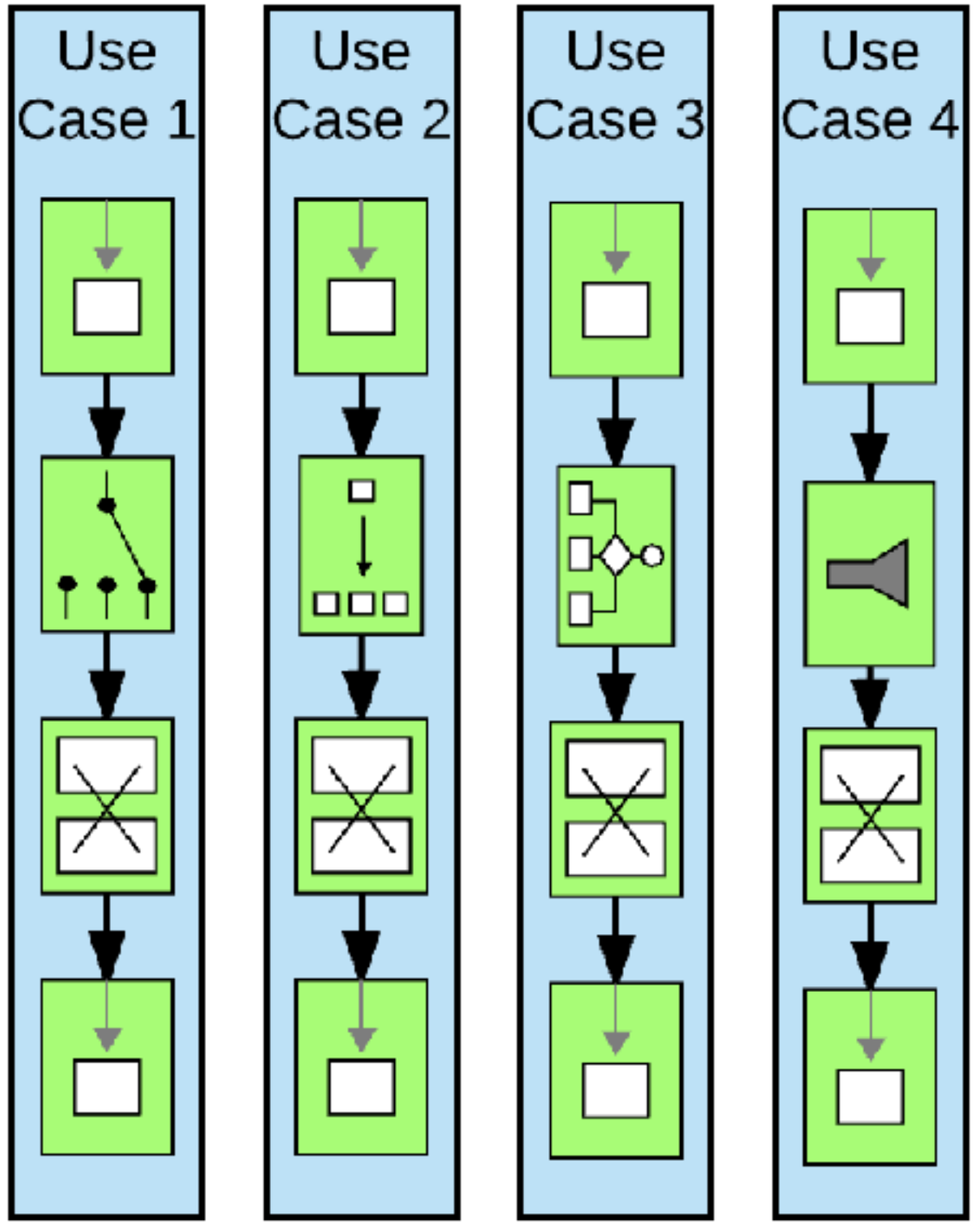
Eclipse IDE. Downloading the basic Eclipse product provides you little more than a fancy editor. However, once you start adding plug-ins, it becomes a highly customizable and useful product.


Invoke logic()
Layer 1
Data1
Data2
Layer 2
Data3
Layer 3
Data4
Layer 4
Data5


Process Data()
Filter 1
Data1
Pipe
Filter 2
Data2
Filter 3
Data3
Filter 4
Data4
Data5



Layered Architecture



Pipes and Filters

Invoke logic()

Data1

Layer 1

f1(int) f2() f3(int,int)

Data2

Layer 2

f3(int,int)

f1(bool) f4()

f2(double)

Data3

Layer 3

f1() f4()

f2(int) f3() f5(int)

Data4

Layer 4

f1(int) f2()

Data5

Process Data()

Data1

Pipe

Filter 1

f1(int)

Data2

Filter 2

f2(int)

Data3

Filter 3

f3(int)

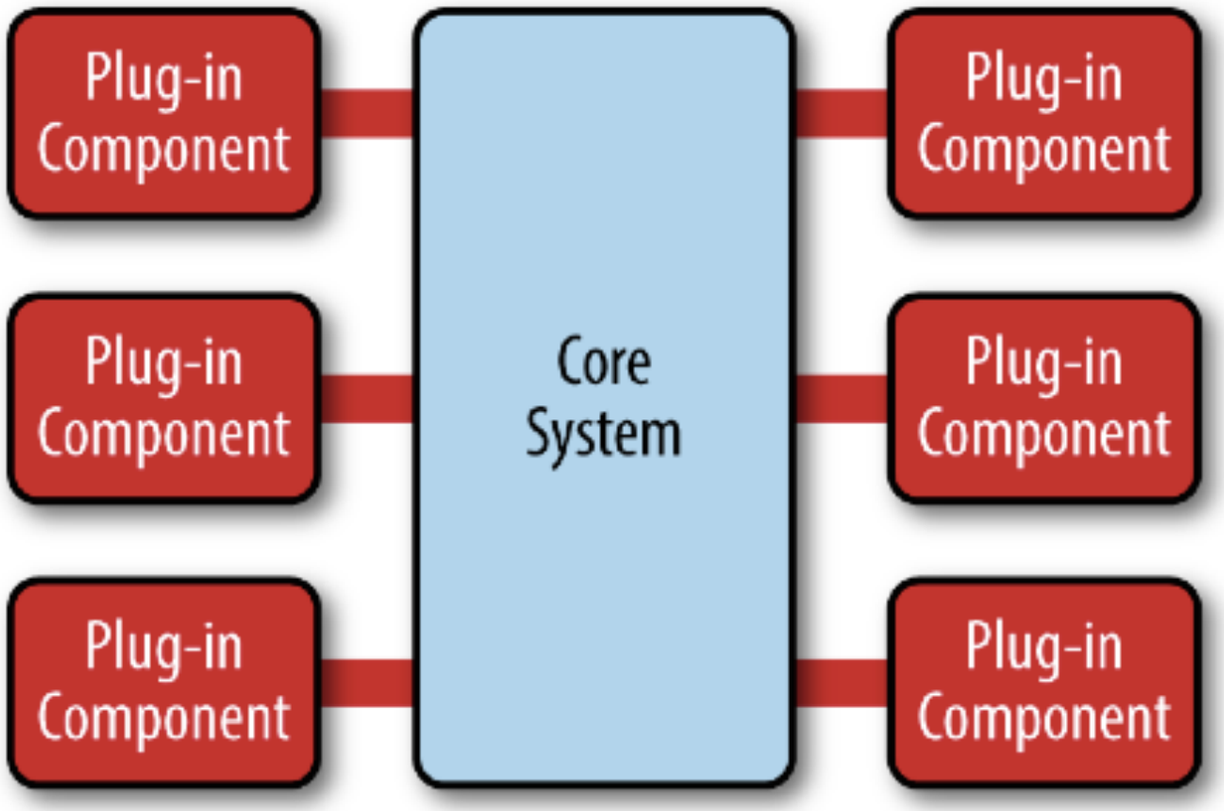
Data4

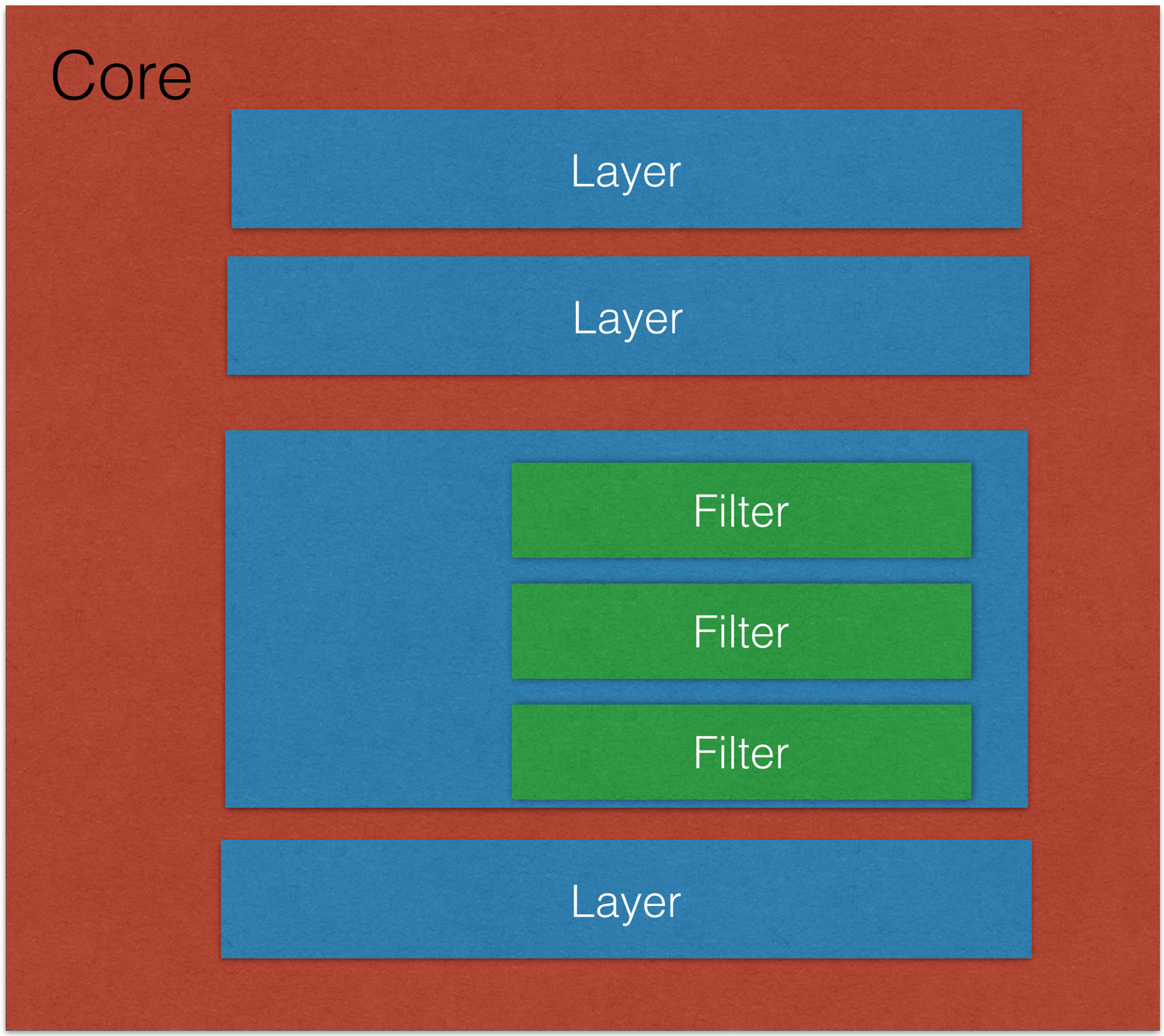
Filter 4

f4(int)

Data5

Microkernel Architecture

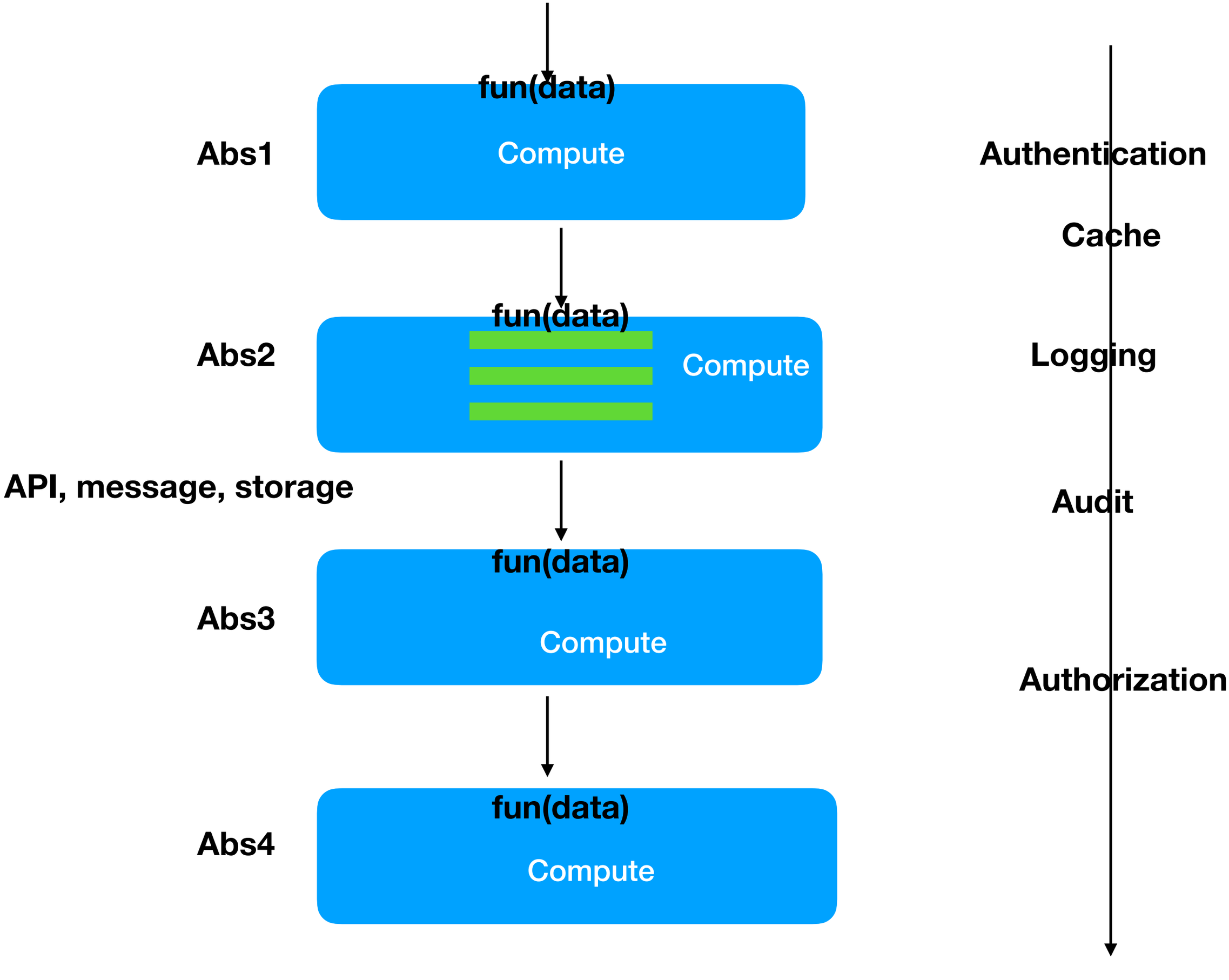


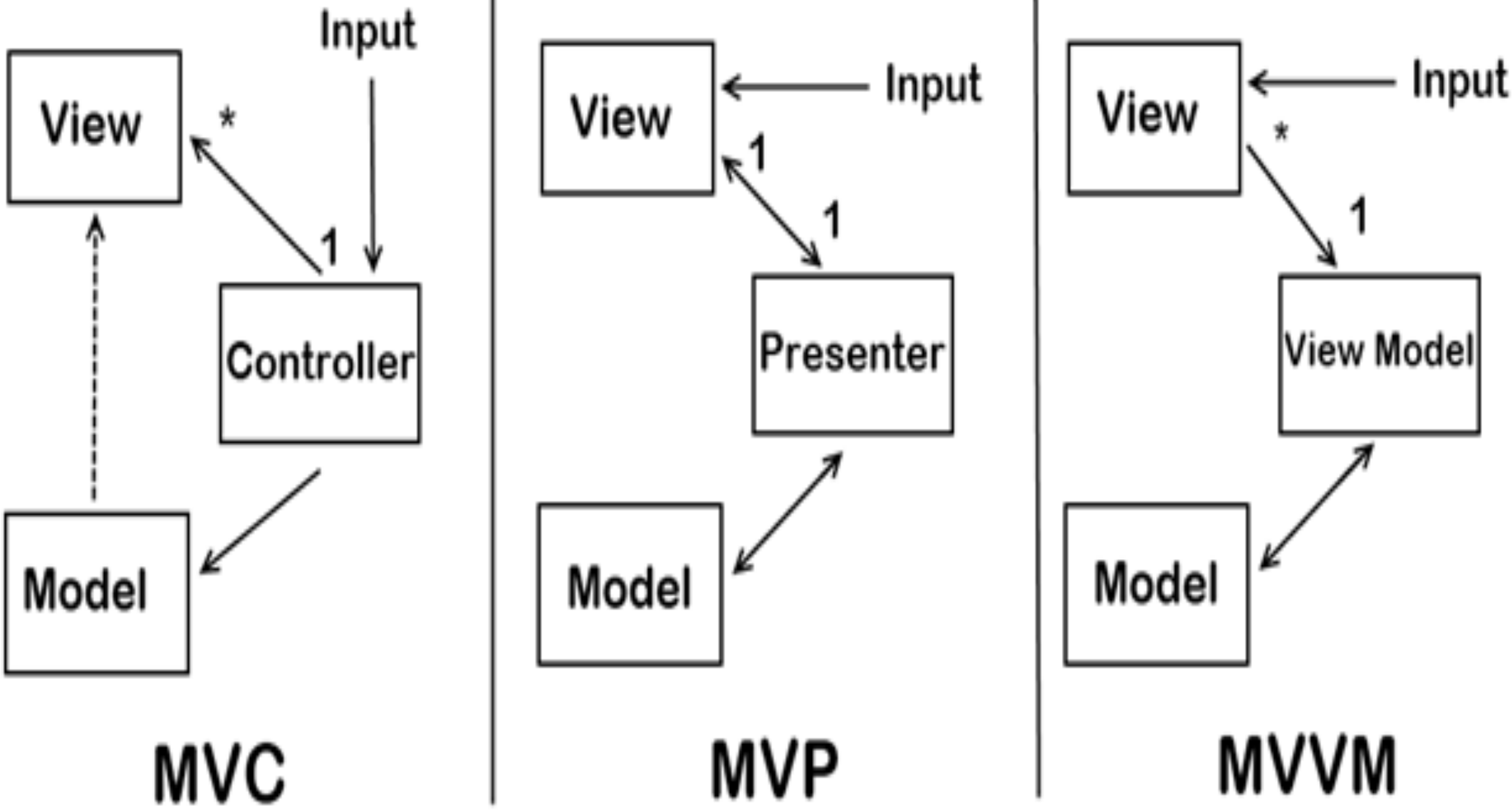


Plugins

Plugins

Plugins


fun(data)
Abs1
Compute
fun(data)
Abs2
Compute
API, message, storage
fun(data)
Abs3
Compute
fun(data)
Abs4
Compute
Authentication
Cache
Logging
Audit
Authorization


View
Input
*
1
Controller
Model
MVC
View
Input
1
1
Presenter
Model
MVP
View
Input
*
1
View Model
Model
MVVM

1

Ship one binary

Application
Exe

Source file

Source file

Compile time
monolithic

2

Ship one binary

Application
Exe

Lib binaries

Lib binaries

Source file

Link time
monolithic

3

Ship multiple binary

Process

Application
Exe

Dll/so/jar

Dll/so/jar

Dll/so/jar

Runtime
monolithic

In process
comp

Ship multiple binary & host in multiple process

4

Process

Exe

Dll/so/jar

Distributed
comp

Process

Exe

Dll/so/jar

Distributed
comp

Distributed Application

5

Process

Exe

Dll/so/jar

Application1

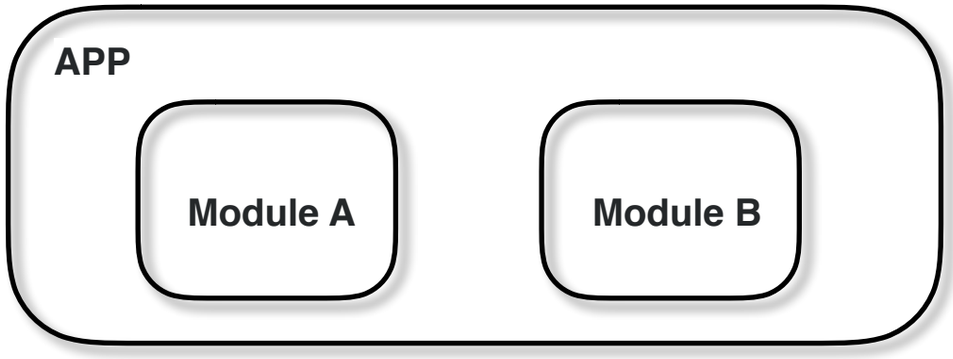
Process

Exe

Dll/so/jar

Application2

Micro Application



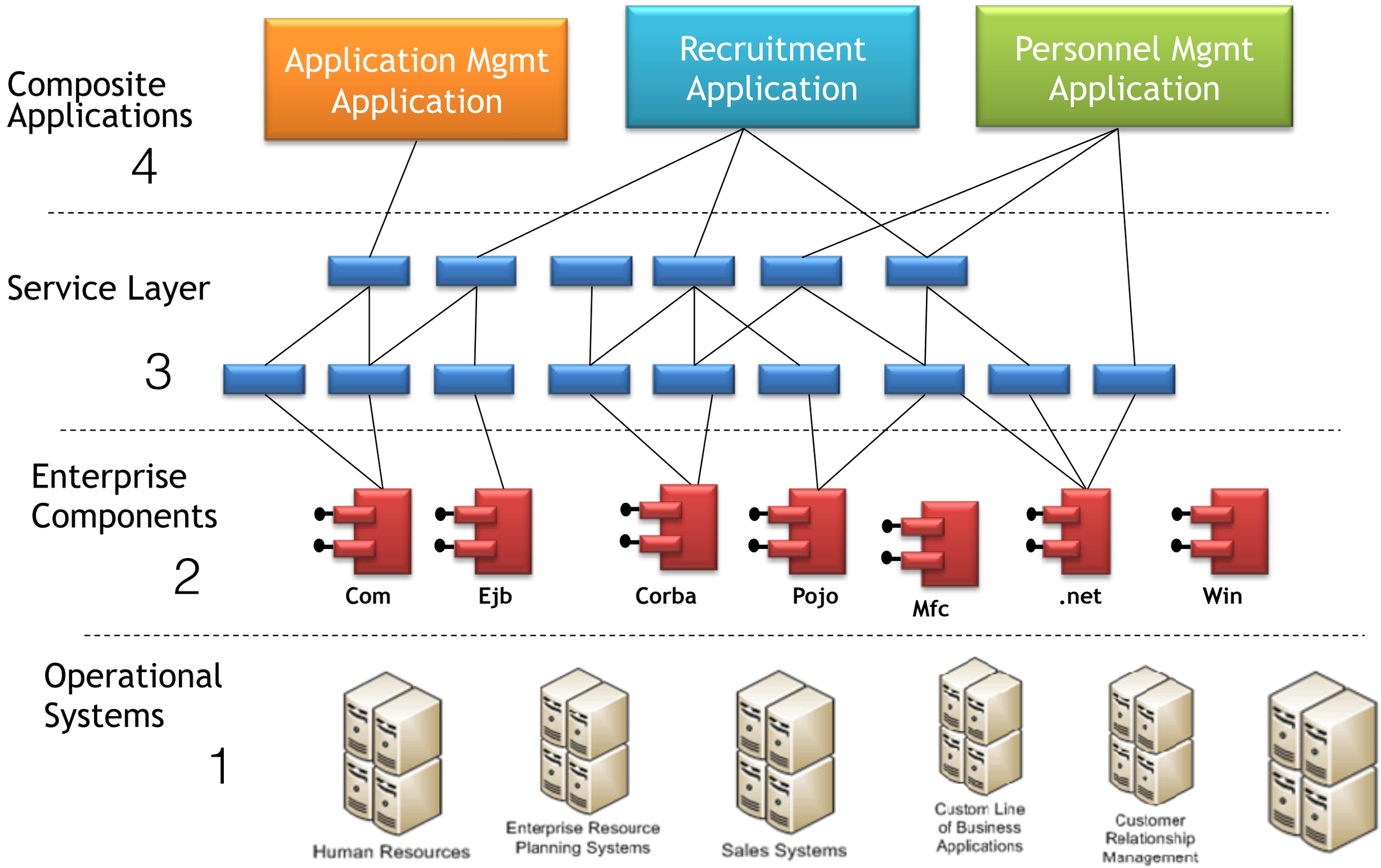
Can share persistence

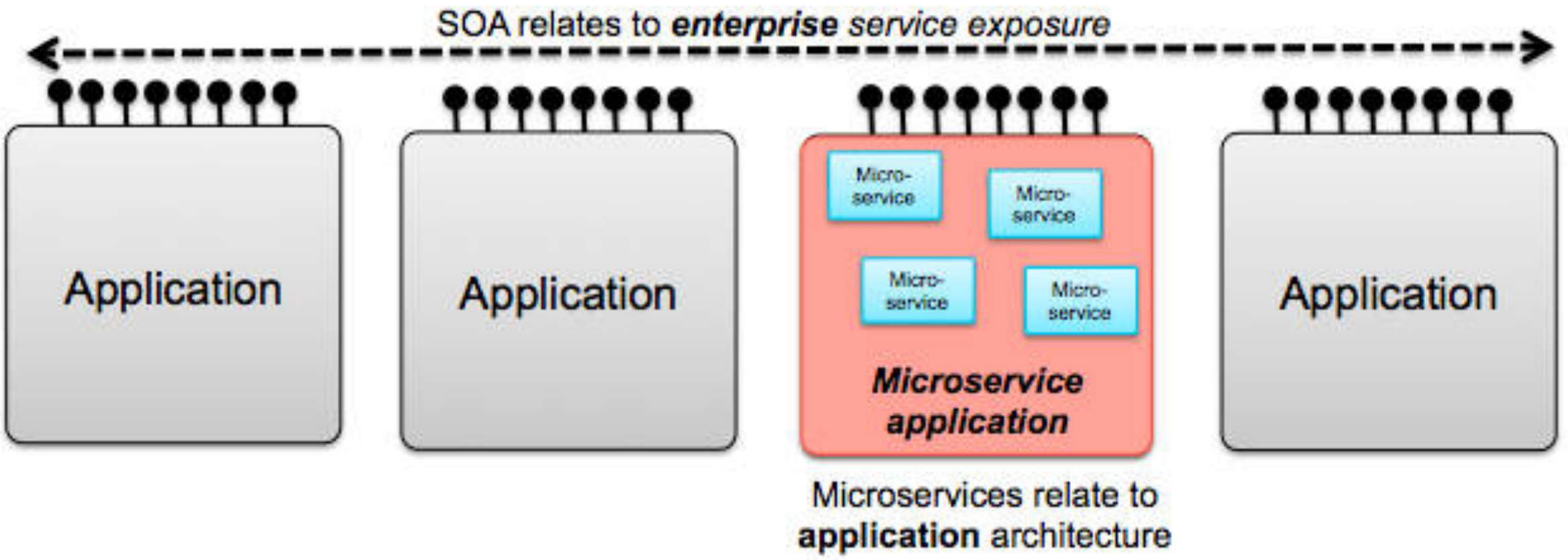
Should not share persistence

Can share Infra

Should not share Infra

Service Oriented Architecture

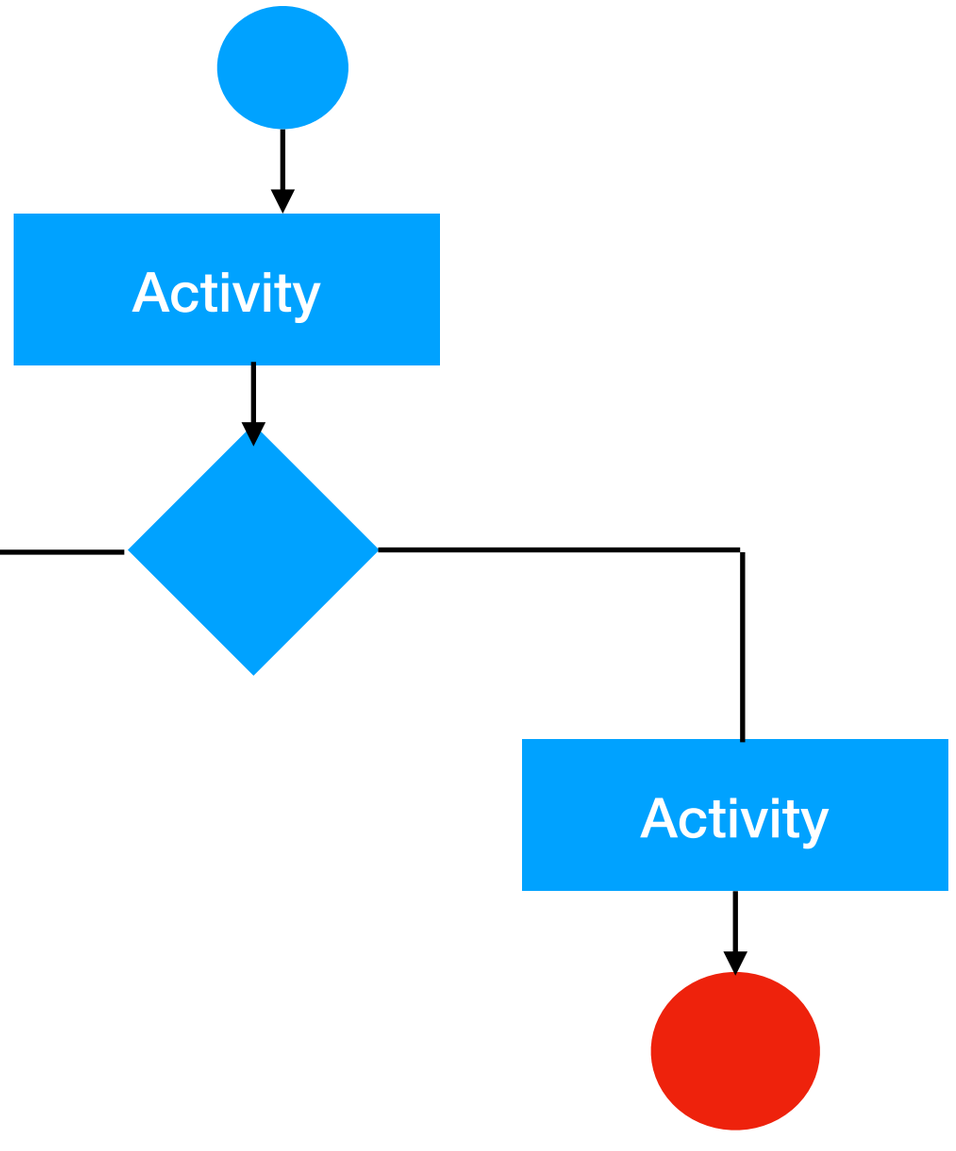


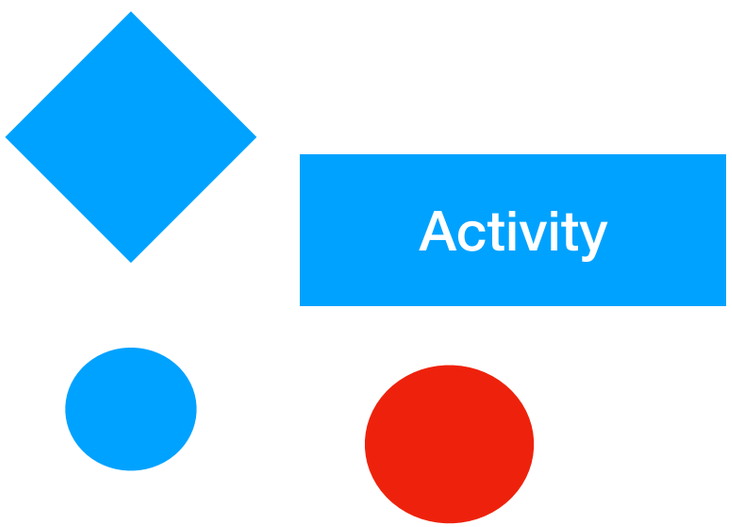


Microservices relate to **application** architecture

An application is required to perform a variety of tasks of varying complexity on the information that it processes. The processing tasks performed by each module, or the deployment requirements for each task, could change as business requirements are updated. Also, additional processing might be required in the future, or the order in which the tasks performed by the processing could change. A solution is required that addresses these issues, and increases the possibilities for code reuse.

A workflow implementation. The implementation of a workflow contains concepts like the order of the different steps, evaluating the results of steps, deciding what the next step is, etc.





A task scheduler. A scheduler contains all the logic for scheduling and triggering tasks

Internet browses plug-ins add additional capabilities that are not otherwise found in the basic browser

Networking engineering is a complicated task, which involves software, firmware, chip level engineering, hardware, and electric pulses. To ease network engineering, the whole networking concept is decomposed into more manageable parts.

Watch Dog

FluentD

Image pre processing
(pipes & filter)

Image post processing
(pipes & filter)

Image Matching
Algorithm

Image Matching
Ensembler

Control Flow

Network
Access Layer

Web Socket
Server
(receive image)

Rest
Client
(read config)

Web Socket
Client
(Send image)

Data Access Layer

Cache
Library
(config)

Log
Library

Exception
Handling
Library

Circuit
Breaker

Retry

Commons

Circuit Breaker

Log Library

Cache Library (config)

Exception Handling Library

Web Socket Server

Queue

Control Facade

Config Proxy

Rest APi Client

Cloud Storage Proxy

Queue

Cloud Storage Egress

Web Socket Client

Data Access Layer

Temp Local Storage

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithm

Domain

Network.Commons

Queue Receiver <<Adapter>>

Queue Publisher <<Adapter>>

RestClient <<Adapter>>

Circuit Breaker

CrossCutting.Commons

Log Library

Cache Library (config)

Exception Handling Library

Data Collection Service

<<write>>

Distributed Queue

Defect Detection Service

<<read>>

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Config Service

Distributed Queue

<<read>>

Materialized View Service

<<CRUD>>

<<read>>

UI Portal

Control Center

User App

Field Gateway

web socket

http://

GRPC

MQTT

OAuth Server

API Gateway

spring boot

Event Bus

Cloud Gateway (IOT Hub)

Query Micro Service

Domain Micro Service

logstash

elastic

kibana

Big Data Store

Data Store

Cache server

Operations

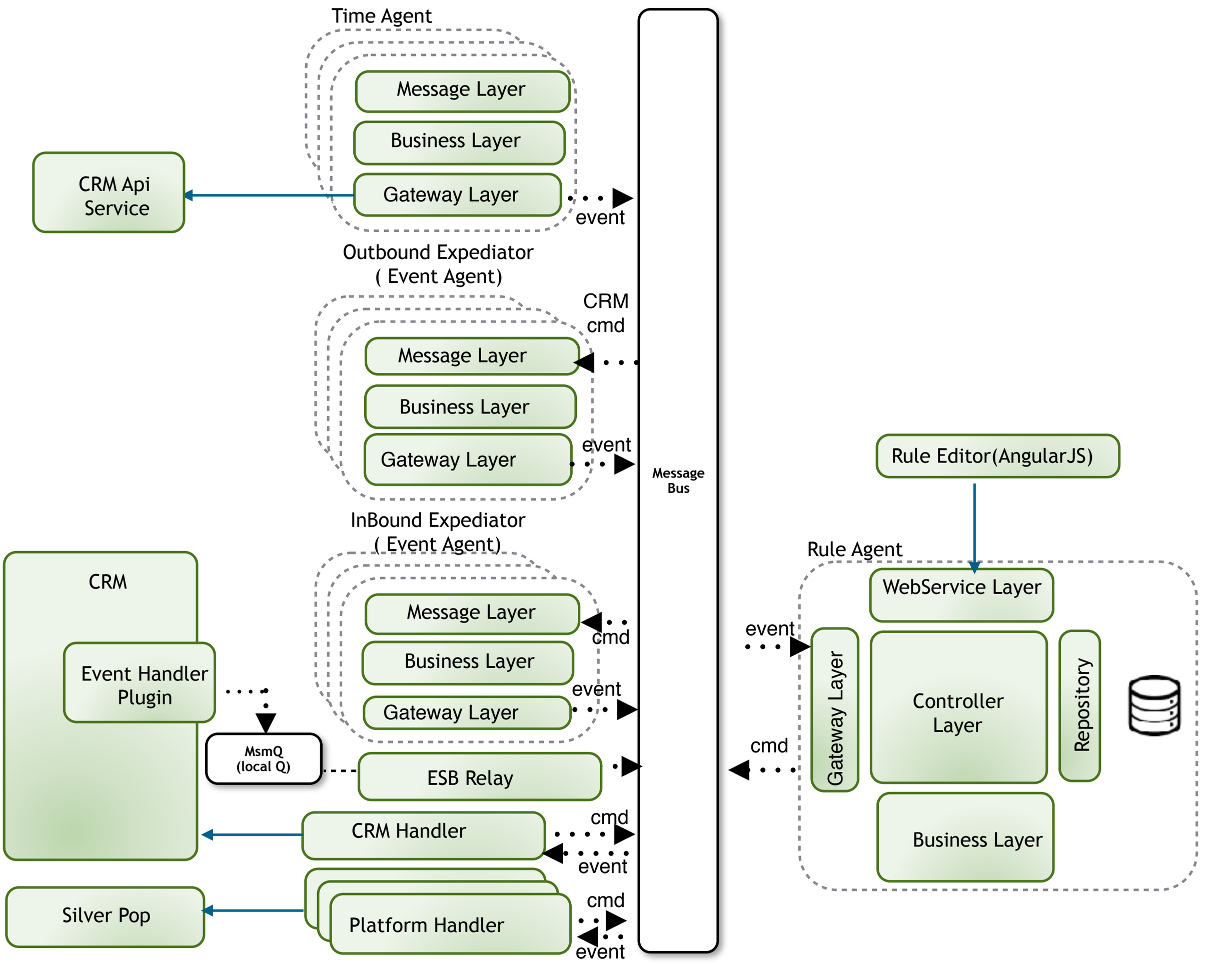
Configuration mgmt

Secret Key Mgmt

docker

kubernetes

Machine Learning


Time Agent
Message Layer
Business Layer
Gateway Layer
CRM Api Service
event
Outbound Expediator
(Event Agent)
CRM
cmd
Message Layer
Business Layer
Gateway Layer
event
Message Bus
InBound Expediator
(Event Agent)
CRM
Message Layer
Business Layer
Gateway Layer
cmd
event
Event Handler Plugin
MsmQ (local Q)
ESB Relay
CRM Handler
cmd
event
Silver Pop
Platform Handler
cmd
event
Rule Editor(AngularJS)
Rule Agent
WebService Layer
event
cmd
Gateway Layer
Controller Layer
Repository
Business Layer

Inbound expediatior ?

Unified Agent Desktop

rest , web socket

API Gateway

grpc

CRM Adapter Microservice

CRUD

Api Layer

Event Handler Plugin

Microsoft Dynamics

CRM Msg Q

Priority Local Msg Q

High

Relay

Low

Relay

outbound Expediator Microservice

query

enriched msg

CRM Action Handler Microservice

CRUD

3rd Party Lib

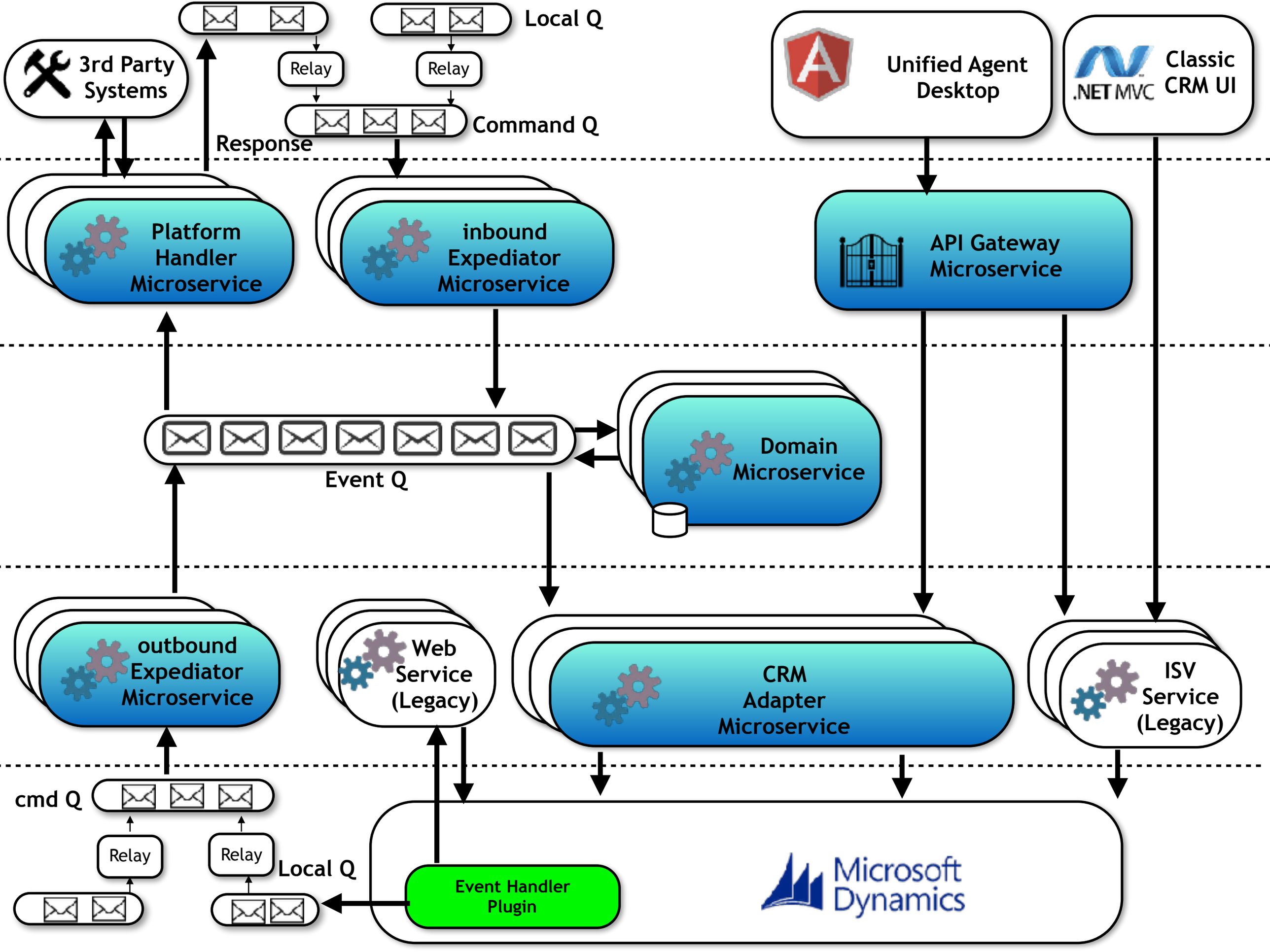
Platform Handler Microservice

Domain Microservice

ISV Service (Legacy)

Classic CRM UI

Event Bus


Local Q
Relay
Relay
3rd Party Systems
Command Q
Response
Unified Agent Desktop
Classic CRM UI
.NET MVC
Platform Handler Microservice
inbound Expediator Microservice
API Gateway Microservice
Event Q
Domain Microservice
outbound Expediator Microservice
Web Service (Legacy)
CRM Adapter Microservice
ISV Service (Legacy)
cmd Q
Relay
Relay
Local Q
Event Handler Plugin
Microsoft Dynamics

Customer

Web Scraping

raw data

Extract
Normalize
Categorize
Disambiguate
Merge
Index
Update

features

Chat Bot

NLU Layer

Report

plain text array

NLG Layer

Decision Engine

Data Lake

Training/Test data

Recommendations

Deep Learning

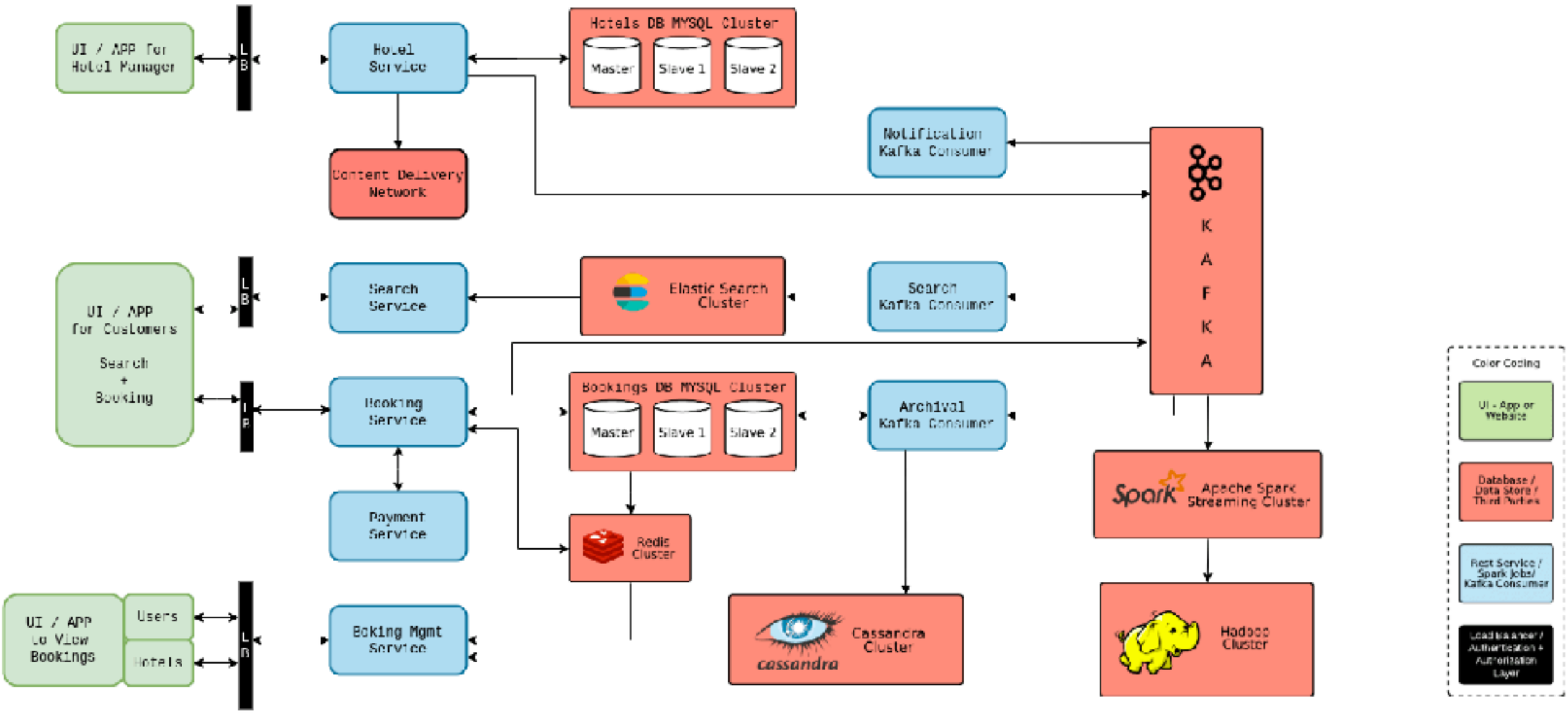
creates

Recommendation Model

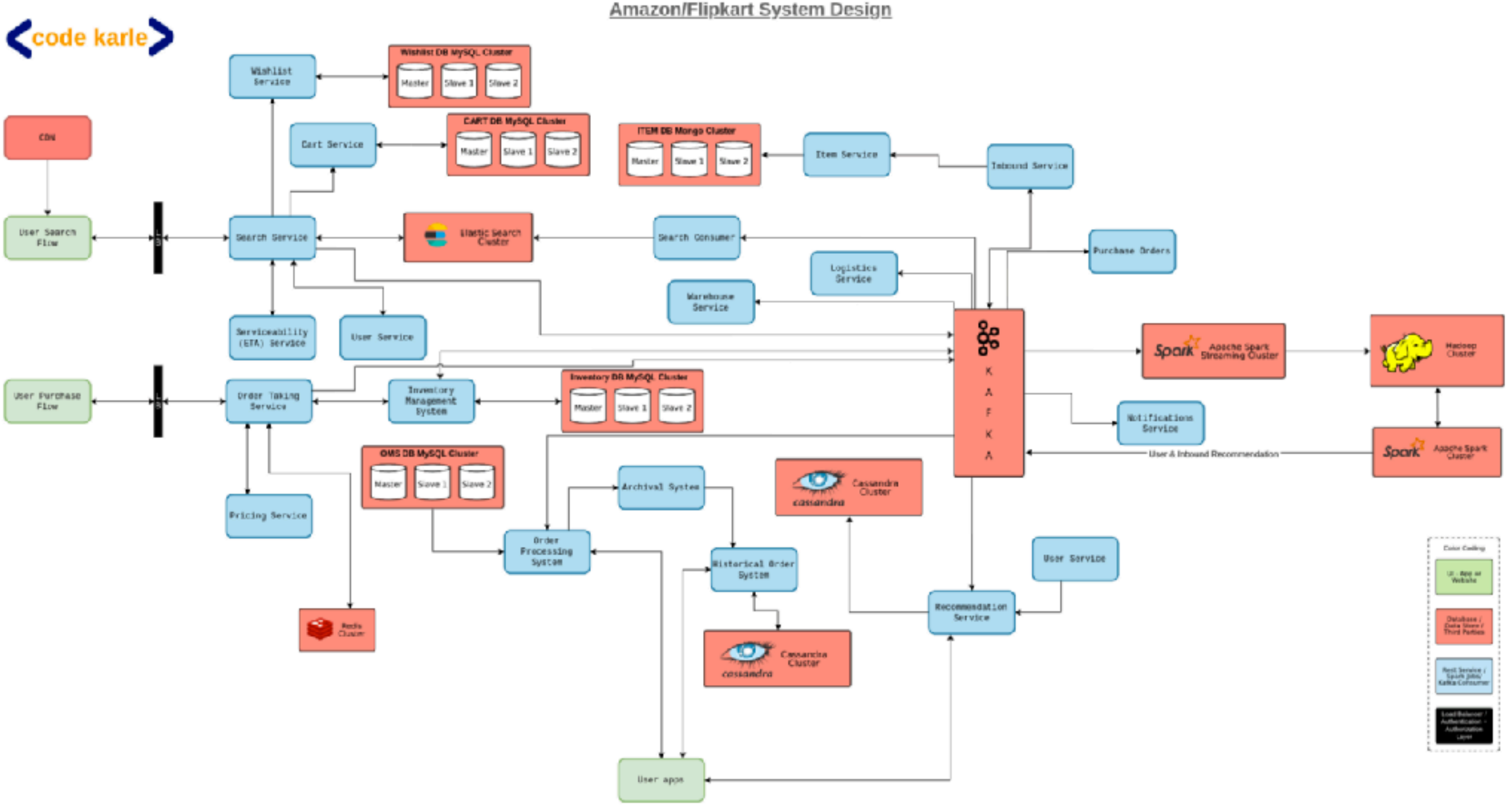
uses

Data Mining

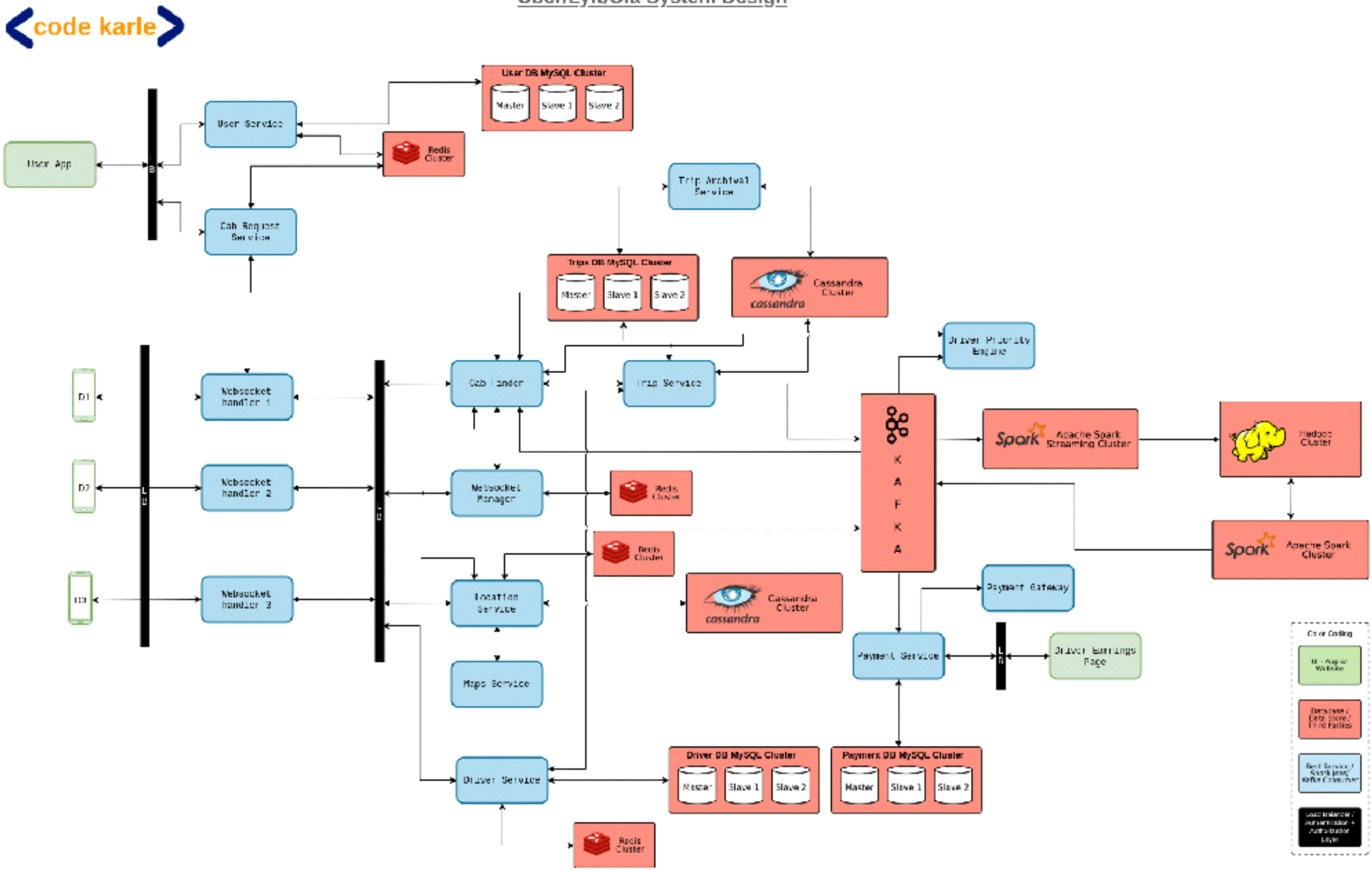
Airbnb/Booking.com System Design



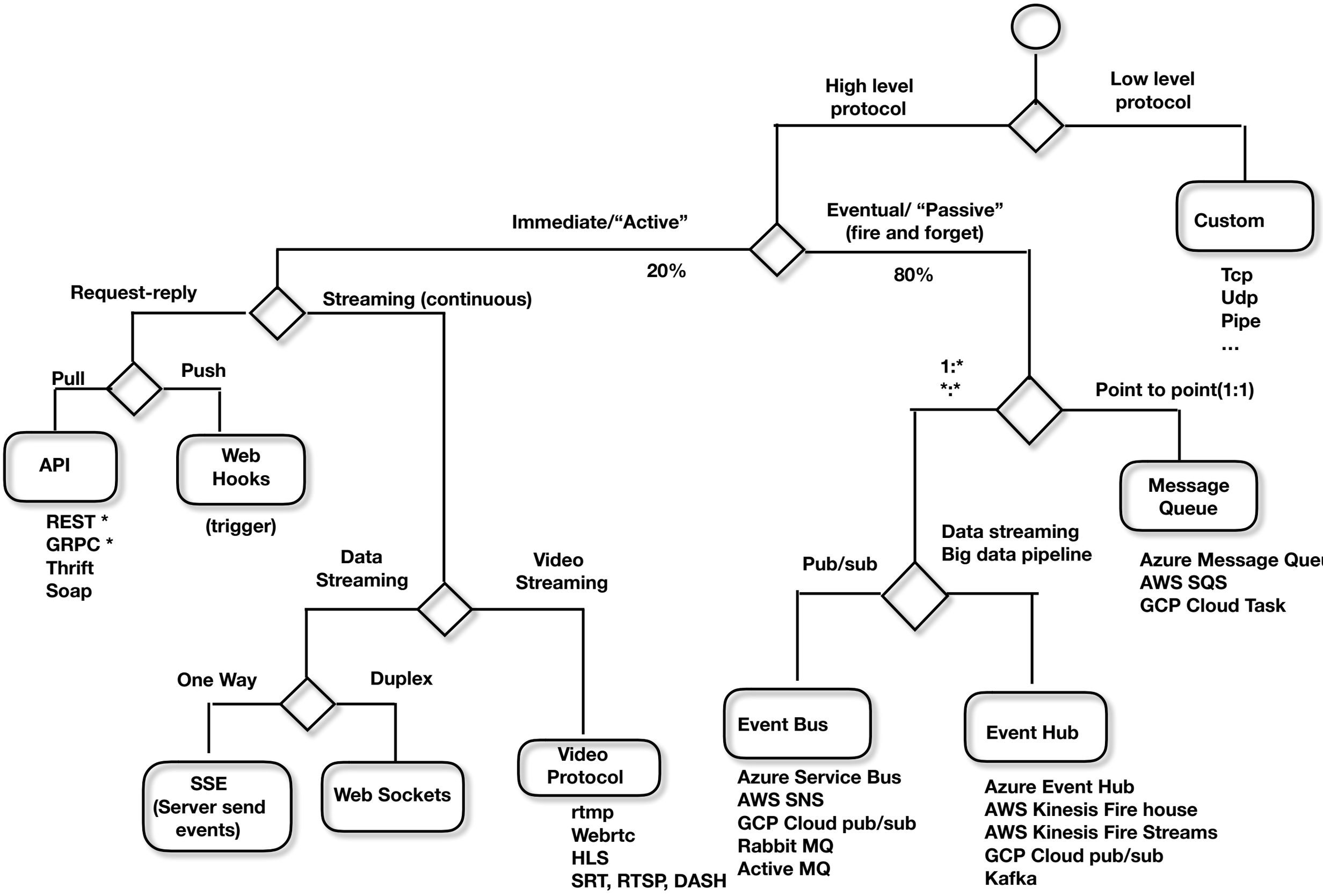
Amazon/Flipkart System Design



Uber/Lyft/Ola System Design



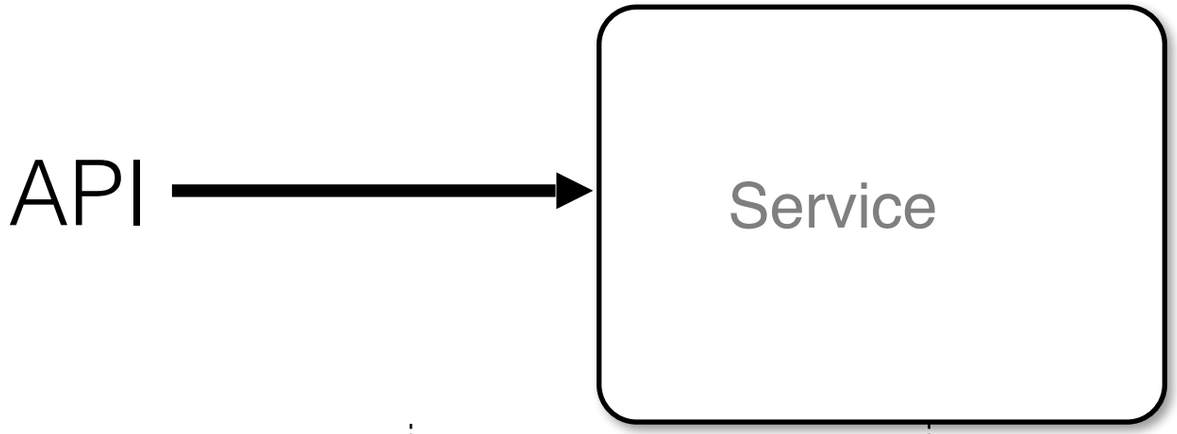
Choose Communication protocol



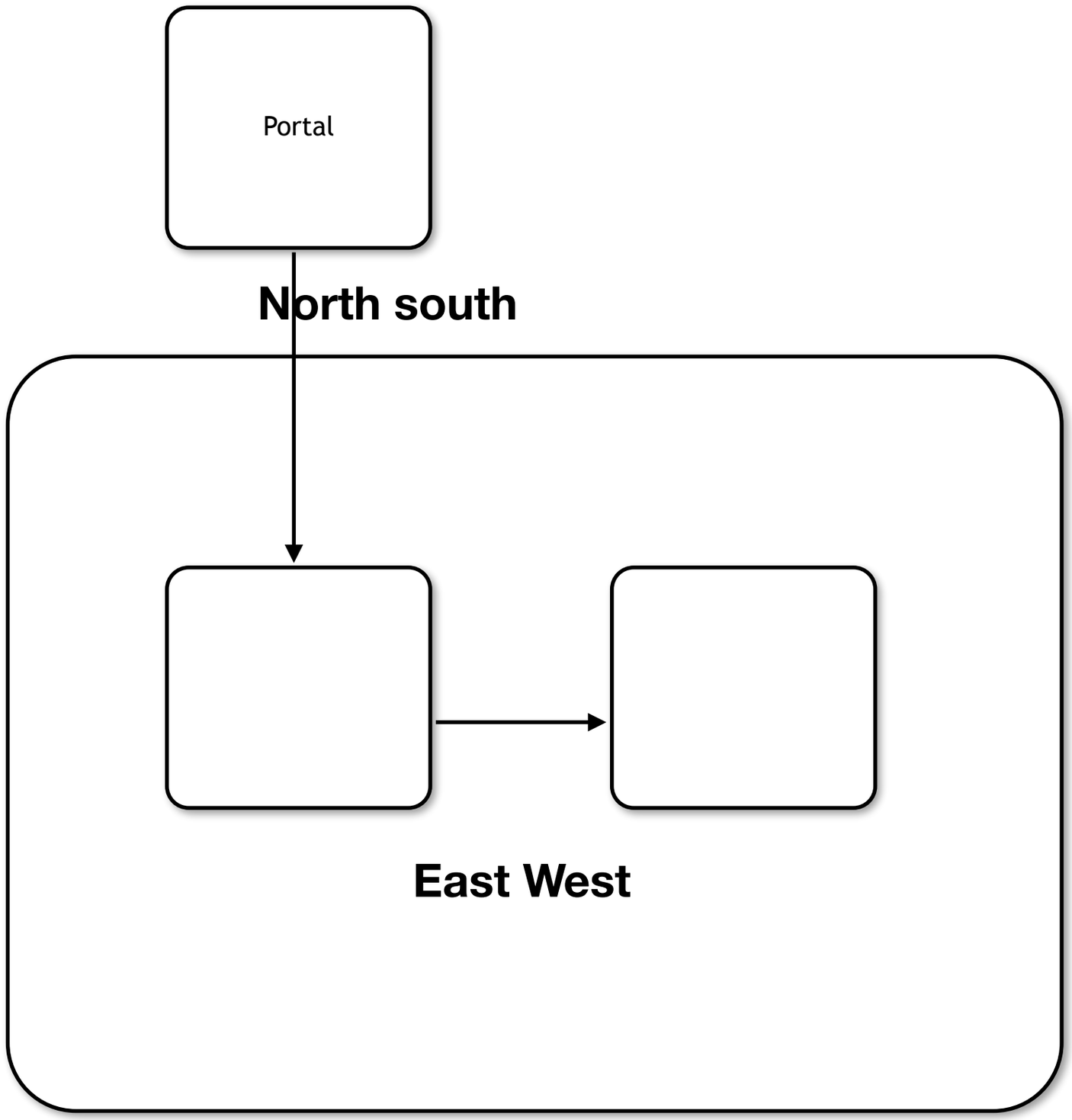
Connected protocol
(REST, WS, GRPC , Thrift)

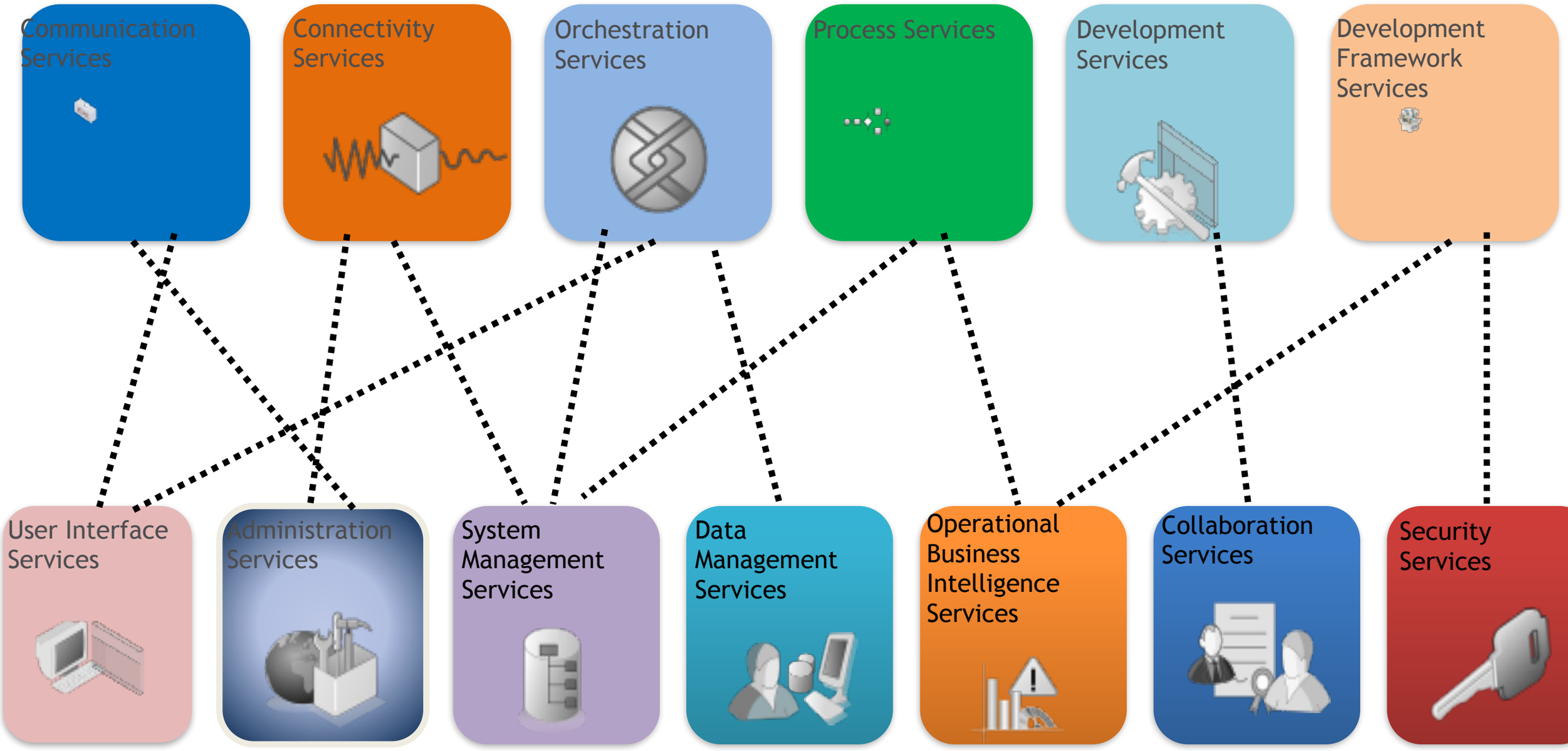
Message protocol
(AMQP, MQTT, …)

	Connected	Msg database (ACID)
Scale	- -	+ +
Coupling	- -	+ +
Reliability	- -	+ +
Developer effort	++	- -
Consistency	Immediate	Eventual
Debugging		
Delivery Order	Ordered ++	Unordered * - -
Duplicate	- -	- -
Dev Ops Effort		
Exception Handling	+ +	- - ?



	REST	WSS	GRPC
Browser Support	**Y**	**Y**	**N**
Format	**TEXT (HTTP 1.x)**	**TEXT (HTTP 1.x)**	**BINARY, HTTP2**
Serialization	**JSON**	**JSON / XML / Any**	**Proto Buf**
Performance	**- -**	**+**	**+ +**
Duplex communication	**Pull**	**Pull & Push**	**Pull**
Use case	**North South**	**Streaming & North South**	**East West**


Portal
North south
East West


Communication Services
Connectivity Services
Orchestration Services
Process Services
Development Services
Development Framework Services
User Interface Services
Administration Services
System Management Services
Data Management Services
Operational Business Intelligence Services
Collaboration Services
Security Services

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

User Interface Services

Administration Services

System Management Services

Data Management Services

Operational Business Intelligence Services

Collaboration Services

Security Services

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

User Interface Services

Administration Services

System Management Services

Data Management Services

Operational Business Intelligence Services

Collaboration Services

Security Services

Event vs Command

Post **Pre**

OrderCreated **InvoiceCreated** **OrderCanceled**

Communication Services

Connectivity Services

Orchestration Services

Process Services

Development Services

Development Framework Services

CreateOrder

OderCreated **OderCanceled** **OderCreated**

User Interface Services

Administration Services

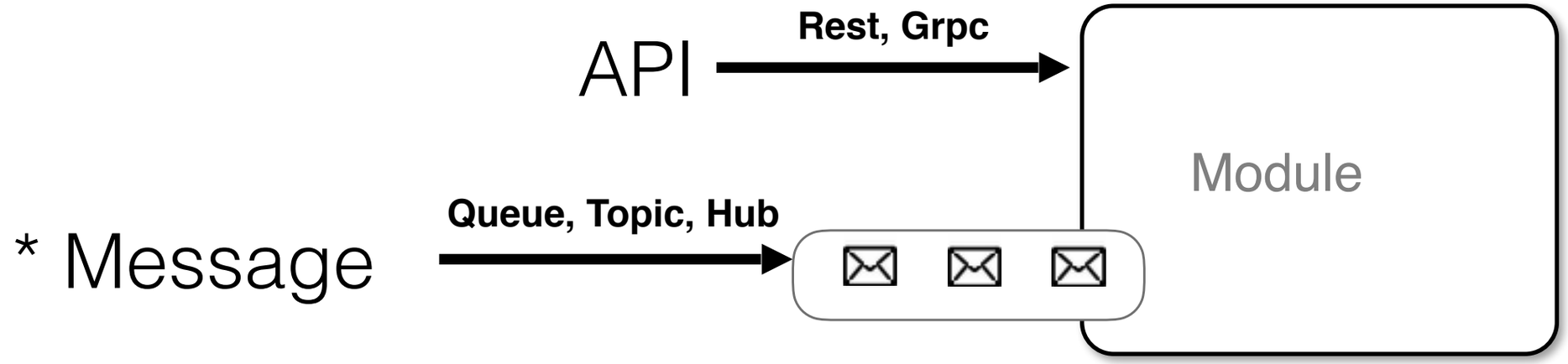
System Management Services

Data Management Services

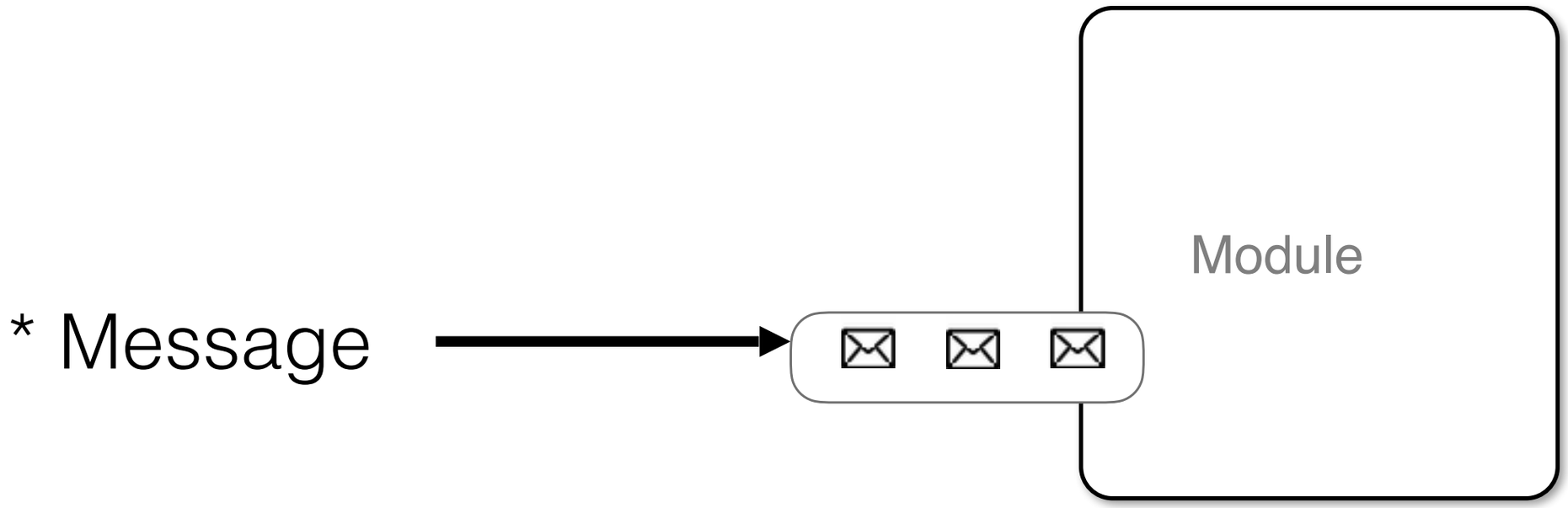
Operational Business Intelligence Services

Collaboration Services

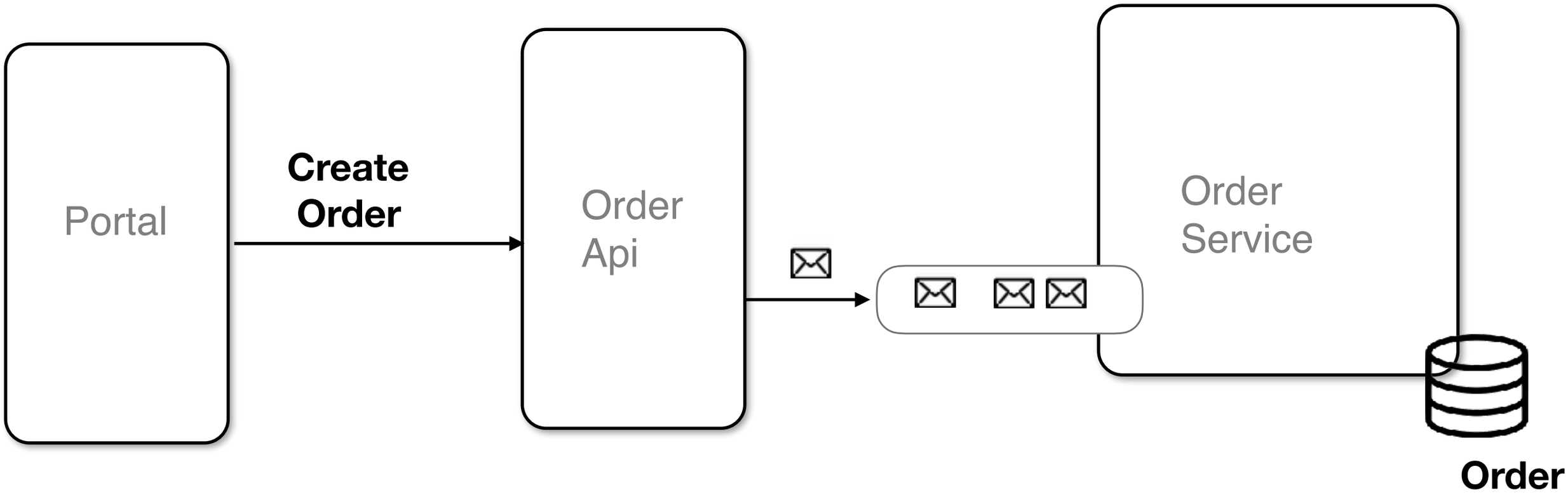
Security Services



	<< API >>	<< Message >>	
Ordering	Ordered (+)	Unordered(-)	
Duplicate	Yes (-)	Yes(-)	Idempotency
Protocol	2 way (+)	One way (-)	
Resilence (recover)	No (-) retry logic	Yes (+)	
Connection	Always connected	Occousionaly connected	
Scalability	- -	+ + +	
Consistency	Immediate (++)	Eventual (- -)	
Load Leveling	- -	+ +	
Low Coupling (maintainability)	- -	+ +	
Distributed Comm Patterns	- -	+ +	SAGA, Materialized View
Internet	Yes	Yes *	
Browser support	Yes	No	
Dev effort	++	- -	
Operational effort	+ +	- -	




```
fun(){
    while(true){
        Msg m = GetMessage();
        .....
        m.ack();
    }
}
```

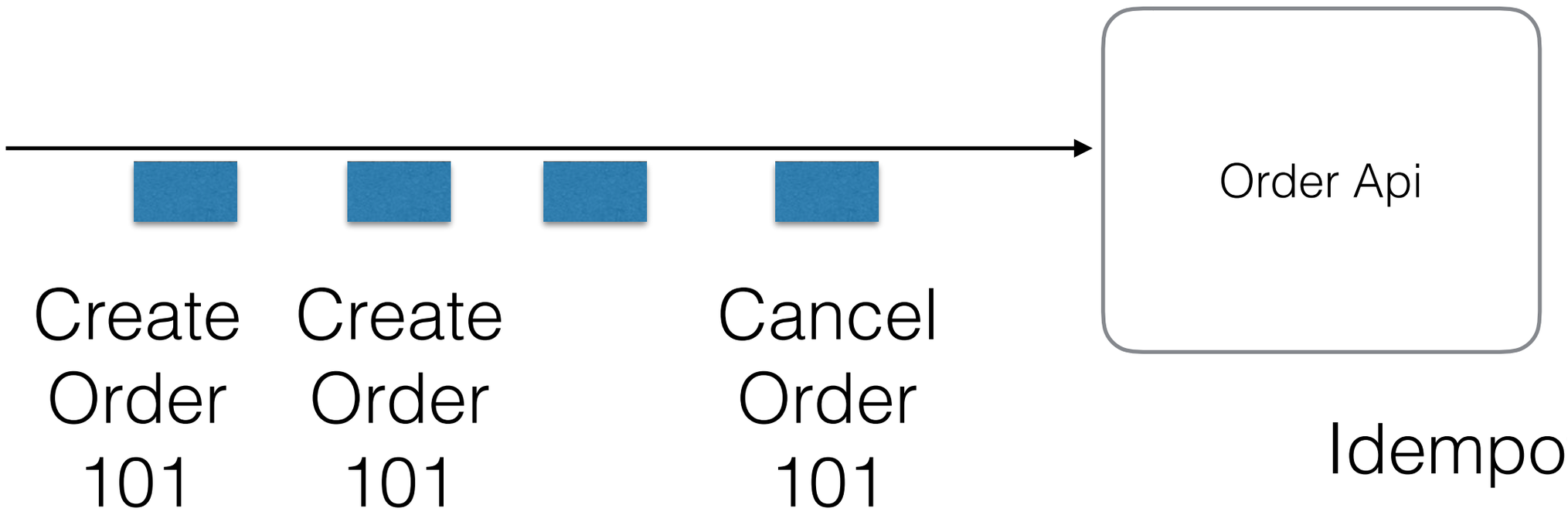

Portal
Create Order
Order Api
Order Service
Order
Order Service
Account Service
Account

Client

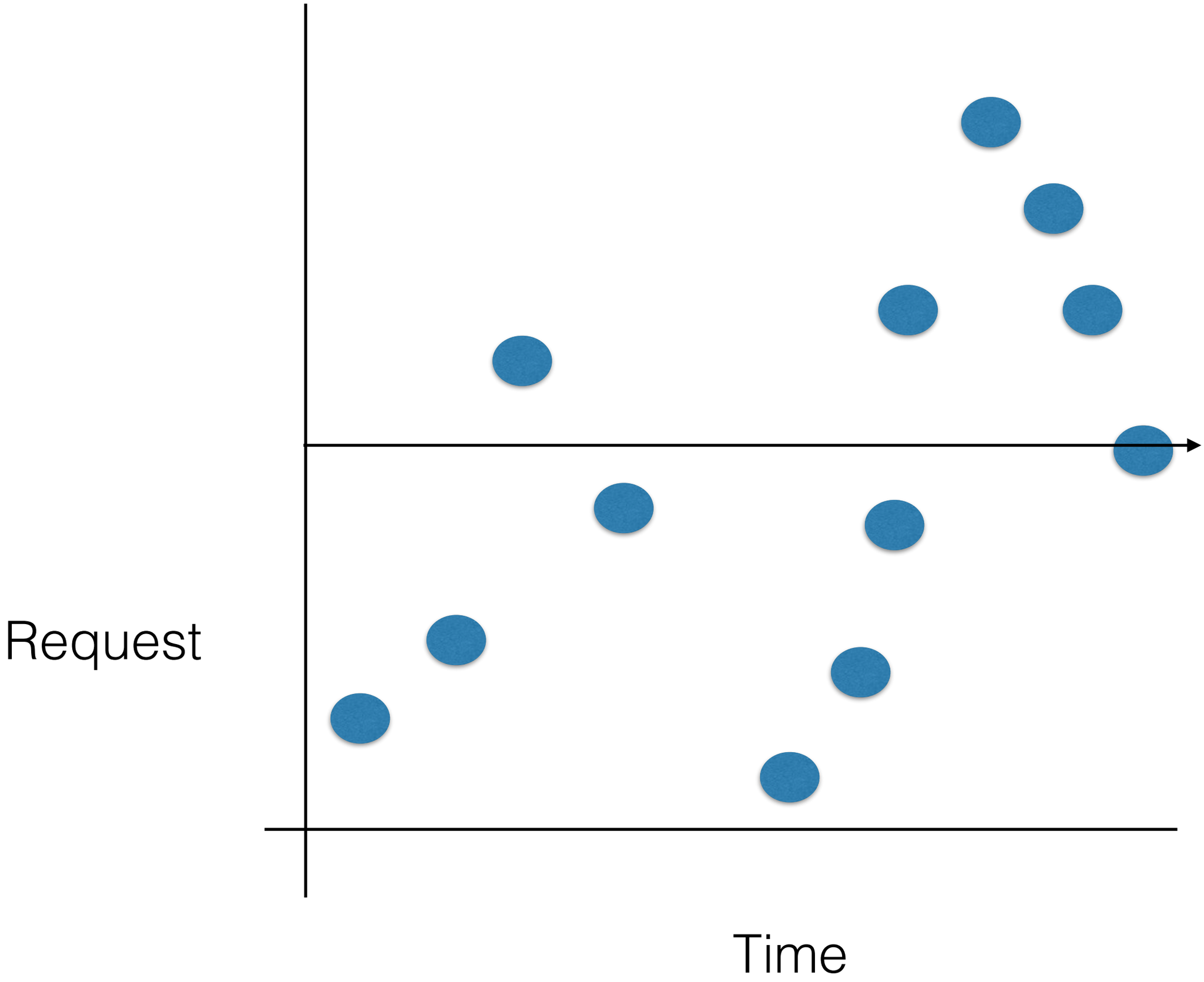
Server

Appp

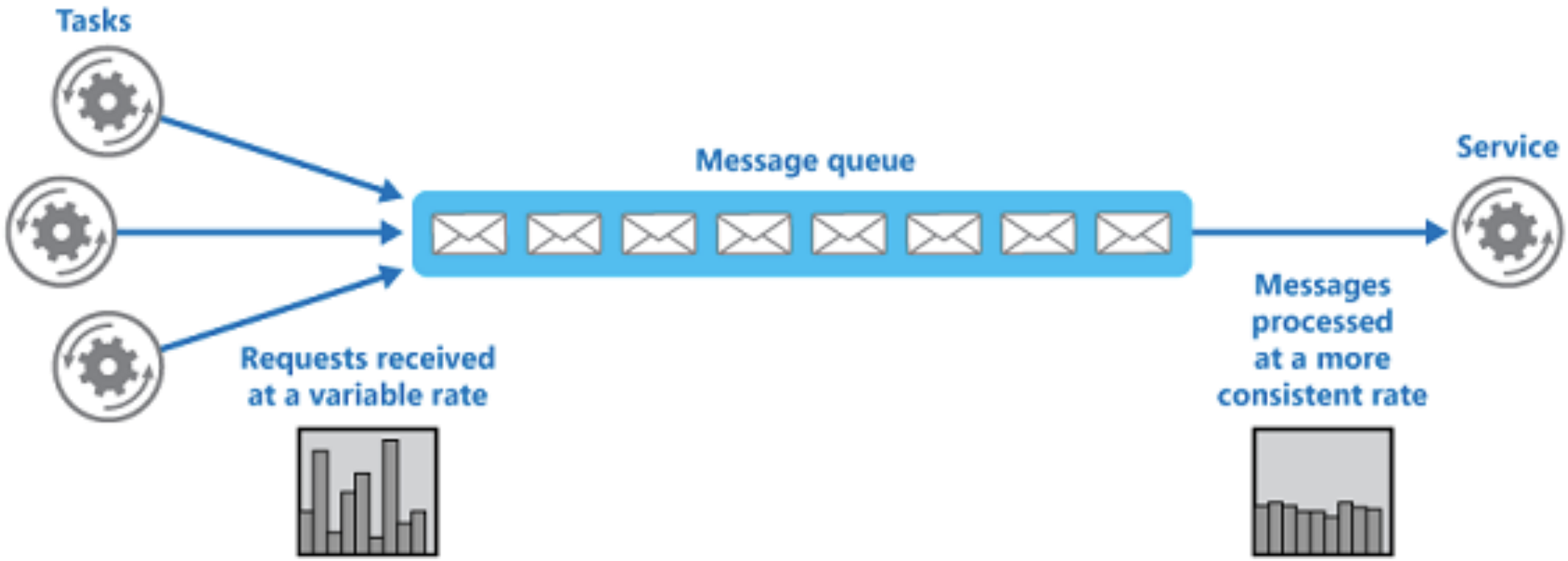
Service



Idempotency


Request
Time

Queue-based Load Levelling



source:msdn

p2p
SQS
MSMQ
...

Load leveling

Publisher

Queue

Payroll Subscriber 1

Payroll Subscriber 2

pubSub
RabbitMQ
SNS
Redis
MQ Series

Publisher

Topic

Tax Subscriber 1

Payroll Subscriber 2

Payroll Subscriber 3

T3 T2 T1

Publisher

Kafka
Event Hub
Kinesis

Tax Subscriber

State P1 :t

Tax Subscriber

State p2: t

Payroll Subscriber

State P1:t

<<Topic>>

Image

Task

Parallelism

Logic 1(data) , Logic 2(data) , Logic 3(data)

OCR(data) , Feature(data) ,Masking(data)

<<queue>>

Event1
Event 2
Event 3

Data

Parallelism

Logic(data1) , Logic(data2) , Logic(data3)

Anonyimization (Event1) ,
Anonyimization(Event2) ,
Anonyimization(Event3)

	Kafka	Pulsar	RabbitMQ (Mirrored)
Peak Throughput (MB/s)	605 MB/s	305 MB/s	38 MB/s
p99 Latency (ms)	5 ms (200 MB/s load)	25 ms (200 MB/s load)	1 ms* (reduced 30 MB/s load)
Number of Programming Languages Supported	17	6	22
Stack Overflow Questions	21,233	134	11,430
Pub/sub	Yes	Yes	Yes
Message routing	Medium	Medium	High
Queueing	Low	Medium	High
Event Streaming	High	Medium	No
Mission-critical	High	Low	High
Slack Community Size	23,057	2,332	7,492
Meetups	486	1	1
Message replay, time travel	+++	+++	-
Exactly-once processing	+++	+	-
consumption	Pull	Push	Push
Permanent storage	Yes	Partial	No

	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	

Table 1

	Rabbitmq	Kafka	Redis
Scale	50K msg per second	1 million/ sec	1 million/ sec
Transient messages	Yes	No	Y
Persistent yes	Yes	Yes	Y
Protocol	AMQP		
One to one	Y	N	Y
One to many	Y	Y	Y
Advanced Message Queueing Protocol-based routing	Y	N	N
consumption pattern	Pull+Push	Pull	
Client Libraries	Py, java, php, .net, js,…	Py, java, php, .net, js,…	
Managed Service	yes, but not native in aws	Azure, aws, Confluent	Azure, aws
License	MPL	Apache	
Message Priority	Yes	No	
Monitoring	yes	yes	
Authentication	OAuth2, Std Auth	Kerberos, Oauth2, std Auth	
Message delay	microsecond	Millisecond	
Potential data loss	Yes	Yes	
Community and vendor support	Good	Good	Good
Building an event store system (used as a store)	No	Yes	No
Ordering	not guaranteed	Guaranteed	
Replay events	No	Yes	No
Transactions	No	Yes	No

API VS WEBHOOKS

API

An API stands for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

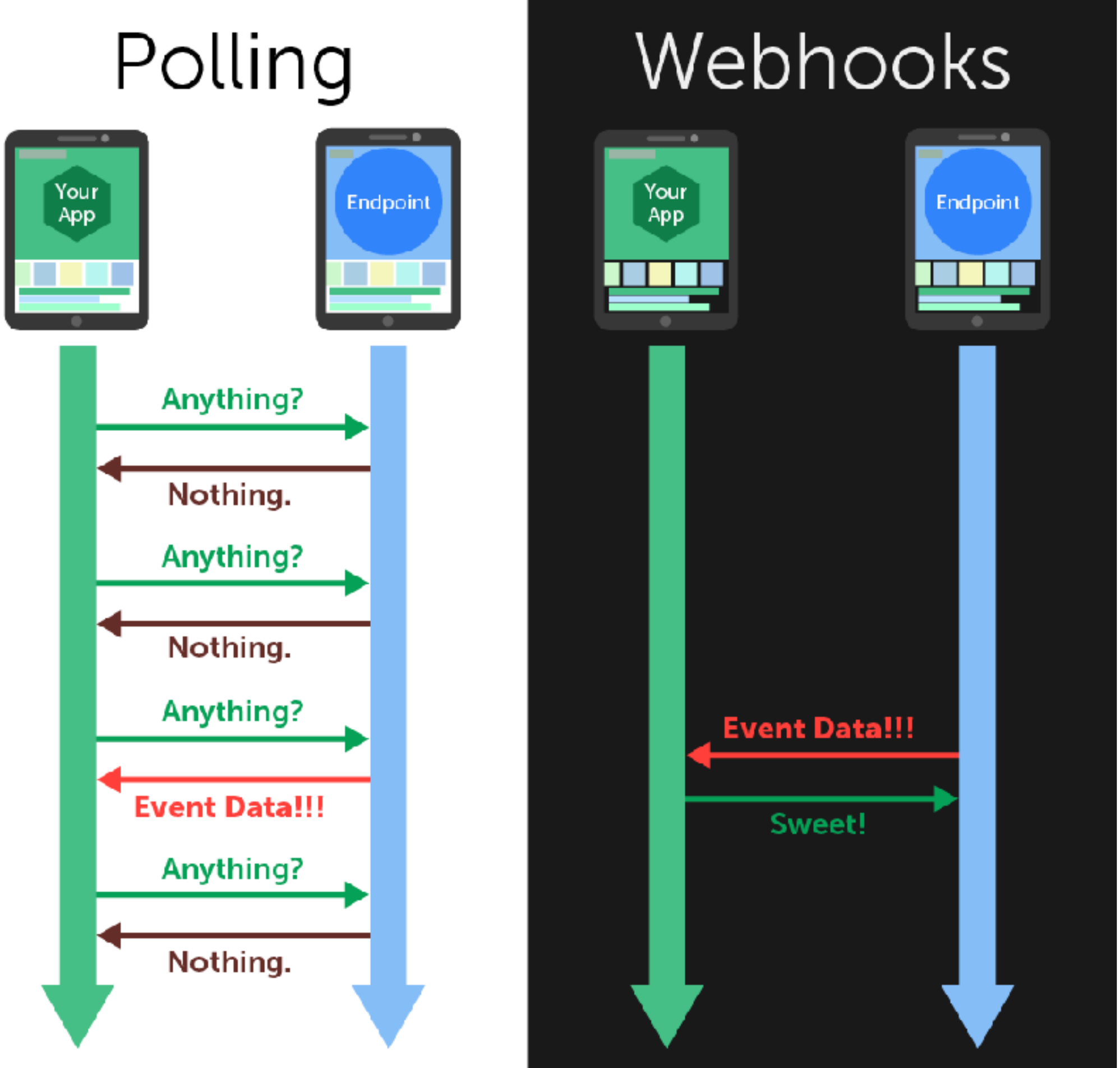
APIs are request-based, meaning that they operate when requests come from 3rd party apps.

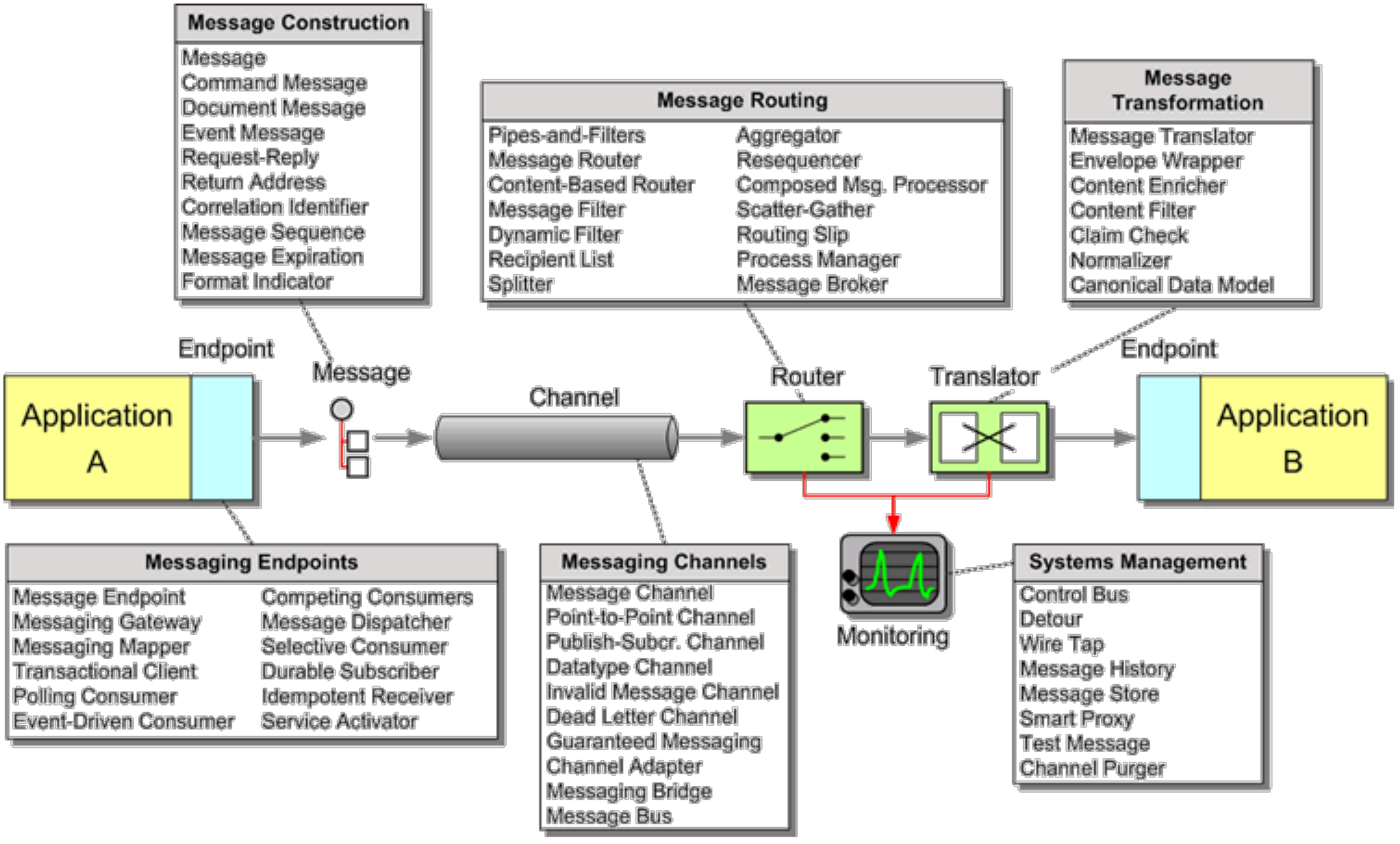
To use a real-world analogy, APIs would be likened to you repeatedly calling a retailer to ask if they've stocked up on a brand of shoes you like.

Webhook is also called reverse API, web callback, or an HTTP push API. It delivers data as an event happens or almost immediately.

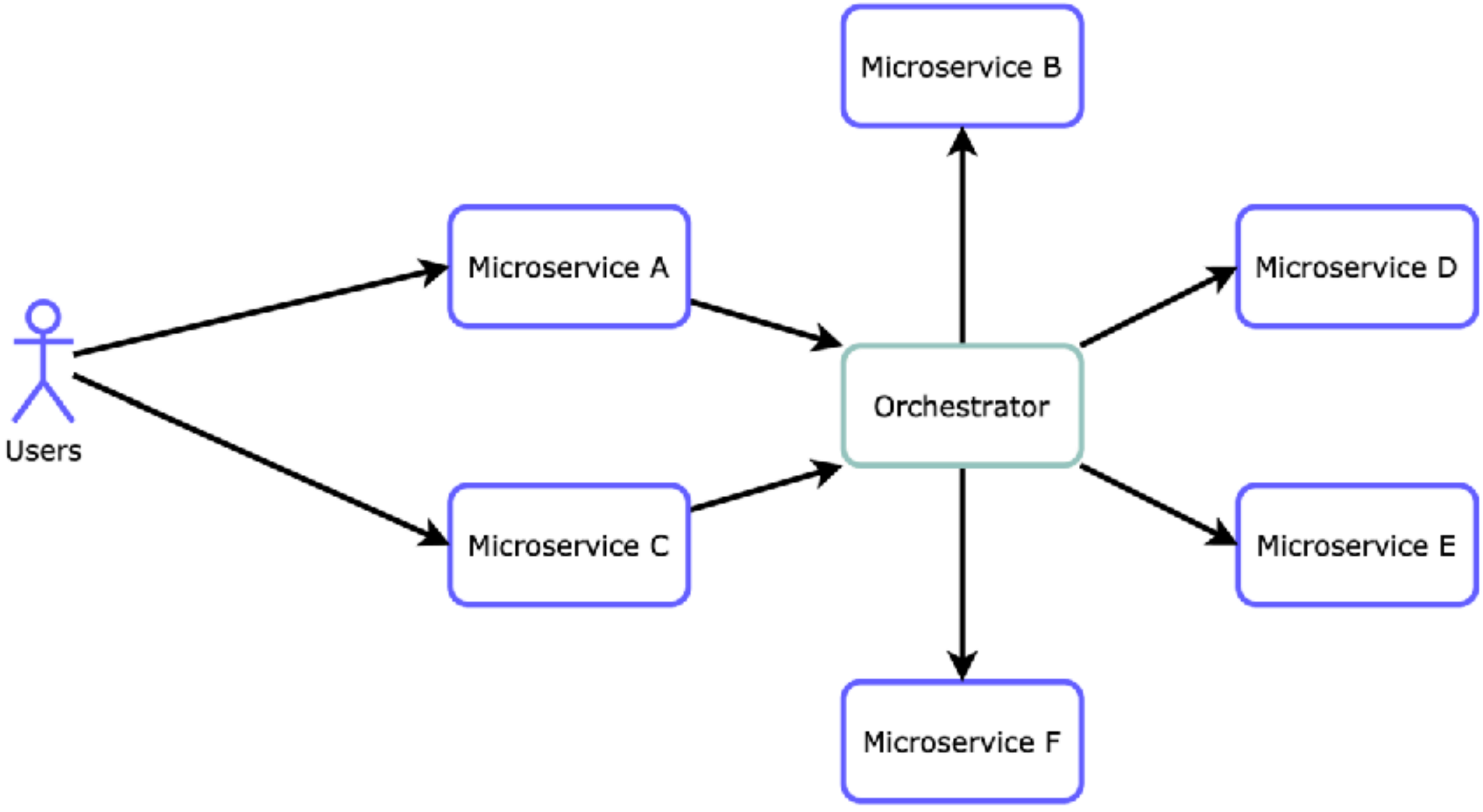
Webhooks are event-based, meaning that they will run when a specific event occurs in the source app.

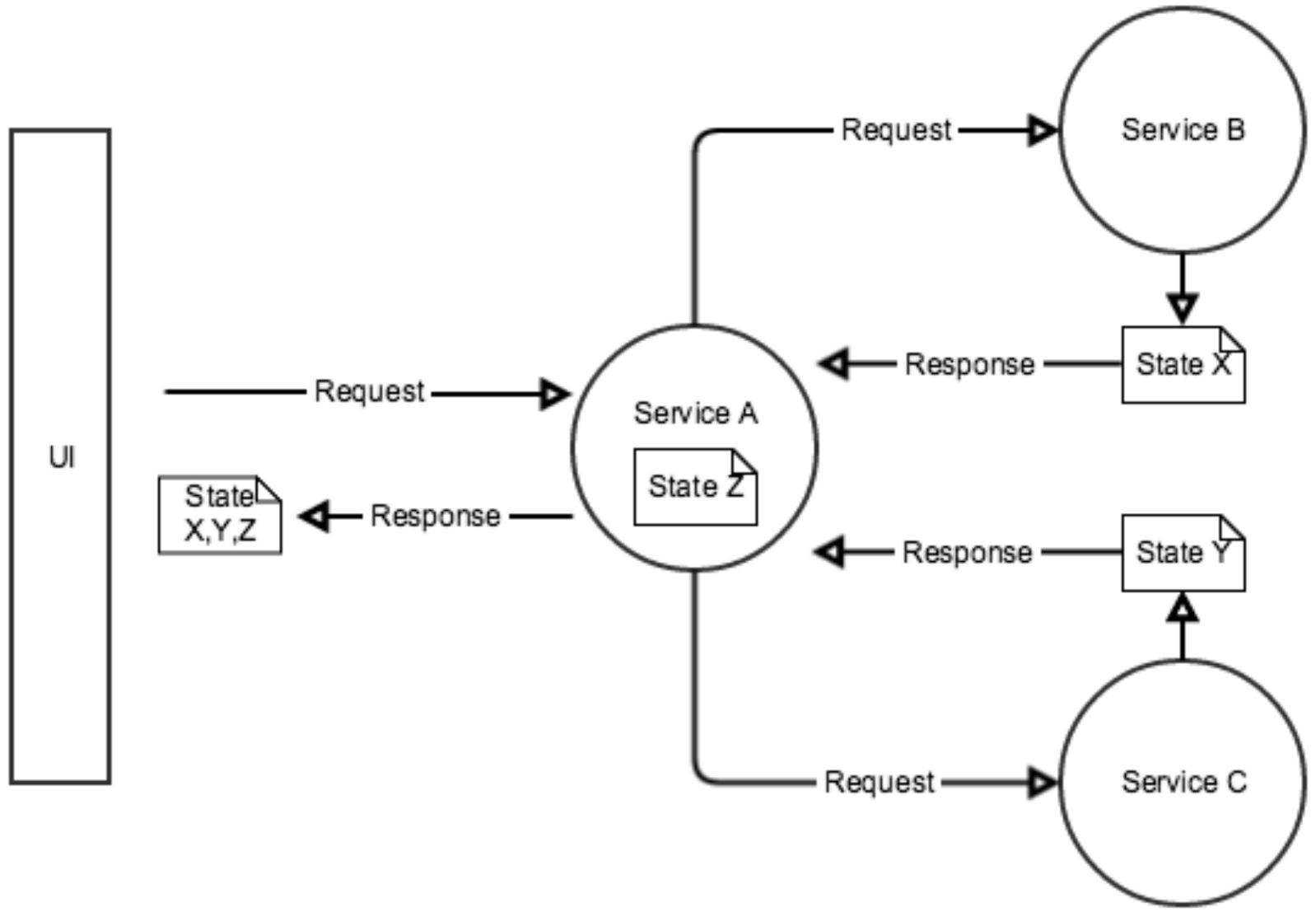
Webhooks would then be like asking the retailer to call you whenever they have the shoes in stock, which frees up time on both sides.

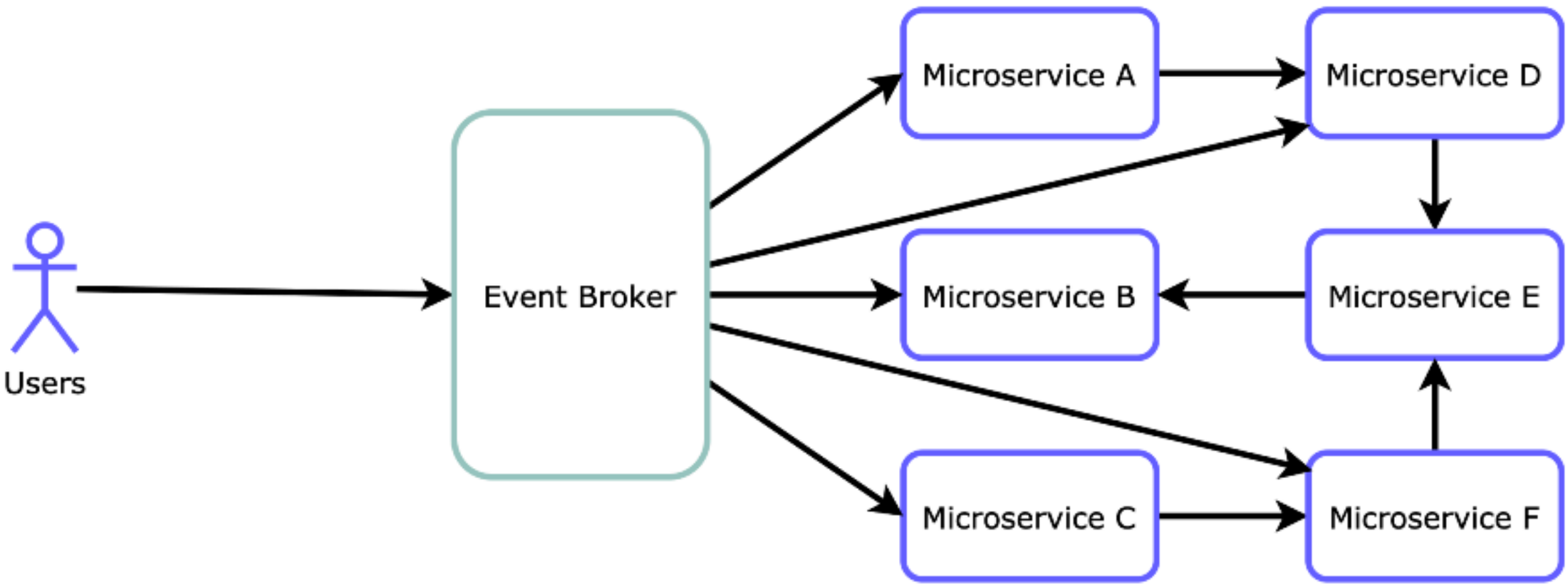

Polling
Your App
Endpoint
Anything?
Nothing.
Anything?
Nothing.
Anything?
Event Data!!!
Anything?
Nothing.
Webhooks
Your App
Endpoint
Event Data!!!
Sweet!



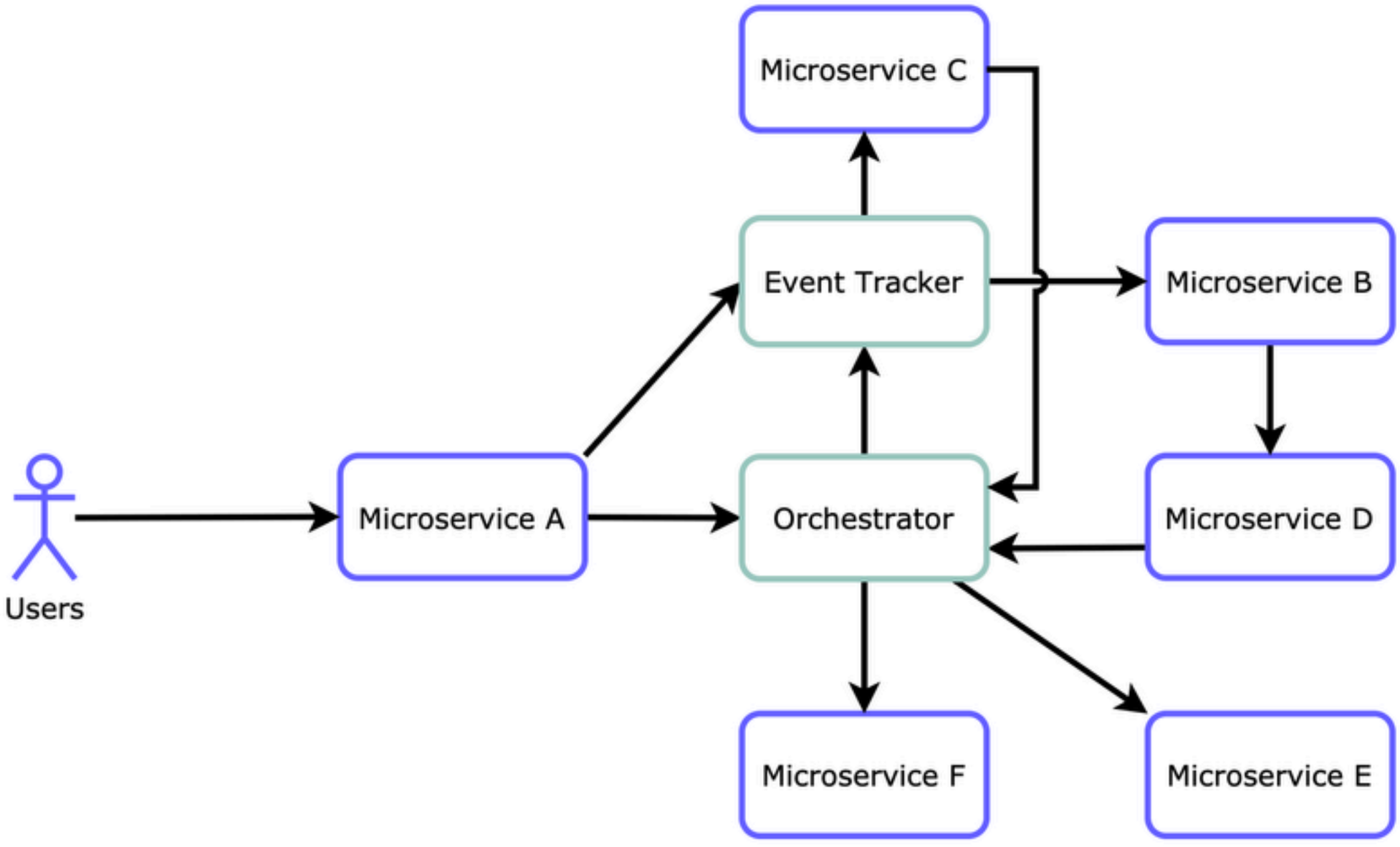
https://www.enterpriseintegrationpatterns.com/patterns/messaging/

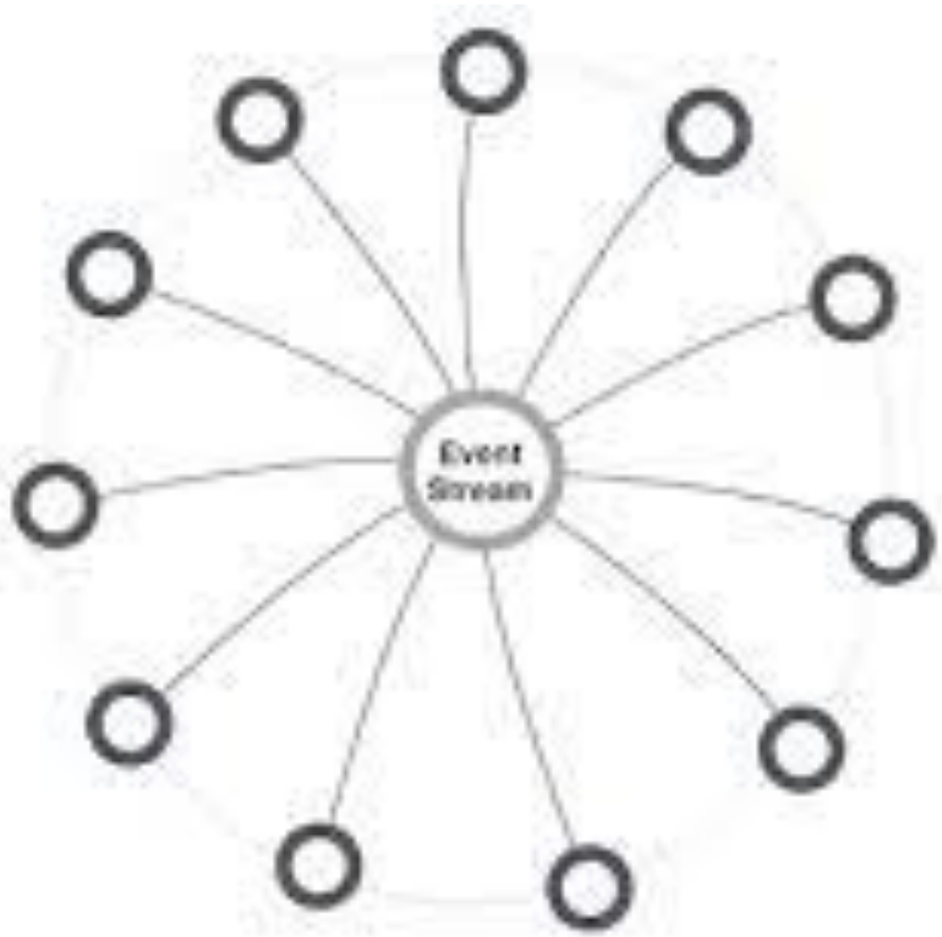

Microservice B
Microservice A
Microservice D
Users
Orchestrator
Microservice C
Microservice E
Microservice F


UI
Request
Service A
State Z
Request
Service B
Response
State X
State X,Y,Z
Response
Response
State Y
Request
Service C

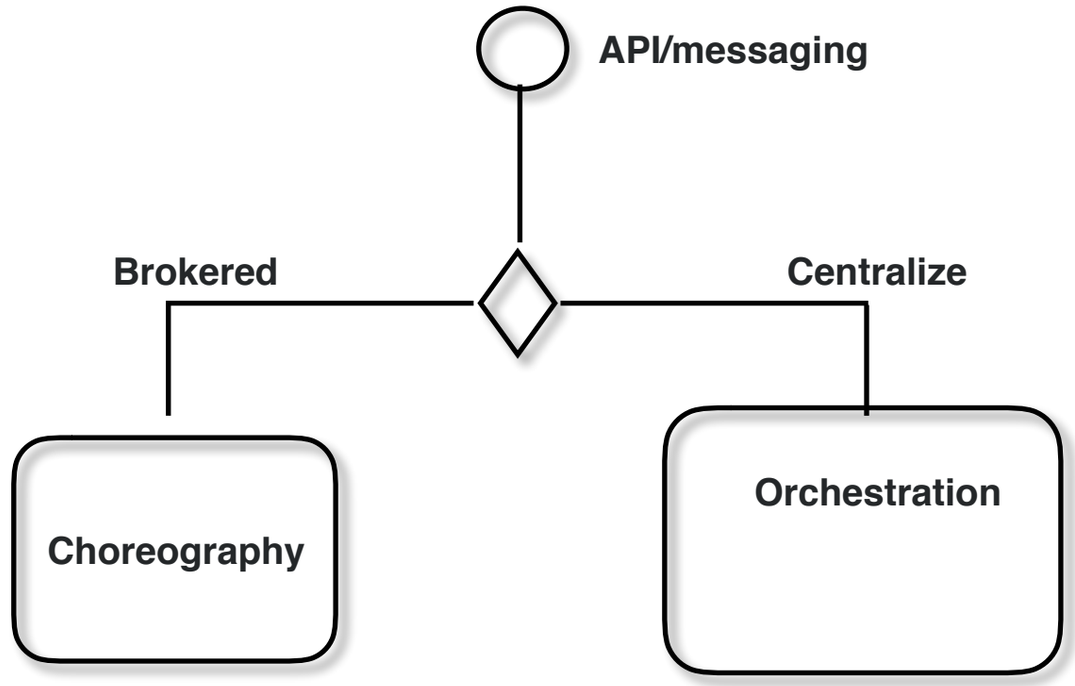

Microservice A
Microservice D
Users
Event Broker
Microservice B
Microservice E
Microservice C
Microservice F

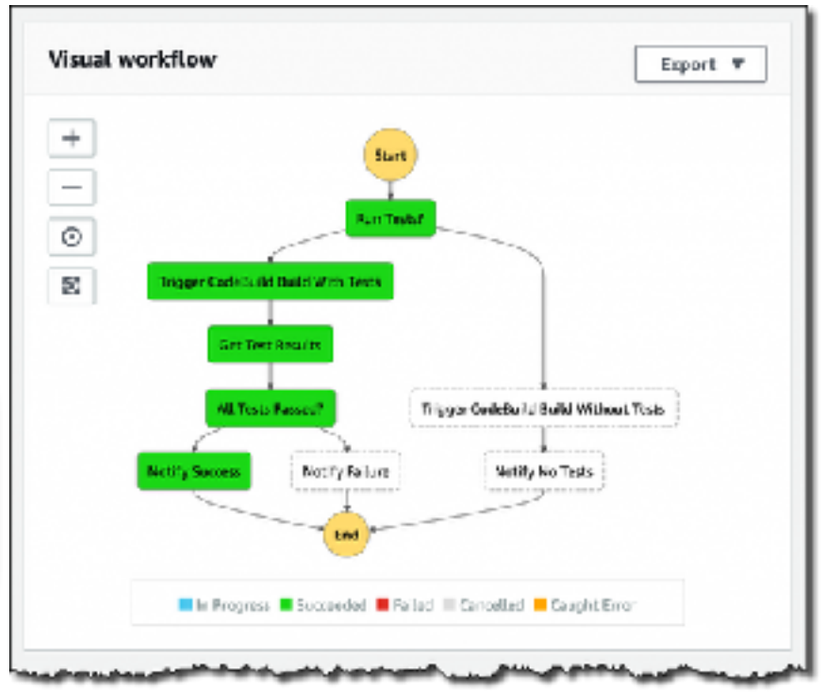
Hybrid


Microservice C
Event Tracker
Microservice B
Users
Microservice A
Orchestrator
Microservice D
Microservice F
Microservice E

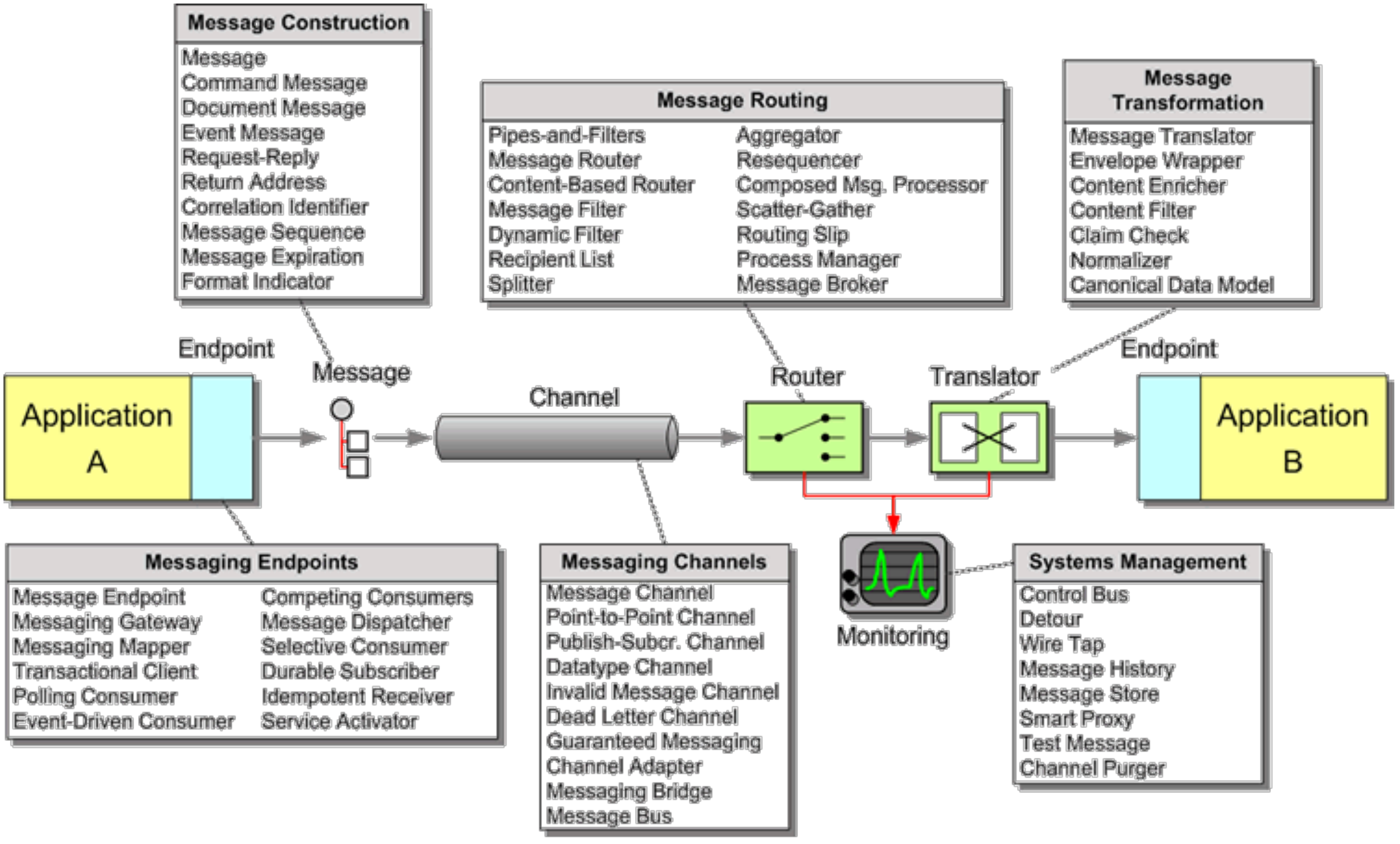

Event
Stream

Choose Communication Patterns

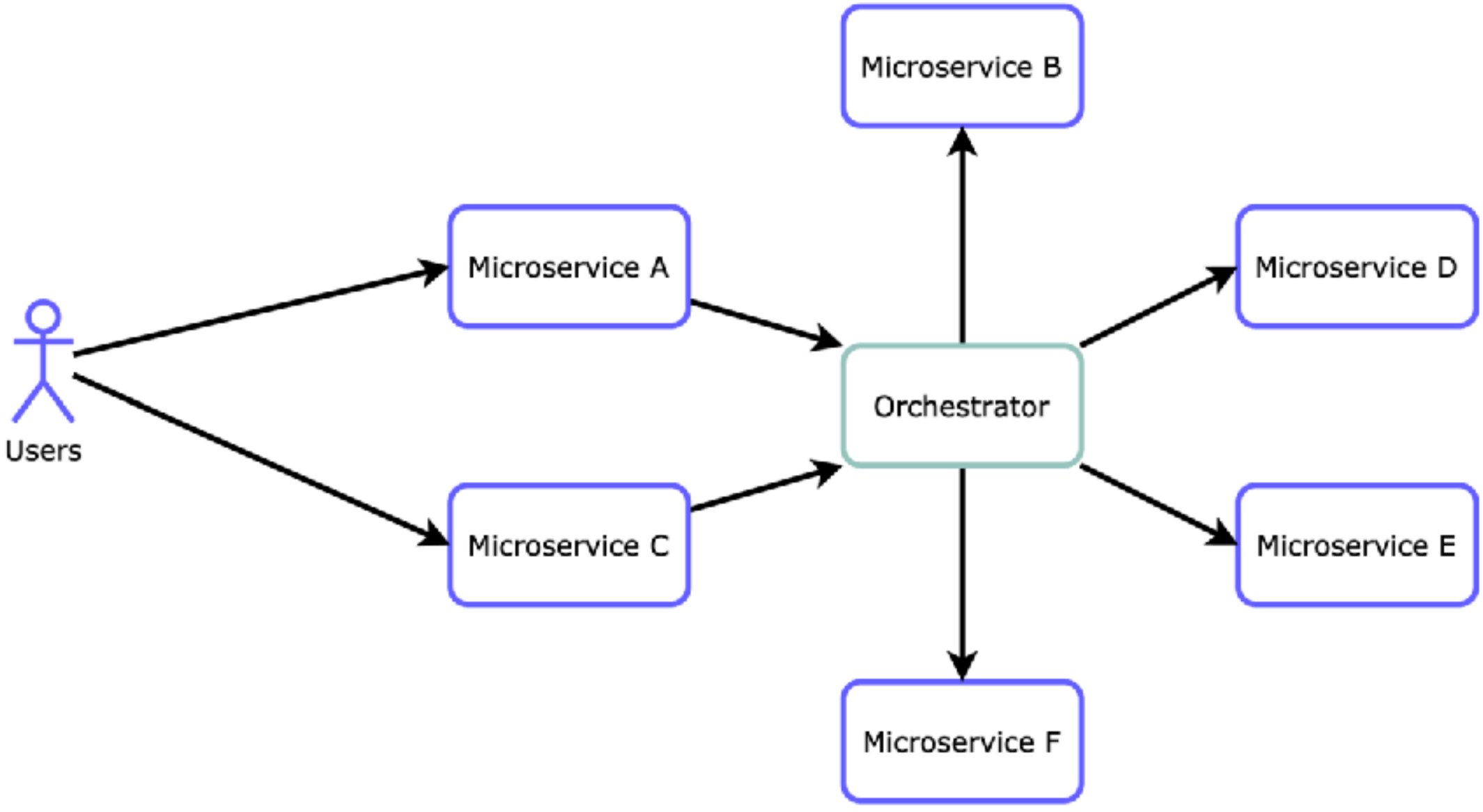


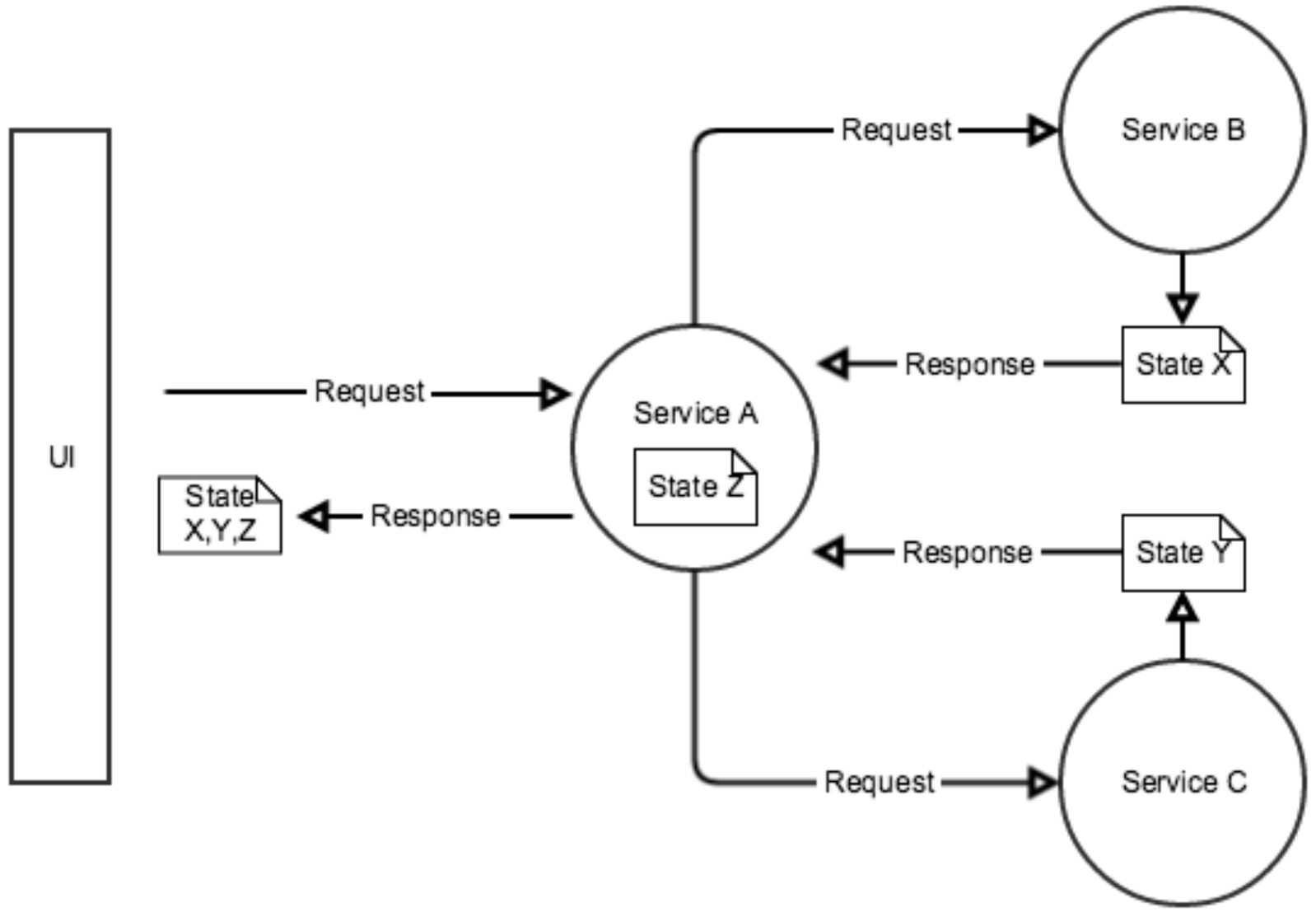


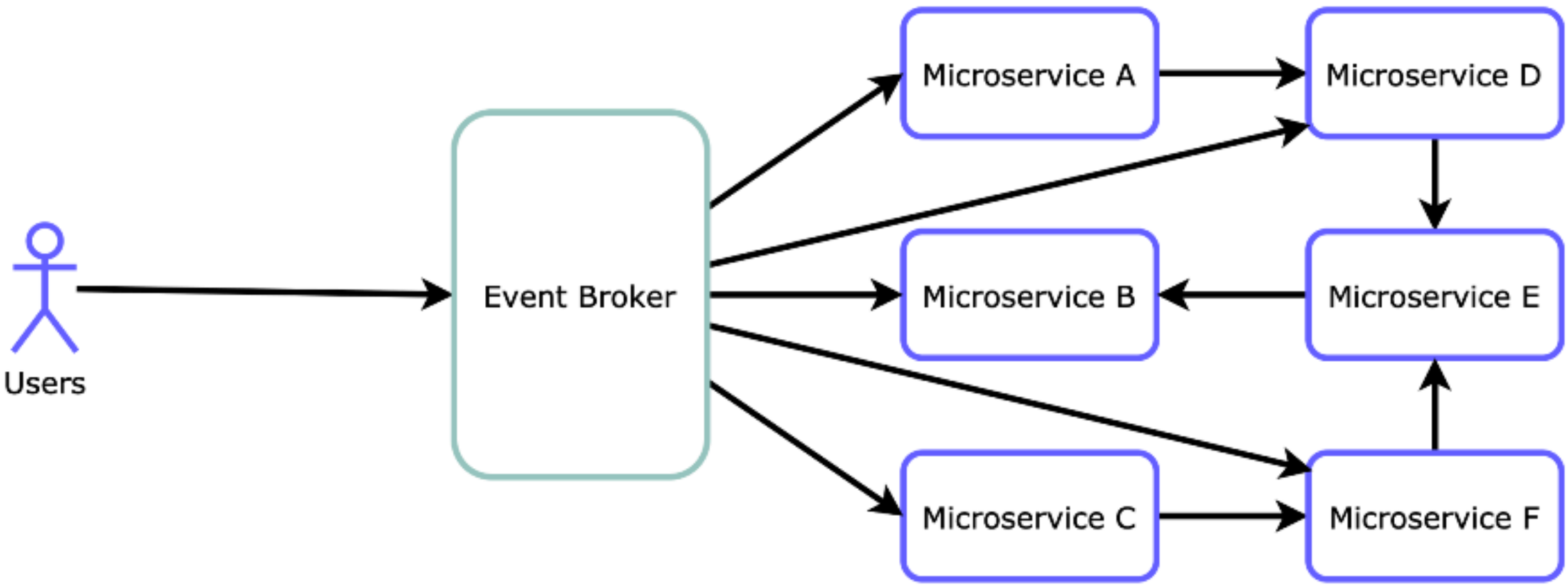
 # AWS Step
 # Azure Logic App
 # Biztalk
 # Webmethods
 # Mule
 # ServiceNow
#



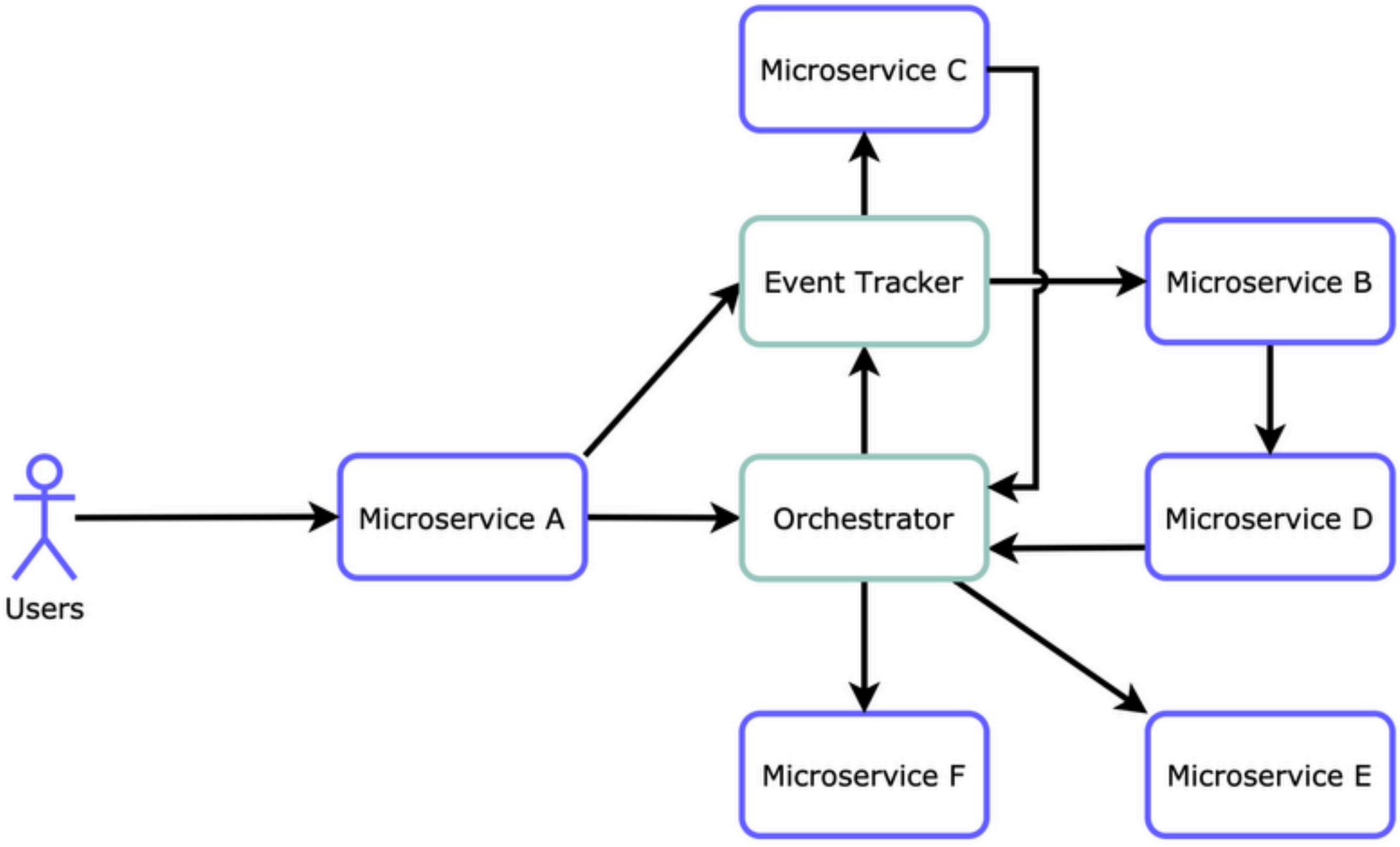
https://www.enterpriseintegrationpatterns.com/patterns/messaging/

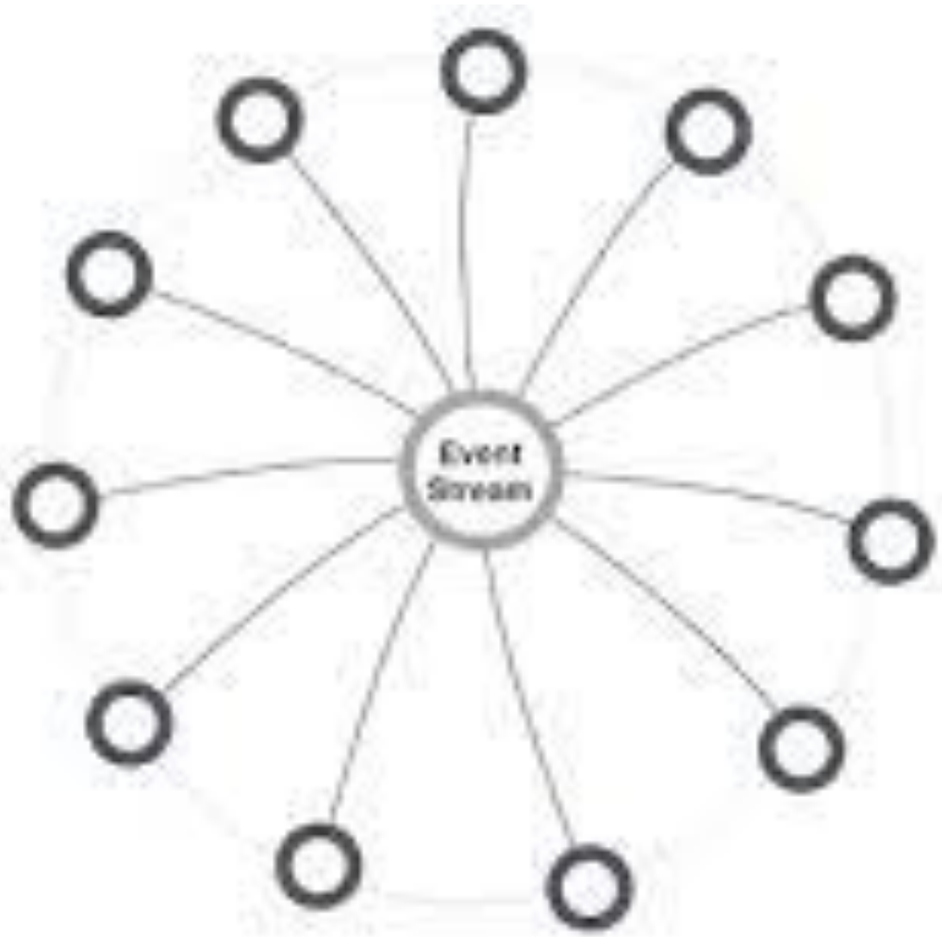

Microservice B
Microservice A
Microservice D
Users
Orchestrator
Microservice C
Microservice E
Microservice F


UI
Request
Service A
State Z
Request
Service B
Response
State X
State X,Y,Z
Response
Response
State Y
Request
Service C


Microservice A
Microservice D
Users
Event Broker
Microservice B
Microservice E
Microservice C
Microservice F

Hybrid


Microservice C
Event Tracker
Microservice B
Users
Microservice A
Orchestrator
Microservice D
Microservice F
Microservice E


Event
Stream

Binary Storage

Object Storage

Immutable
$
Low Perf
High Volume
0.01$/GB

- Azure Blobs
- AWS S3
- google Cloud Storage

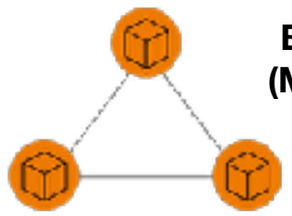
4

File Storage

Mutable
$$$$
Medium Perf (ex: 500 ms)
Medium Voulme

- Azure Files
- AWS EFS
- Google Filestore
- NFS

3



Block Storage
(Managed Disk)

Mutable
$$$
High Perf (10 ms)
Low Volume
1$/GB

- Azure Disks
- AWS EBS
- Google Persistent Disk

2

Unmanaged Disk

Mutable
$
High Perf

- Local Disk

1

Access across VM
(Network Storage)

Single VM Storage

Rest API /
SOAP protocol

Network File System (NFS) Protocol /
Server Message Block (SMB) protocol

Internet

Lan

Persistent

Temporary

* AWS DataSync subscription is required to provide support for Server Message Block (SMB) protocol

Operational (OLTP)

Structured

**[ACID - transaction,
LIMITED SCALABLE - not distributed
Fixed Schema]**

RDBMS (OLTP)

- AWS RDS
- Azure SQL
- Google Cloud Sql
- Google Cloud Spanner
- **Postgres**
- **Mysql**

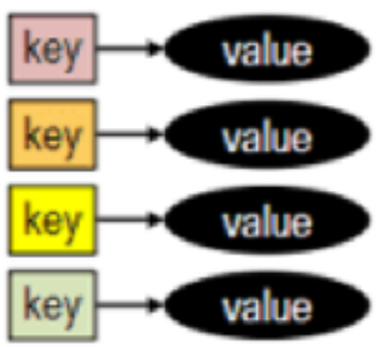
1

Semi Structured

**[* BASE,
SCALE- shareded (distributed),
* Flexible Schema]**

Dictionary (key value)

- AWS Dynamo Db
- Azure Cosmos -> Table
- Azure Table
- Google Big Table



2

Nested Objects

Document

- AWS Document DB
- Azure Cosmos -> Document Db
- Firestore
- **Mongodb**, **Couchbase**,Solr

3

Entity Relationship

Graph

- AWS Neptune
- Azure Cosmos -> Graph
- Google Janusgraph
- **Neo4j**
- Graph,orientdb

4

Rows vs. Documents

	Table
Row 1	Data
Row 2	Data
Row 3	Data
...	:

Document 1

```
{
  data
]
```

Document 2

```
{
  data
}
```

Document 3

```
{
  data
]
```

...

Columns vs. Properties

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

```
{
```

Schema vs. Schema-Free

ID	Name	IsActive	Dob
1	John Smith	True	8/30/1964
2	Sarah Jones	False	2/18/2002
3	Adam Stark	True	7/13/1987

Document 1

```
[
  "id": "1",
  "name": "John Smith",
  "isActive": true,
  "dob": "1964-30-08"
]
```

Document 2

```
{
  "id": "2",
  "fullName": "Sarah Jones",
  "isActive": false,
  "dob": "2002-02-18"
}
```

Document 3

```
{
  "id": "3",
  "fullName":
  {
    "first": "Adam",
    "last": "Stark"
  },
  "isActive": true,
  "dob": "2015-04-19"
}
```

Normalized vs. Denormalized

User Table

UserID	Name	Dob
1	John Smith	8/30/1964

Holdings Table

StockID	UserID	Qty	Symbol
1	1	100	MSFT
2	1	75	WMT

Document

```
{
  "id": "1",
  "name": "John Smith",
  "dob": "1964-30-08",
  "holdings": [
     { "qty": 100, "symbol": "MSFT" },
     { "qty": 75, "symbol": "WMT" }
   ]
}
```

Counter

C

BP

Counter

C

BP

Counter

E

BP

Counter

?

BP

General

C

E

?

BP

BP

BP

BP

BP

BP

BP

General

C

E

?

BP

D BP

I BP

BP

D BP

I BP

General

C

E

?

D BP

D BP

D BP

D BP

D BP

D BP

D BP

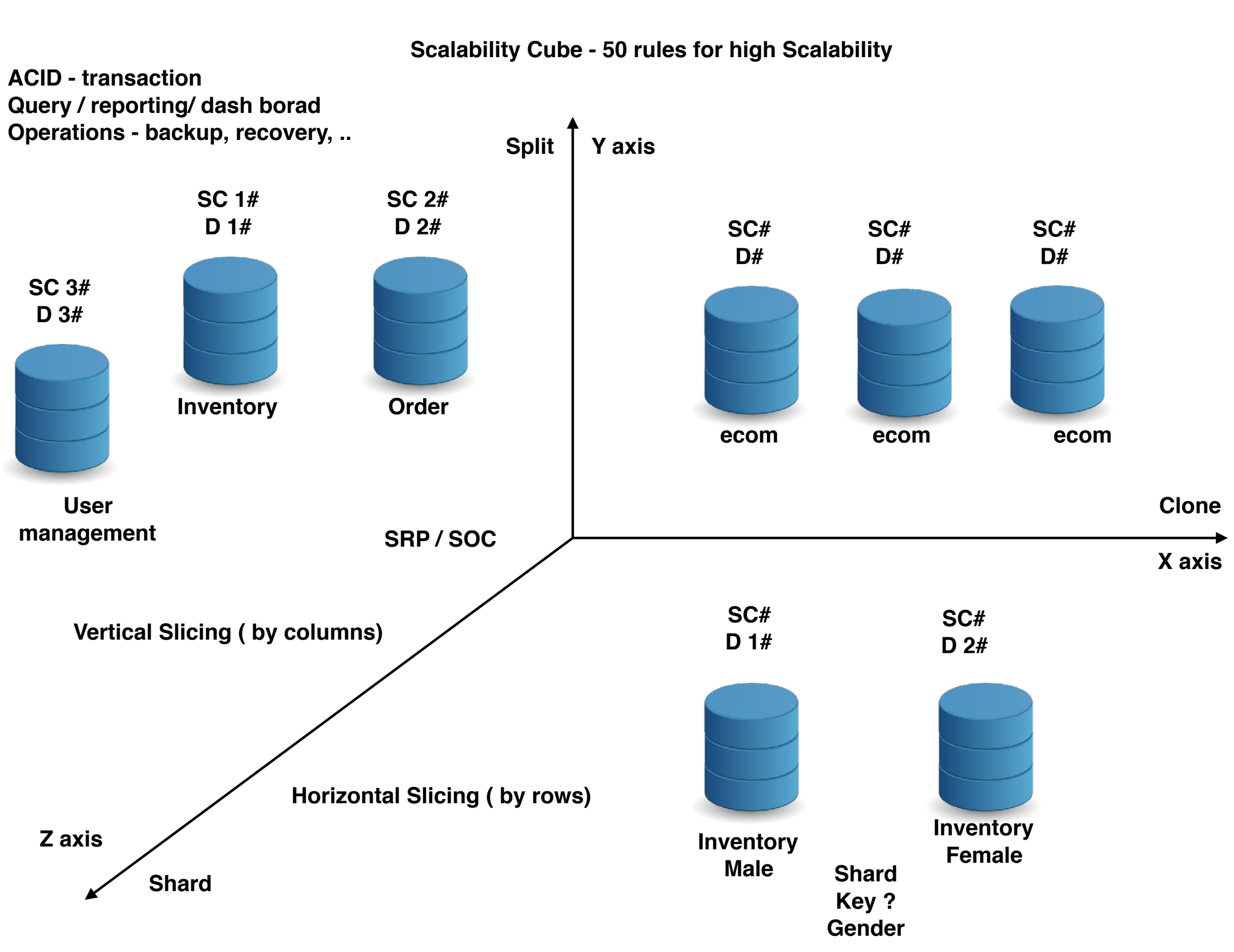
I BP

I BP

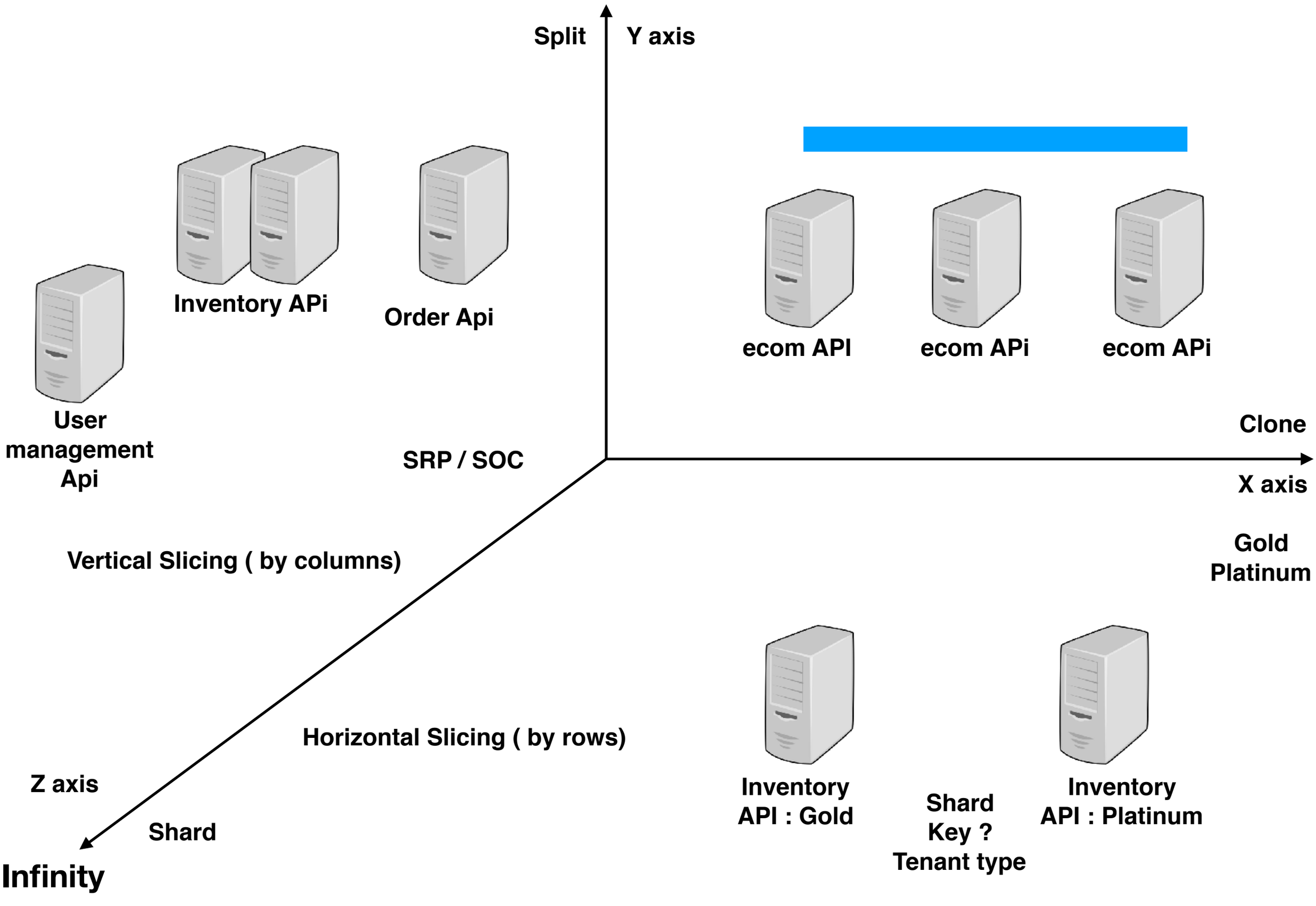
I BP

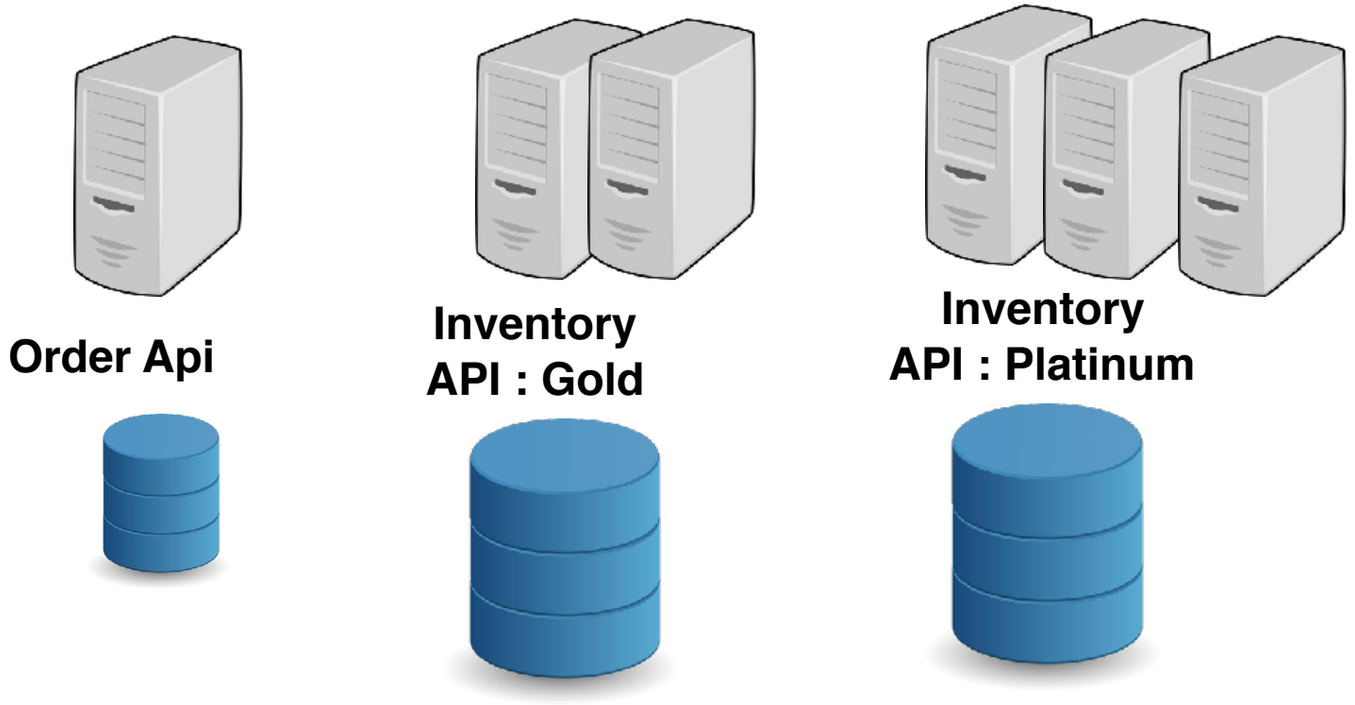
Scalability Cube - 50 rules for high Scalability

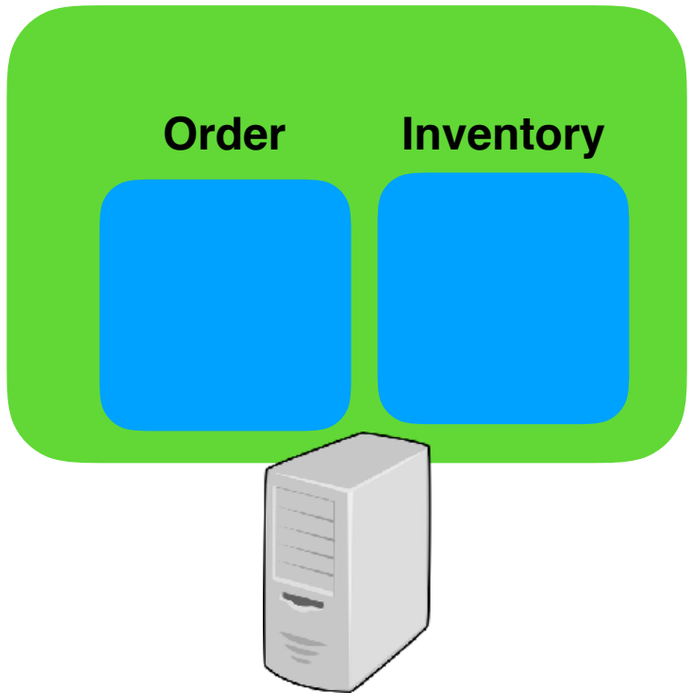
ACID - transaction
Query / reporting/ dash borad
Operations - backup, recovery, ..



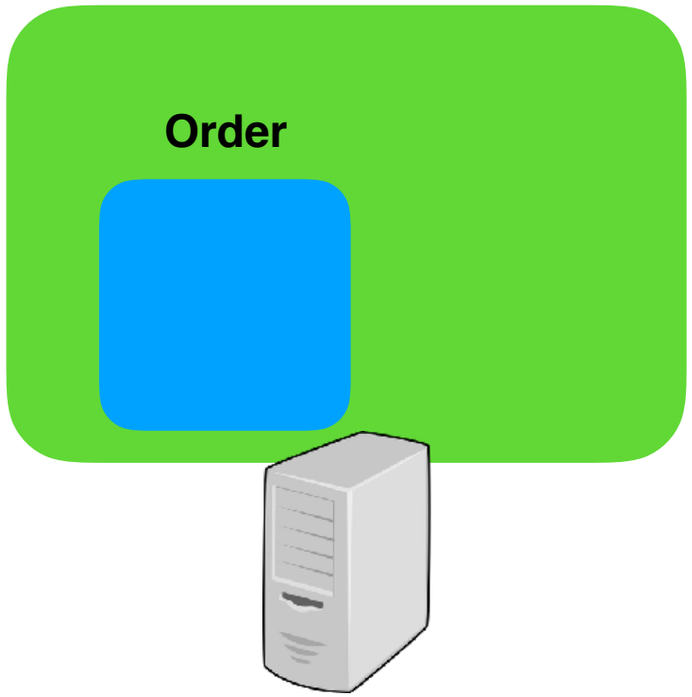
Scalability Cube - 50 rules for high Scalability



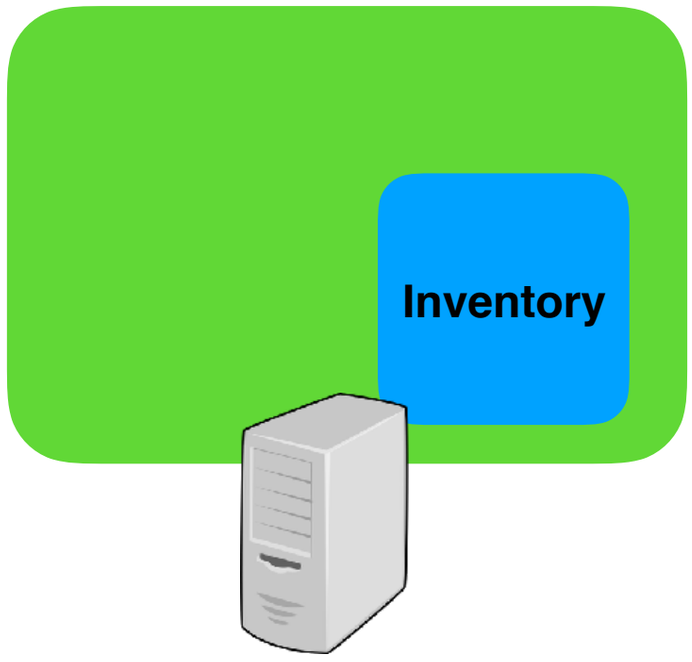

Order Api
Inventory
API : Gold
Inventory
API : Platinum



ecom

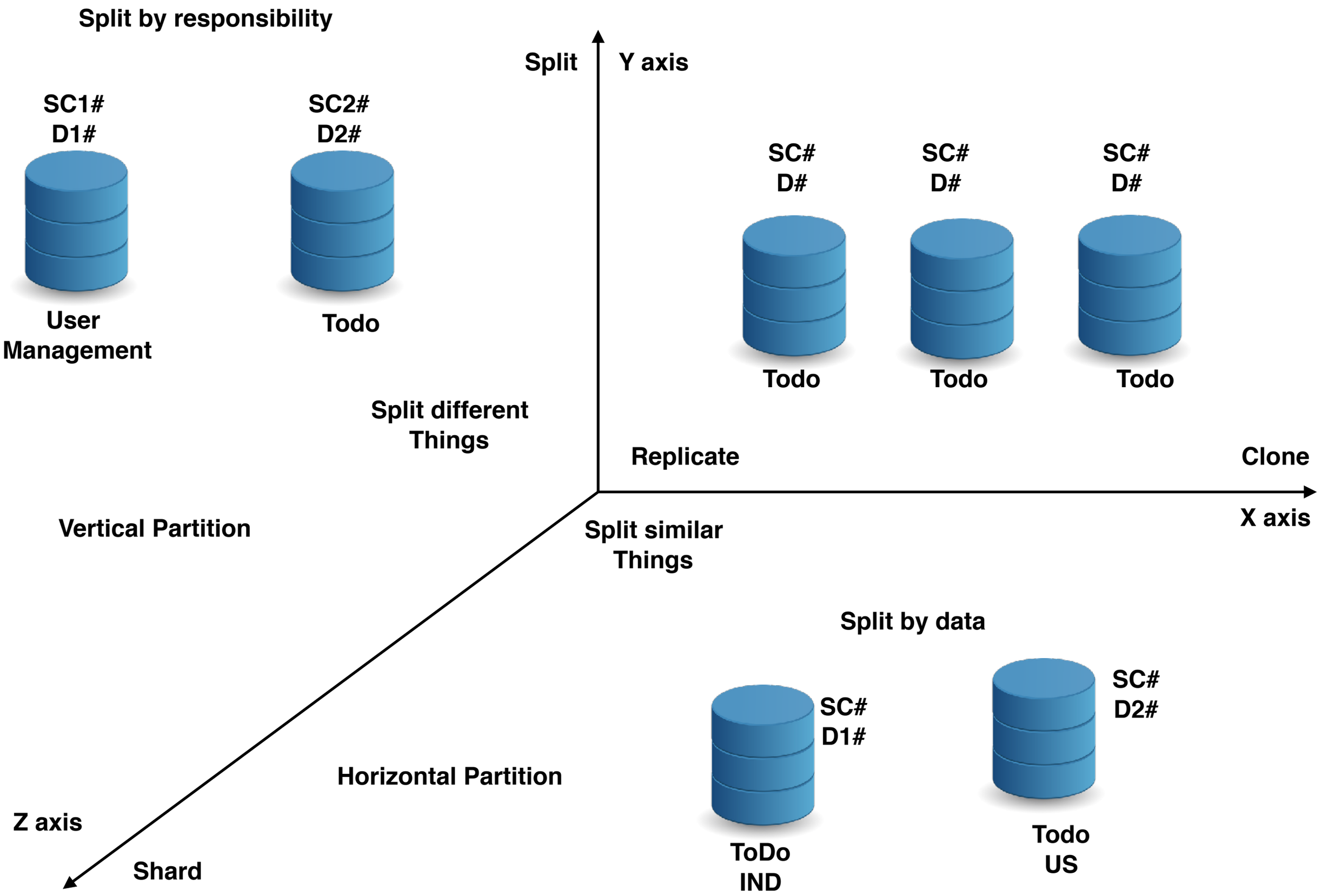


Order



Inventory

Scalability Cube - 50 rules for high Scalability



Ecom

Split different Things

Vertical Partition

User

Inventory

Accounts

Order

Horizontal Partition

Shard

Inventory: M

Inventory : F

Order: AUS

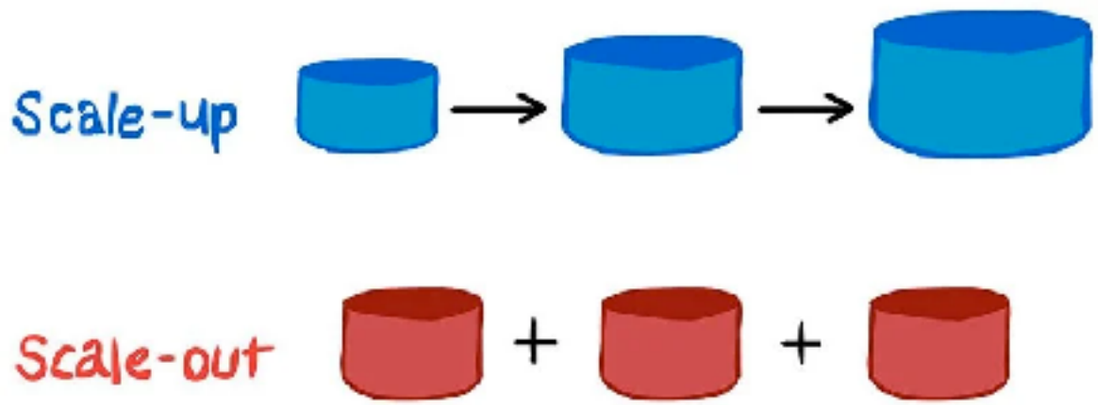
Order: USD

Clone

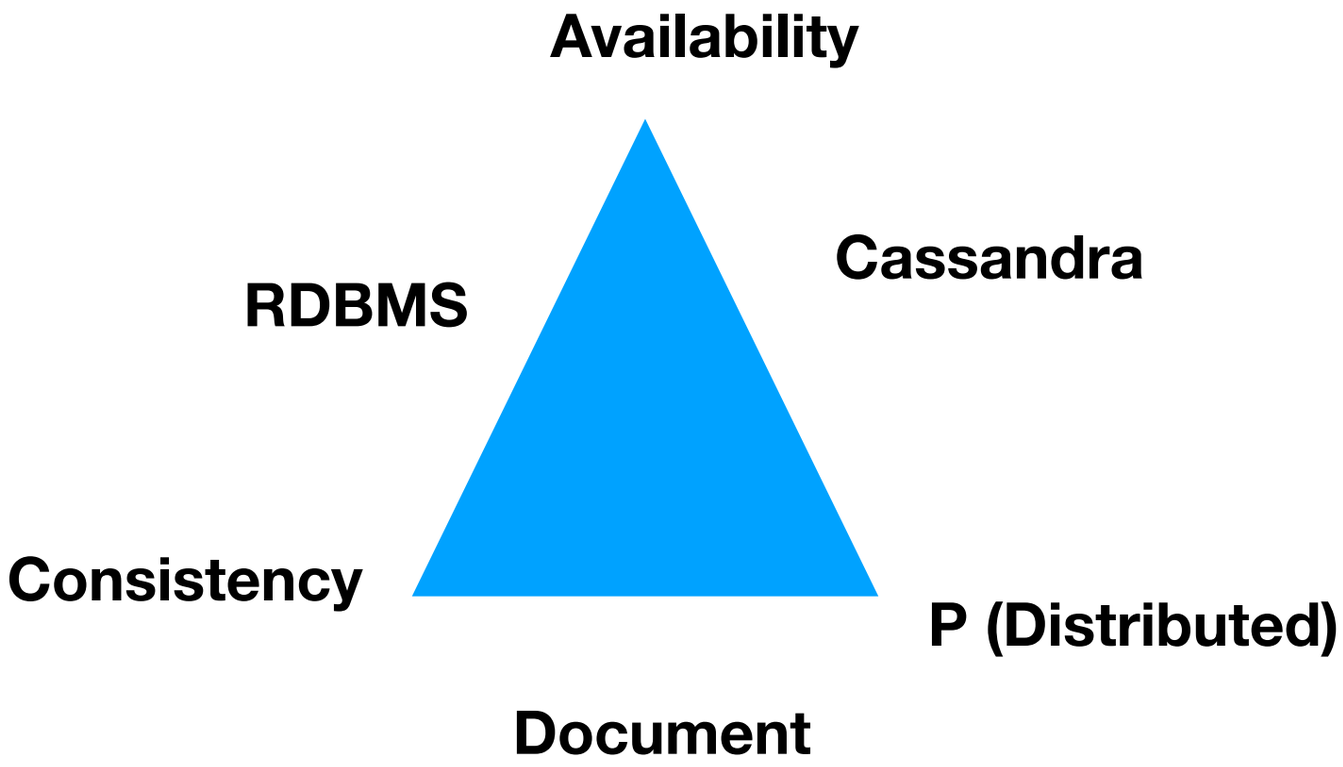
Inventory: M

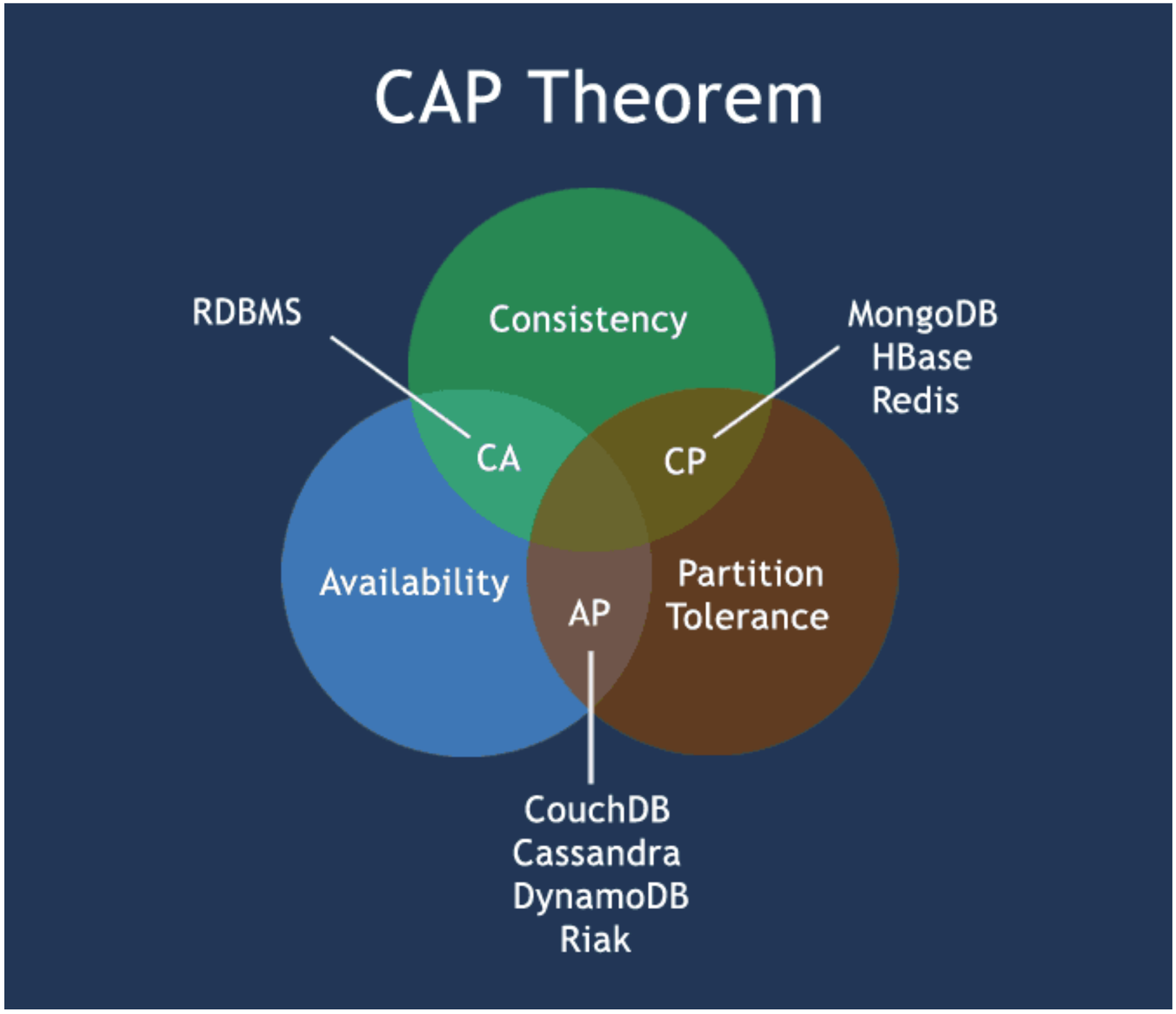
Inventory: M

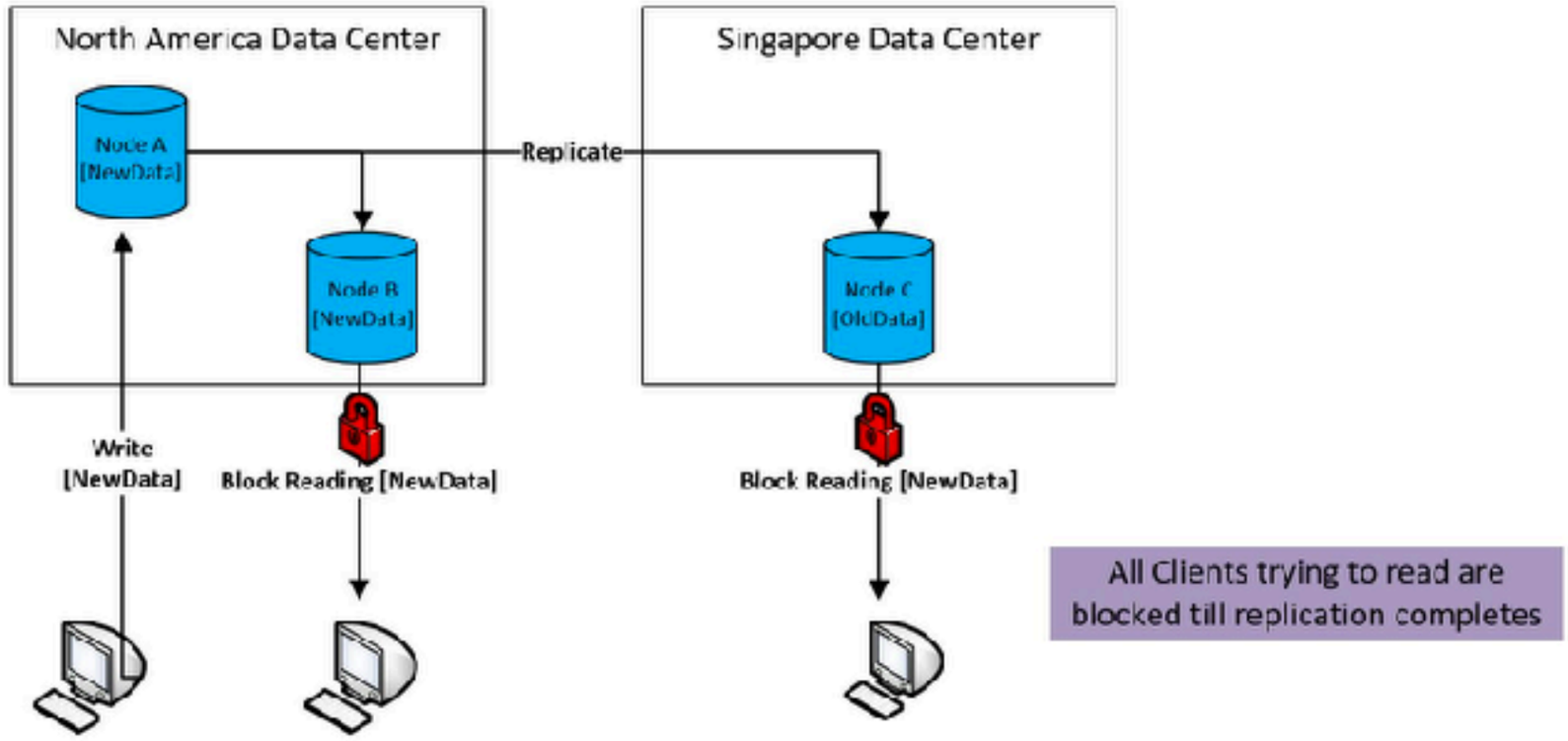
Scale-Up vs. Scale-Out

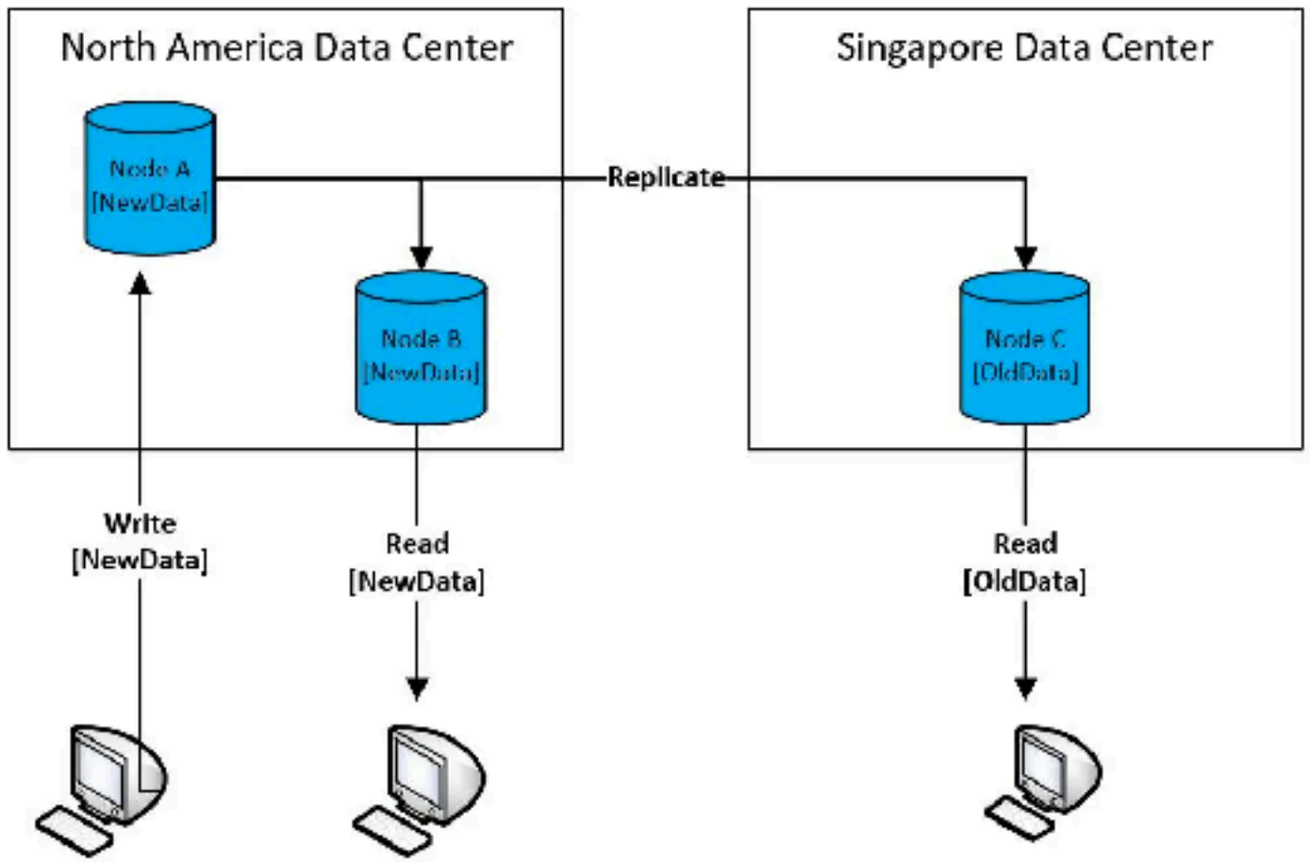


Strong Consistency vs. Eventual Consistency




CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak


North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Block Reading [NewData]
Block Reading [NewData]
All Clients trying to read are blocked till replication completes


North America Data Center
Singapore Data Center
Node A [NewData]
Replicate
Node B [NewData]
Node C [OldData]
Write [NewData]
Read [NewData]
Read [OldData]

COMMON COMPARISONS BETWEEN

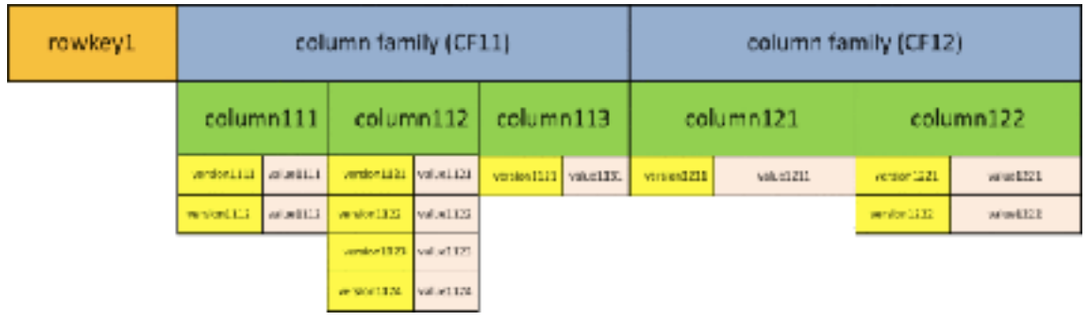
MYSQL & NOSQL

	MySQL	NoSQL
Nature	Relational Database	Non-Relational Database
Design	Based on the concept of tables	Based on the concept of documents
Scalable	Tough to scale due to its relational nature	Easily scalable big data compared to relational
Model	Detailed database model is needed before creation	No need of a detailed database model
Community	Vast community available	Community is growing rapidly, but still smaller compared to MySQL
Standardization	SQL is standard language	Lacks standard query language
Schema	The Schema is rigid	The Schema is dynamic
Flexibility	Not very flexible in terms of design	Very flexible in terms of design
Insertions	Inserting new columns or fields affect the design	No effect on the design with the insertion of new columns or fields

Facebook, Wikipedia, Quora, Flickr	MySQL
Twitter	MySQL for tweets and users their own special kind of graph database, FlockDB, built on top of MySQL their own version of Memcached
LinkedIn	Oracle Database and Voldemort
YouTube	MySQL -> BigTable
Microsoft, Myspace	***SQL Server***
Yahoo	***PostgreSQL***

Key Value	○ Twitter uses Redis to deliver your Twitter timeline
	○ Pinterest uses Redis to store lists of users, followers, unfollowers, boards, and more
	○ Coinbase uses Redis to enforce rate limits and guarantee correctness of Bitcoin transactions
	○ Quora uses Memcached to cache results from slower, persistent databases
Graph	○ Walmart uses Neo4j to provide customers personalized, relevant product recommendations and promotions
	○ Medium uses Neo4j to build their social graph to enhance content personalization
	○ Cisco uses Neo4j to mine customer support cases to anticipate bugs
Document	○ SEGA uses MongoDB for handling 11 million in-game accounts
	○ Cisco moved its VSRM (video session and research manager) platform to Couchbase to achieve greater scalability
	○ Aer Lingus uses MongoDB with Studio 3T to handle ticketing and internal apps
	○ Built on MongoDB, The Weather Channel's iOS and Android apps deliver weather alerts to 40 million users in real-time

Analytical



Read

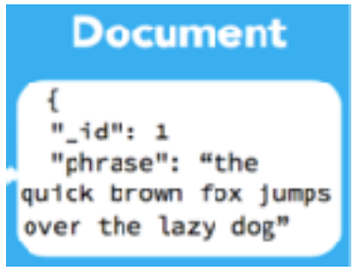
Columnar db (column store)

Cloud DWH (OLAP)

- AWS Redshit (max: 8 PB)
- **Azure Synapse**
- GCP BigQuery
- Snow flakes
- Kudu, Druid, Pinot
- Click house

Read

Text Database



- AWS **Elastic Search**
- AWS cloud Search
- Azure Cognitive Search
- Google Search API
- Elastic Search, Solr

Read/write

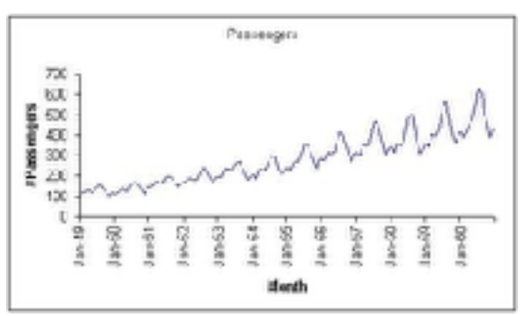
Wide Column db (column-family)

- AWS Keyspaces (unlimited)
- Cosmos
- Big Table
- **Hbase, cassandra**
- Scylla

KairosDB (cassandra)

Time Series

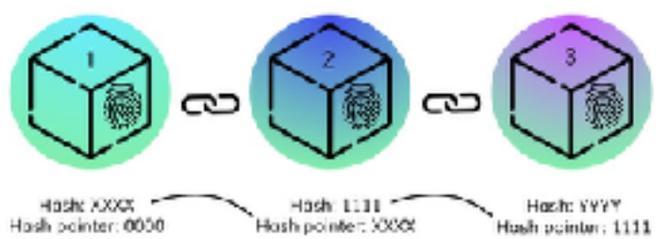
- AWS Timestream
- InfluxDB, Scylla
- OpenTSDB,
- KairosDB



Audit Trial

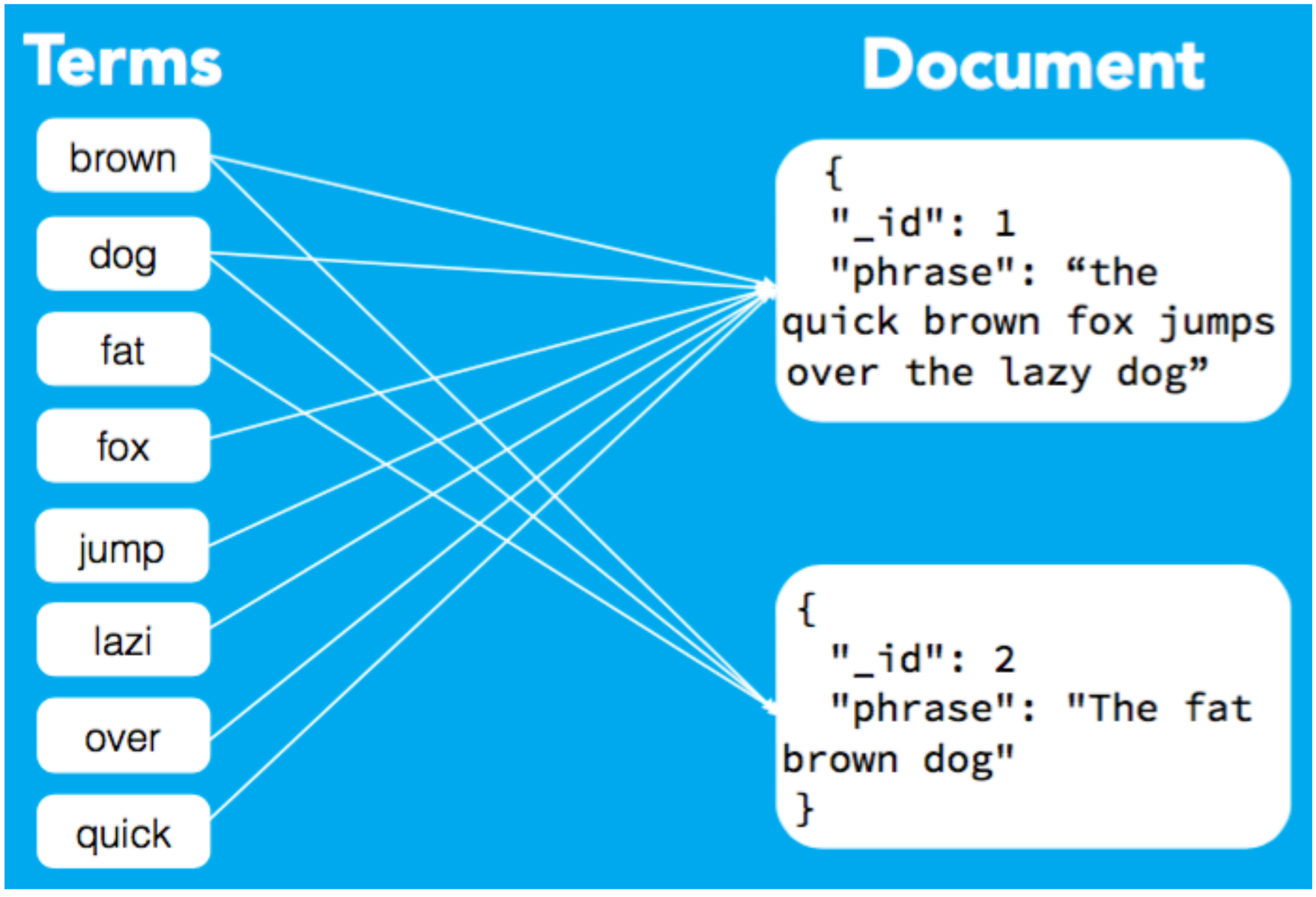
Immutable Logger

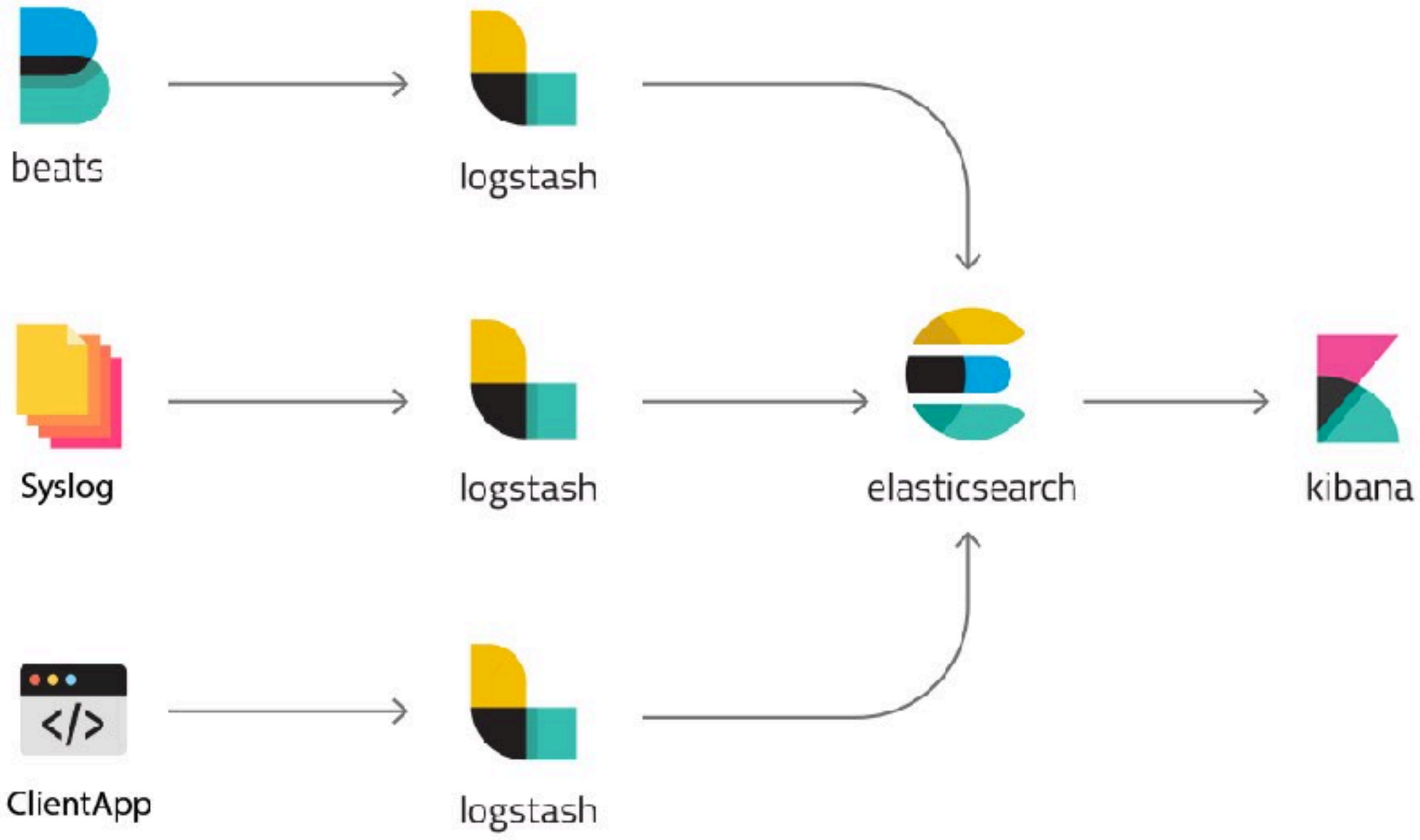
- AWS Quantum Ledger
- Azure SQL Database Ledger
- Google ?
- Hyper ledger Fabric

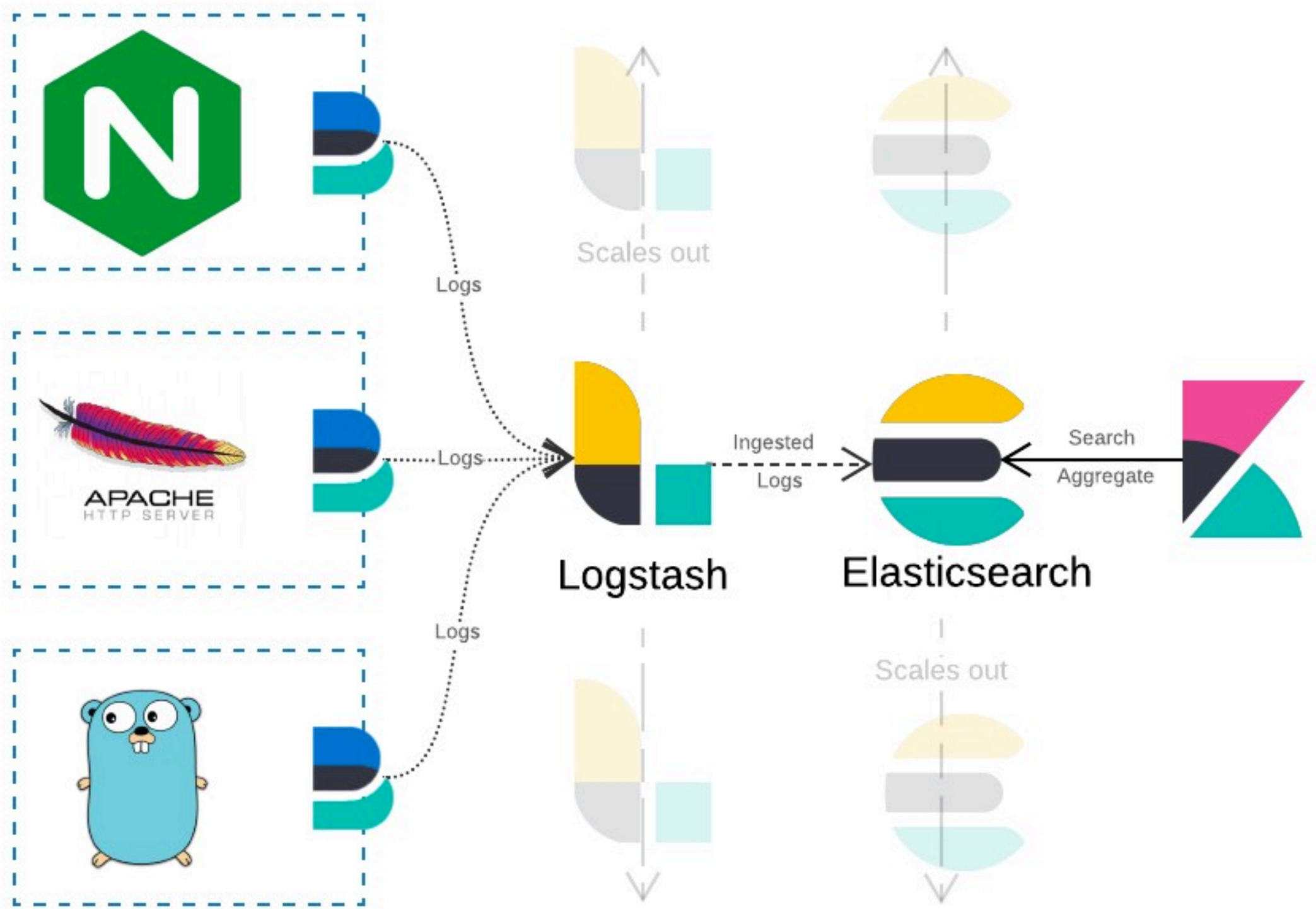


geo Spatial

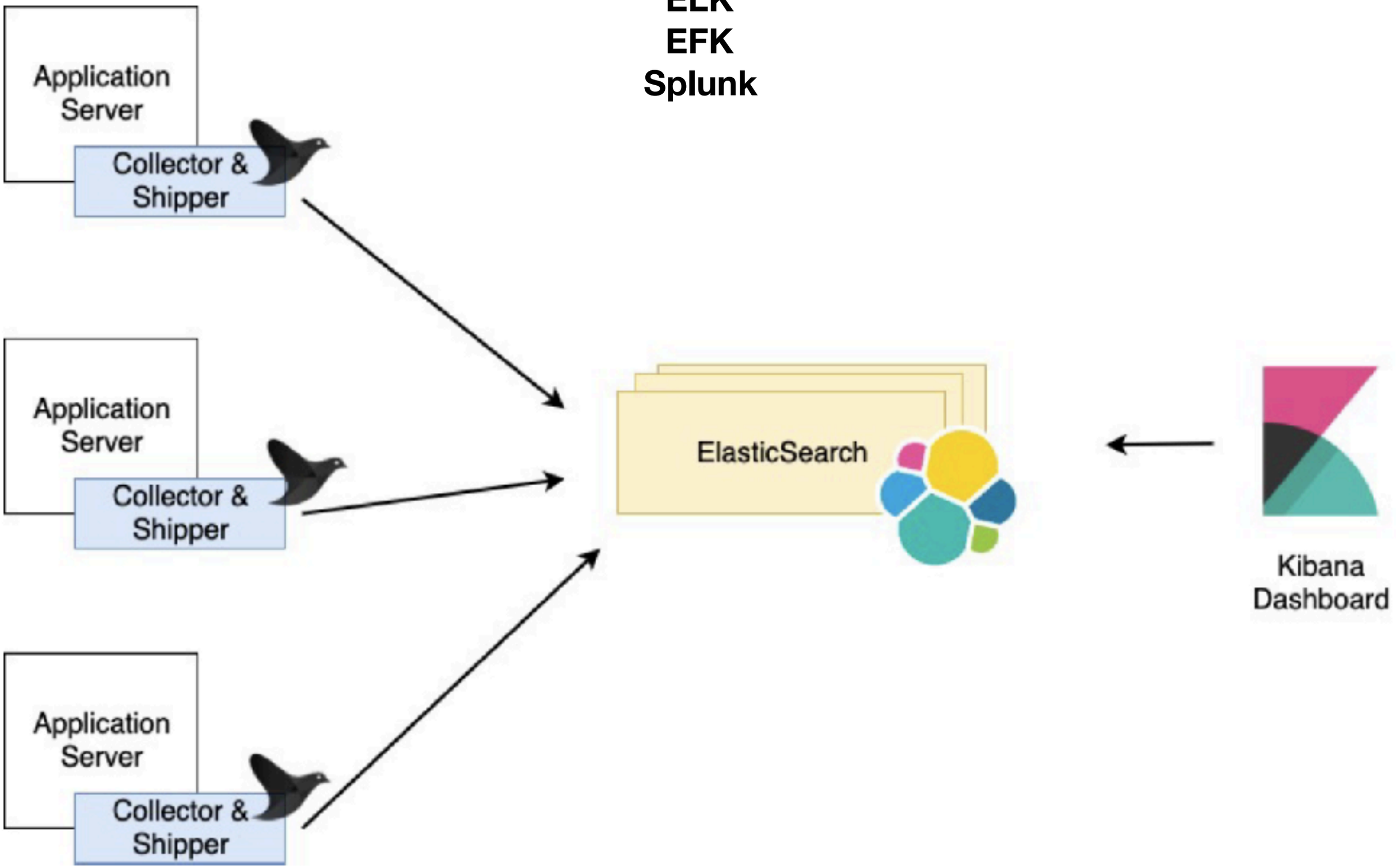
- AWS Keyspaces
- Azure Cosmos
- Big Table, Big Query
- **Solr**, PostGIS
- Mongoldb (GeoJSON)


Terms
Document
brown
dog
fat
fox
jump
lazi
over
quick
{
"_id": 1
"phrase": “the quick brown fox jumps over the lazy dog”
{
"_id": 2
"phrase": "The fat brown dog"
}


beats
logstash
Syslog
logstash
elasticsearch
kibana
ClientApp
logstash


Logs
Scales out
APACHE
HTTP SERVER
Logs
Ingested
Logs
Search
Aggregate
Logstash
Elasticsearch
Logs
Scales out

ELK
EFK
Splunk



Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

Row 1	India
	Chocolate
	1000
Row 2	India
	Ice-cream
	2000
Row 3	Germany
	Chocolate
	4000
Row 4	US
	Noodle
	500

Column Store

Country	India
	India
	Germany
	US
Product	Chocolate
	Ice-cream
	Chocolate
	Noodle
Sales	1000
	2000
	4000
	500

Row-oriented

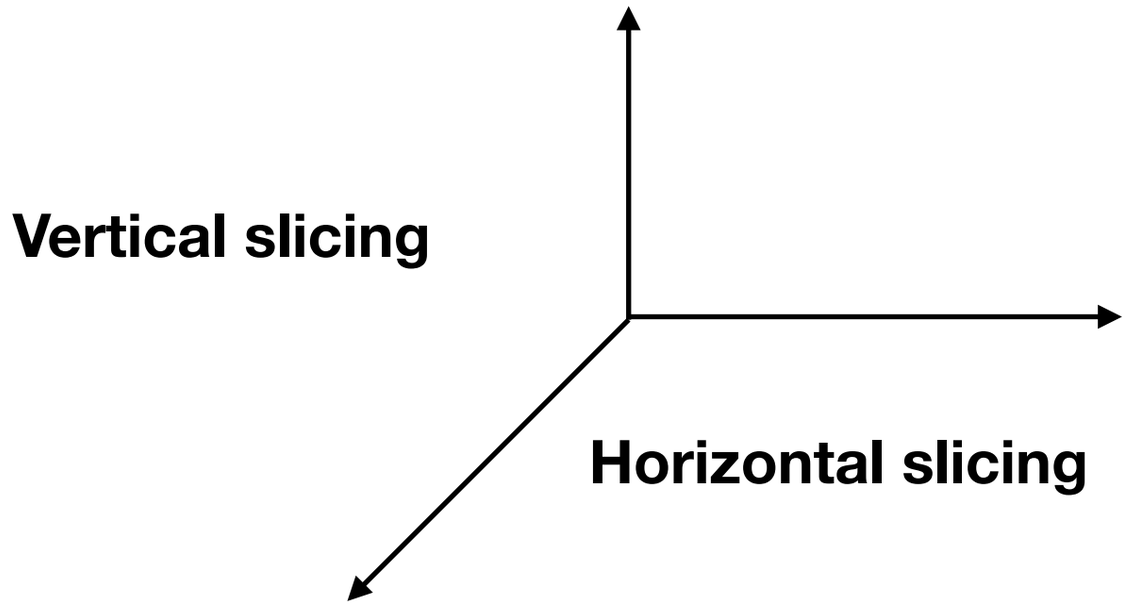
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003



rowkey1	column family (CF11)			column family (CF12)	
	column111	column112	column113	column121	column122
	version1111 value1111	version1121 value1121	version1121 value1131	version1211 value1211	version1221 value1221
	version1112 value1112	version1122 value1122			version1222 value1222
		version1123 value1123			
		version1124 value1124			

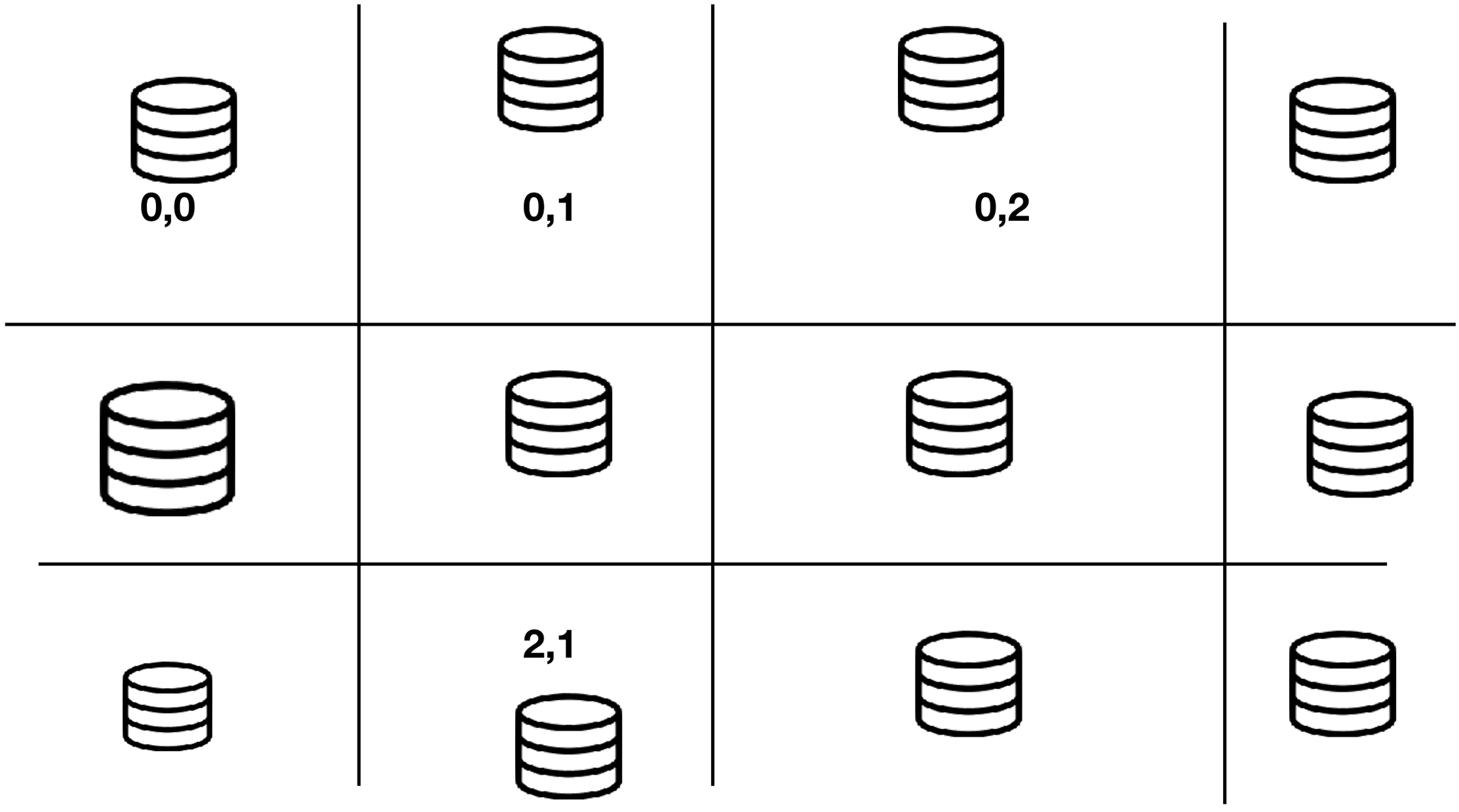
{Ordered, cust name, amount} , {itemcode, qty,price}

(Row partition id (shard key), column family id, row id)

multidimensional map (map of maps)

```
{
  "rowkey1": {"cf11": {"column111": {"version1111": value1111,
                                     "version1112": value1112},
                       "column112": {"version1121": value1121,
                                     "version1122": value1122,
                                     "version1123": value1123,
                                     "version1124": value1124},
                       "column113": {"version1131": value1131}
                      },
             "cf12": {"column121": {"version1211": value1211},
                       "column122": {"version1221": value1221},
                                     "version1222": value1222}
                      }
             },
   "rowkey2": {"cf11": {"column111": {"version2111": value2111,
                                      "version2112": value2112},
                        "column112": {"version2121": value2121,
                                      "version2122": value2122,
                                      "version2123": value 2123,
                                      "version2124": value2124}
                       },
              "cf12": {"column121": {"version2211": value2211},
                       "column122": {"version2221": value2221}
                       }
             }
}
```

Row partition id,
Col partition id



Column Family 1

Column Family 2

	cf1:col-A	cf1:col-B	cf2:col-Foo	cf2:col-XYZ	cf2:foobar
Region 1					
row-1					
row-10					
row-18	A18 - v1	B18 - v3	Foo18 - v1	XYZ18 - v2	foobar18 - v1
Region 2					
row-2					
row-5					
row-6					
row-7					

Physical Coordinates for a Cell: *Region Directory* → *Column Family Directory* → *Row Key* → *Column Family Name* → *Column Qualifier* → *Version*

	CF1:colA	CF1:colB	CF1:colC
Row1	@time7: value3		
Row10	@time2: value1	@time2: value1	
Row11	@time6: value2		
Row2	@time4: value1		@time4: value1

Row-oriented

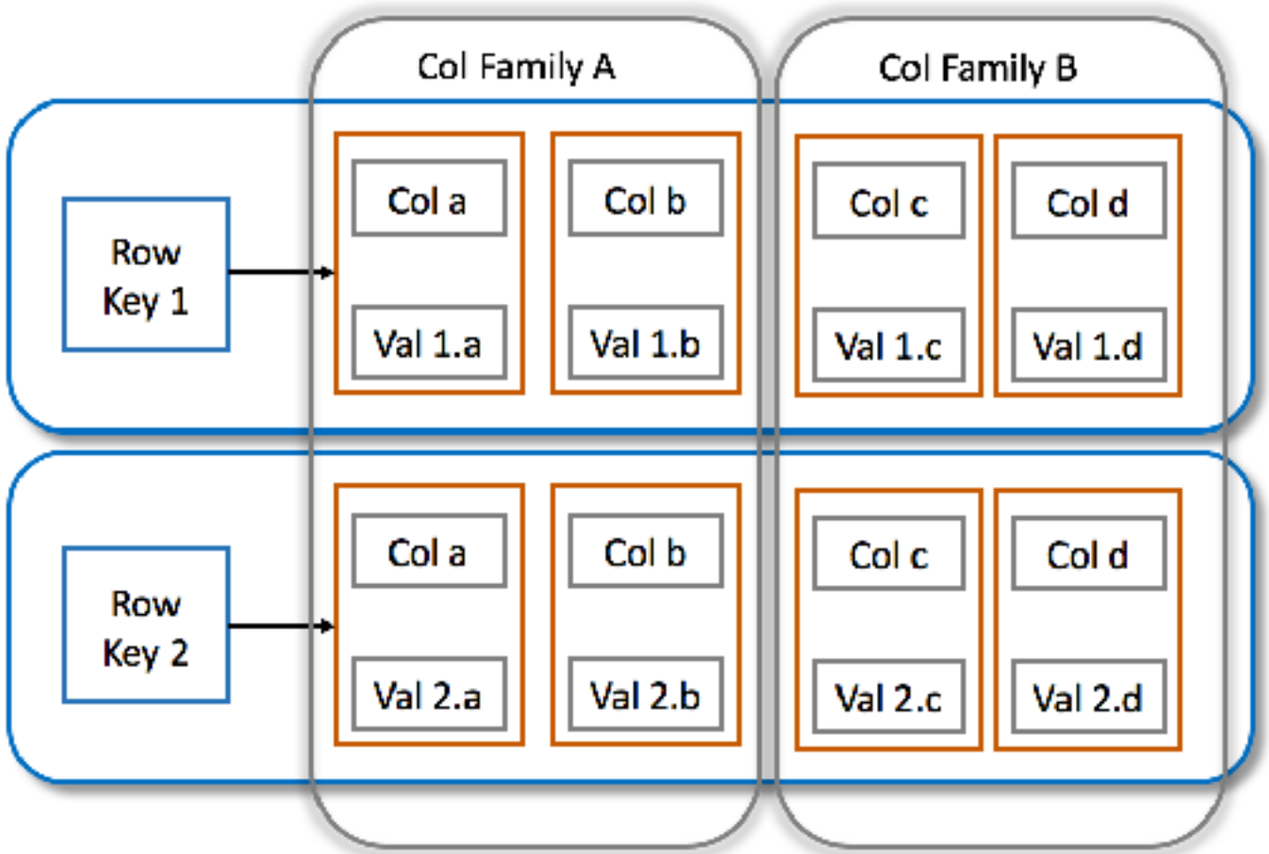
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

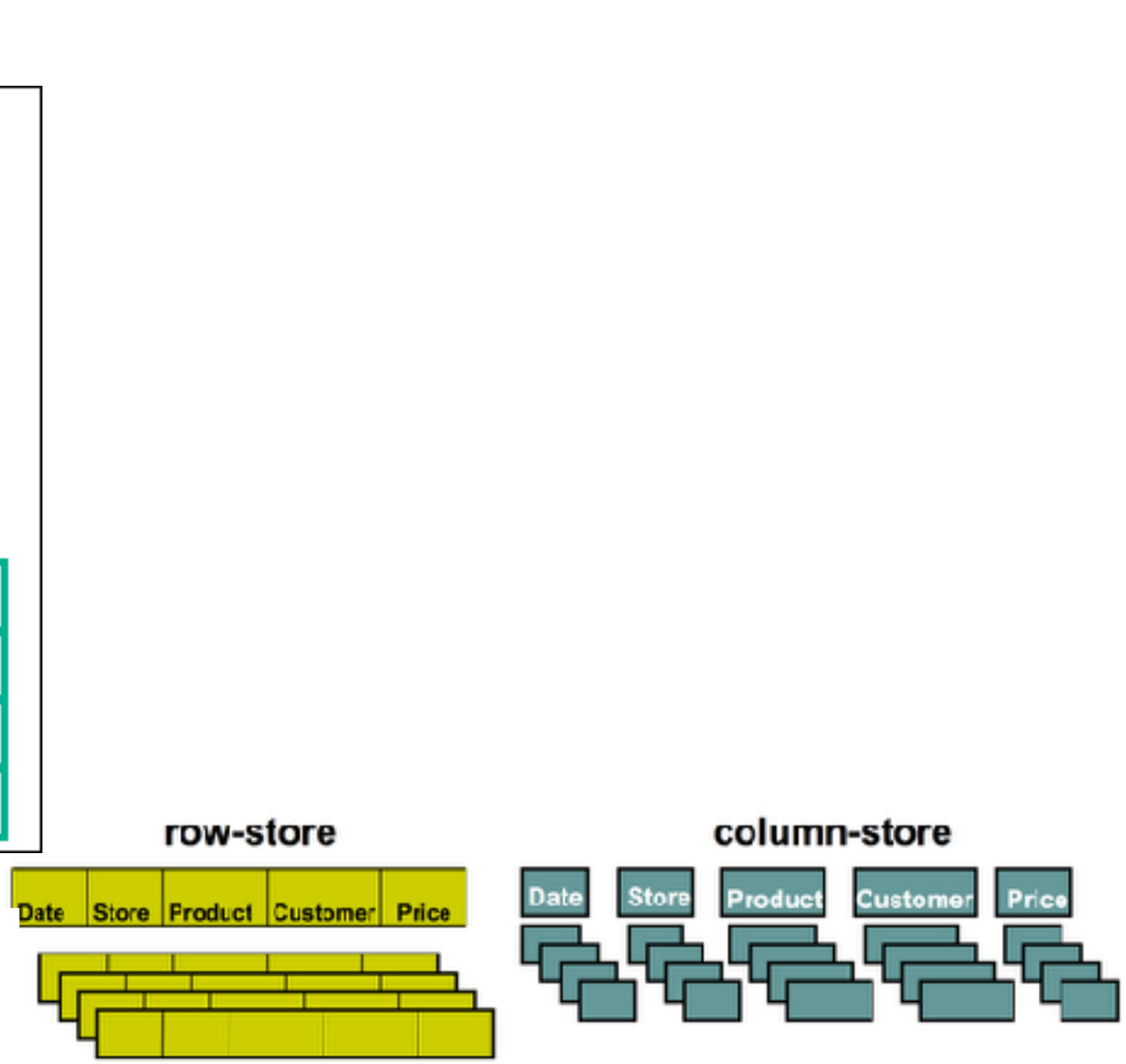
Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003





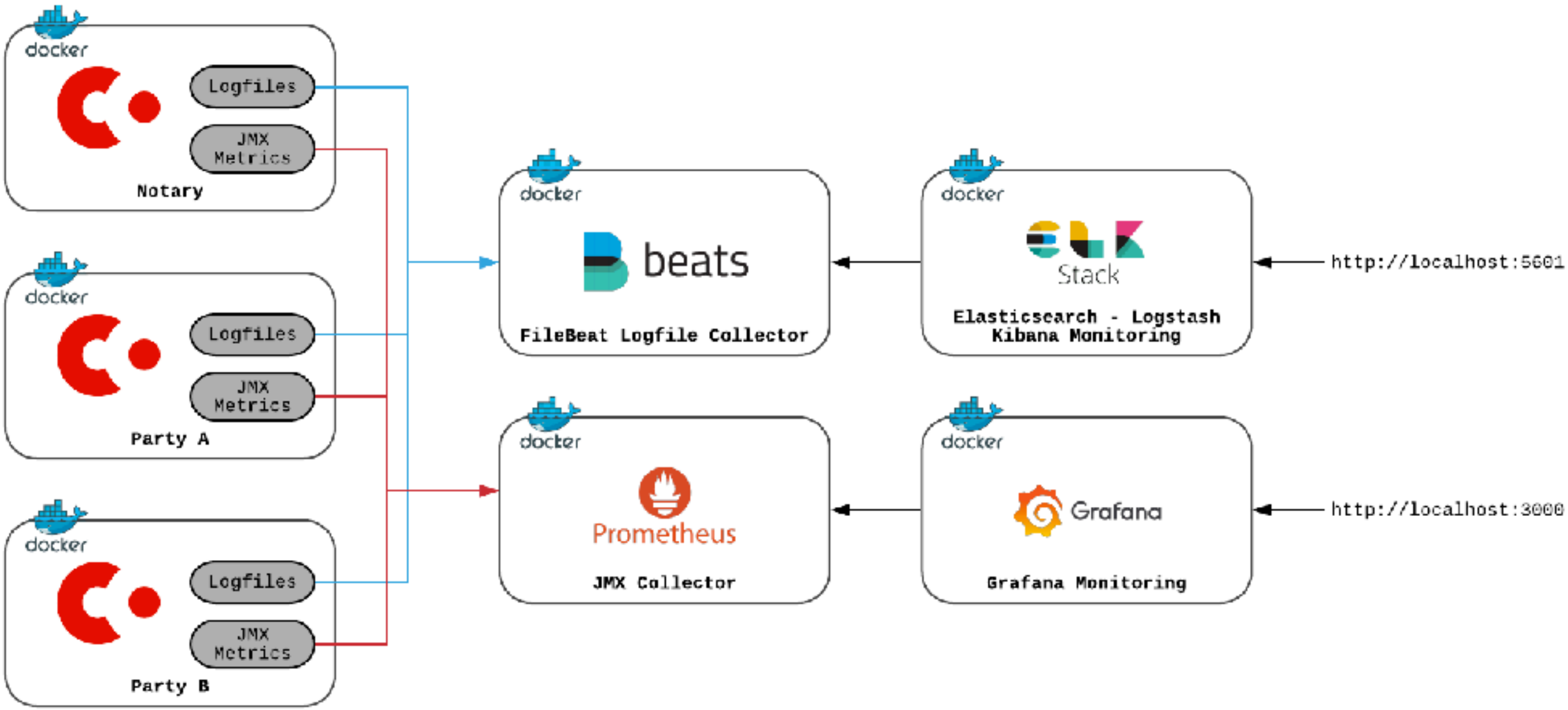
column-family	Columnar
sparse data model	
multi-dimensional map	
column-families are not independently accessible.	every column is stored separately
NoSQL	SQL interface
Reads that use the partition key are incredibly fast	optimized for read-mostly analytical workloads
high throughput writes	very slow writes
	data warehouses

	MongoDB	Cassandra
Expressive object model	Yes	No
Secondary indexes	Yes	No
No downtime on node failure	No	Yes
High write throughput	No	Yes

- Cassandra is optimized for really high throughput on writes. **If your use case is read-heavy (like cache) then Cassandra might not be an ideal choice.**
- It does not support complete transaction management across the tables. Not ACID compliant system.
- Secondary Index not supported. Have to rely on Elastic search /Solr for Secondary index and the custom sync component has to be written.

not possible to aggregate data in a query. Sums and averages like information have to be done by the client-side application.

Use Cassandra as the primary data store for capturing information as it arrives into the system; but then build “query-optimised views” of subsets of that data in other databases specialised to the kinds of access users require. Use an indexing engine such as SOLR to provide full-text search across records, or a graph database such as Neo4j to store metadata in a way that supports “traversal-heavy” queries that join together many different kinds of entity. Use an RDBMS when you need the full power and flexibility of SQL to express ad hoc queries that explore the data in multiple dimensions.


docker
Logfiles
JMX Metrics
Notary
docker
Logfiles
JMX Metrics
Party A
docker
Logfiles
JMX Metrics
Party B
docker
beats
FileBeat Logfile Collector
docker
Prometheus
JMX Collector
docker
Stack
Elasticsearch - Logstash
Kibana Monitoring
http://localhost:5601
docker
Grafana
Grafana Monitoring
http://localhost:3000

Classic Relational Databases

Name	Age	Nickname	Employee
Gianarno Cuttizzoni Founder & CEO	40	Heidi	◉
Marco Botton Tuttofare	38		☑
Mariah Maclachlan Better Half	41	Potato	⊟
Valerie Liberty Head Chef	16	Val	☑

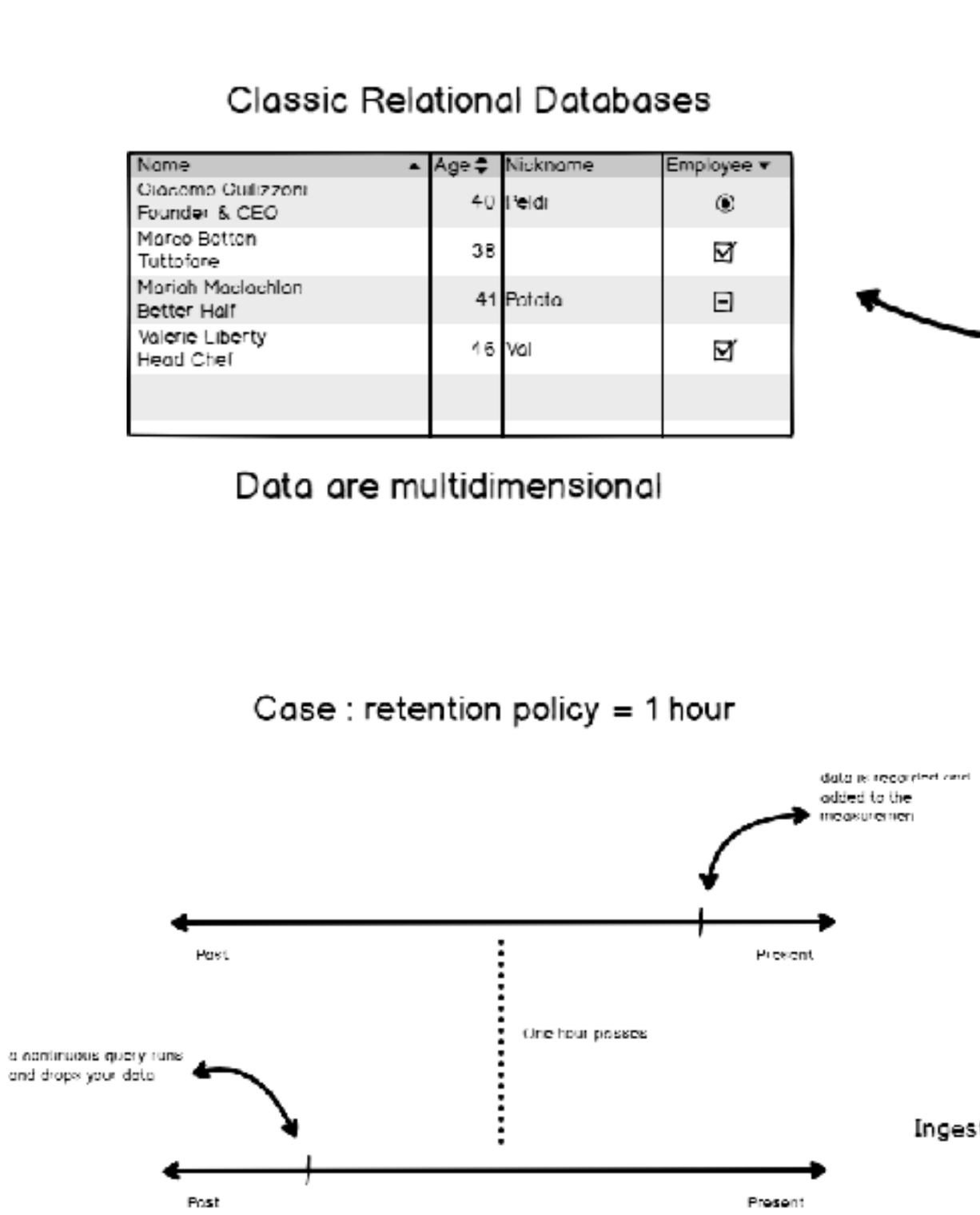
Data are multidimensional

Time Series Databases

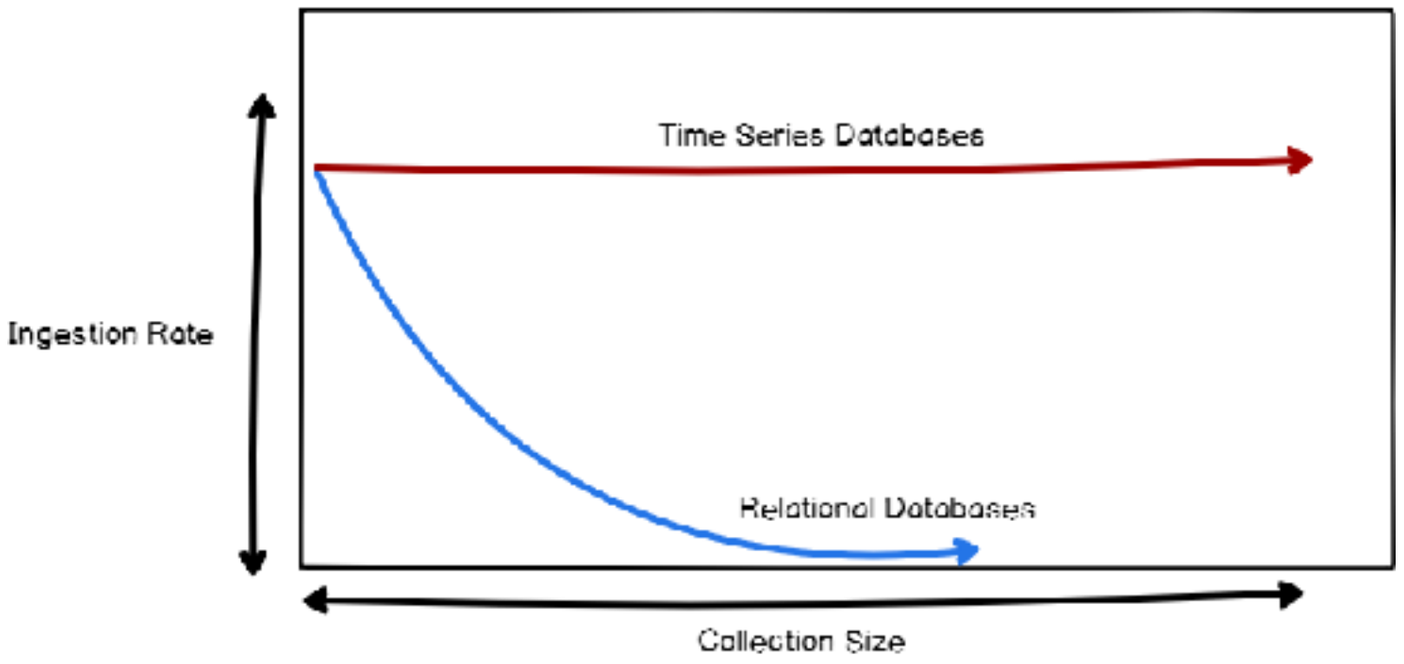
Sensor Temperature	Time
39.6	12/04/19 @ 14:12
41.2	12/04/19 @ 14:13
12.4	14/04/19 @ 12:15
18.5	16/04/19 @ 10:05

Data are aggregated over time

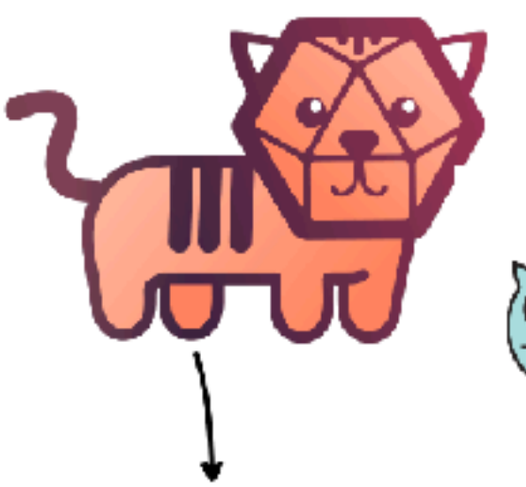
Case : retention policy = 1 hour



DBMS & TSDB Difference

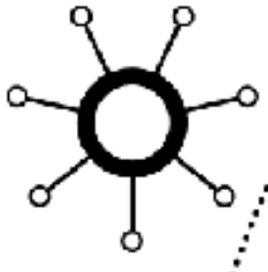


Tools that 'produce' TSDB data



monitoring done via specific query agents

collectd



StatsD

monitoring done via node exporters

Prometheus

Tools that 'consume' TSDB

chronograf

DATADOG

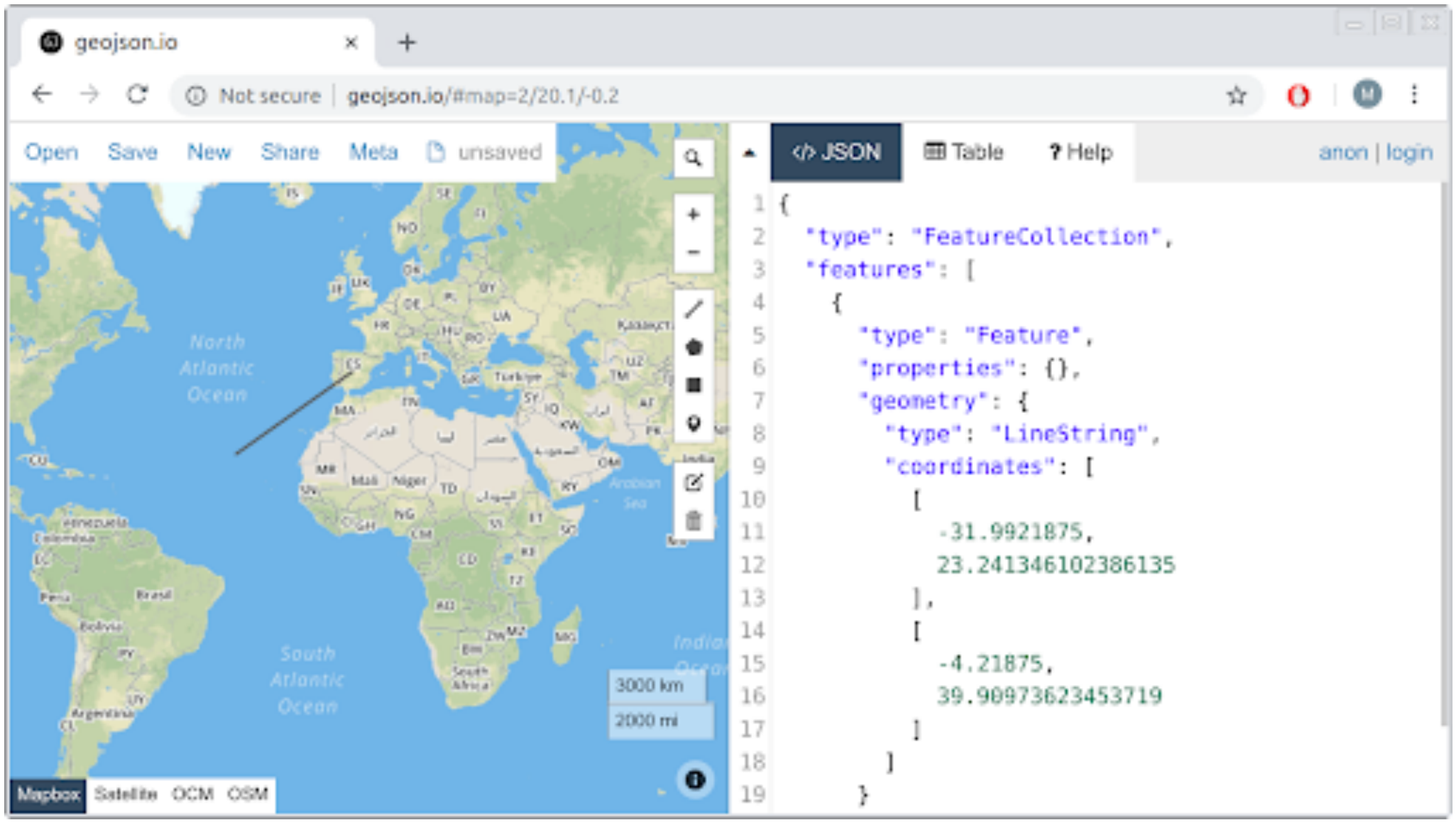
Prometheus

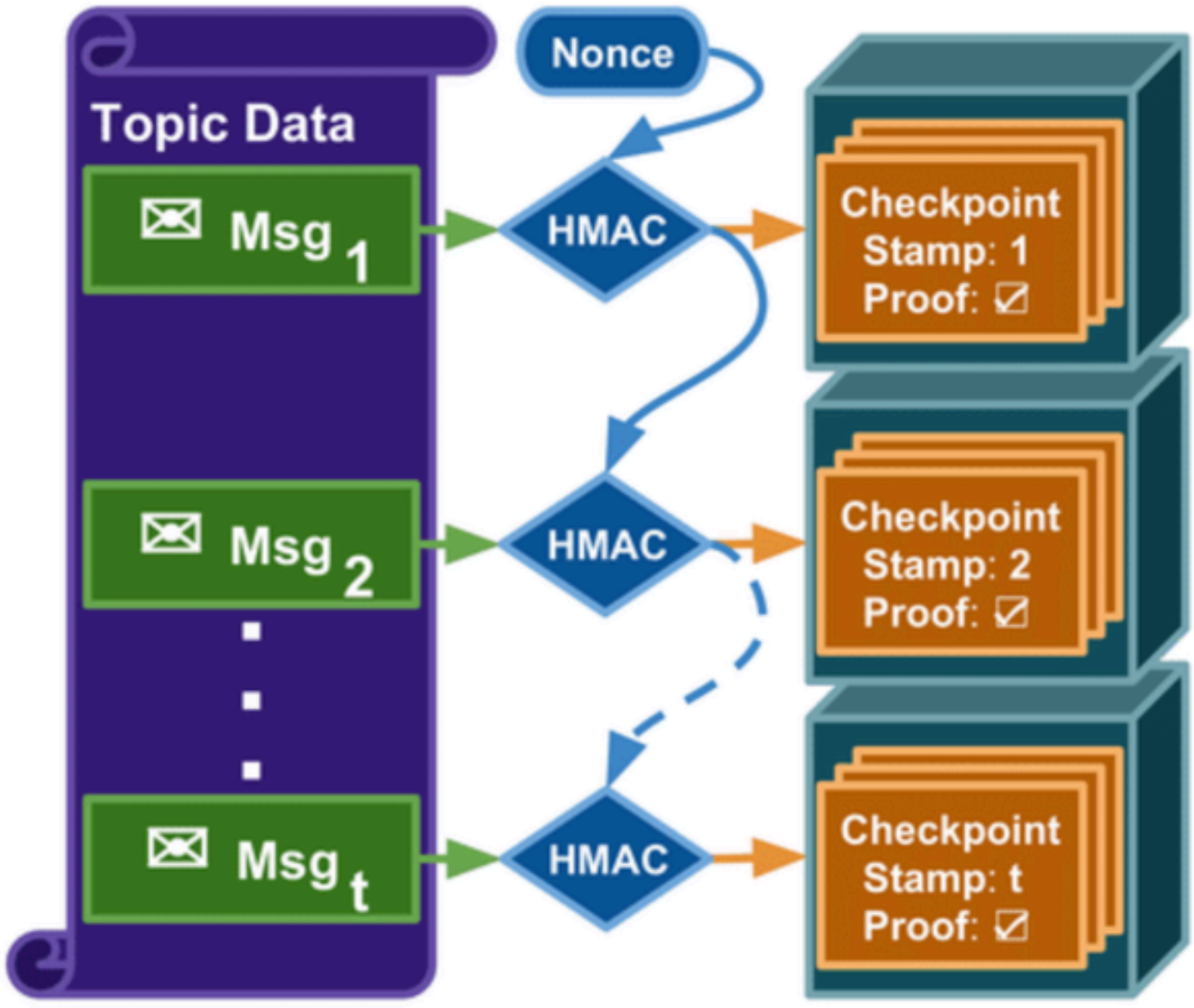
Grafana

kibana

*Non-exhaustive list

Car ID	Departure			Arrival		
	Date-Time	Latitude	Longitude	Date-Time	Latitude	Longitude
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...



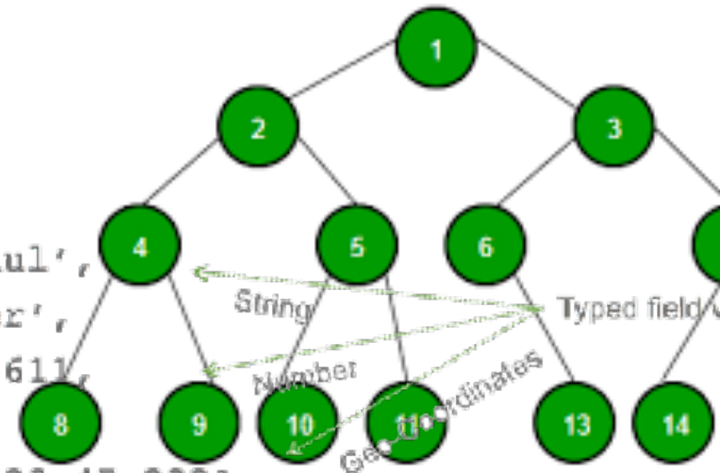

Topic Data
Msg 1
Msg 2
Msg t
Nonce
HMAC
HMAC
HMAC
Checkpoint
Stamp: 1
Proof: ☑
Checkpoint
Stamp: 2
Proof: ☑
Checkpoint
Stamp: t
Proof: ☑

Rdbms (Referential Integrity)

tblPerson			
ID	Name	Email	GenderID
1	Jade	j@j.com	2
2	Mary	m@m.com	3
3	Martin	ma@ma.com	1
4	Rob	r@r.com	NULL
5	May	may@may.com	2
6	Kristy	k@k.com	NULL

tblGender	
ID	Gender
1	Male
2	Female
3	Unknown

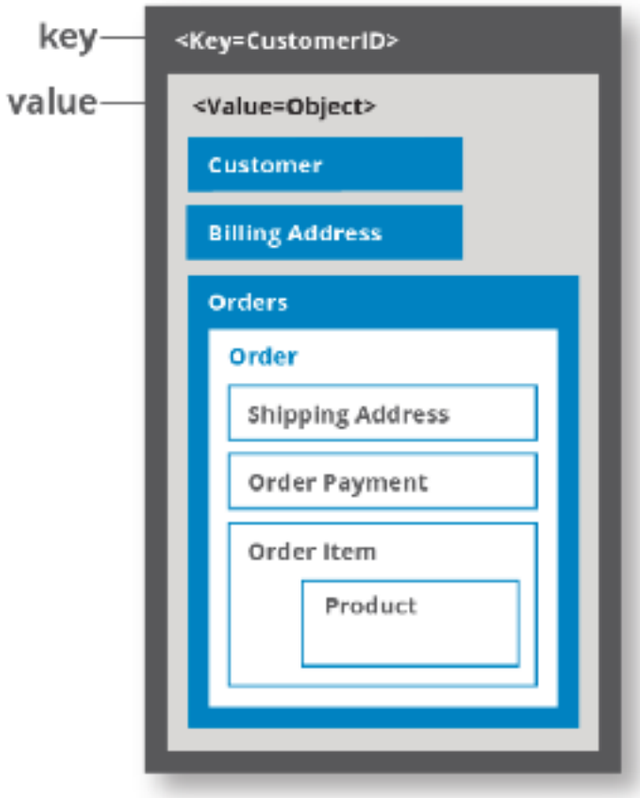
Fields

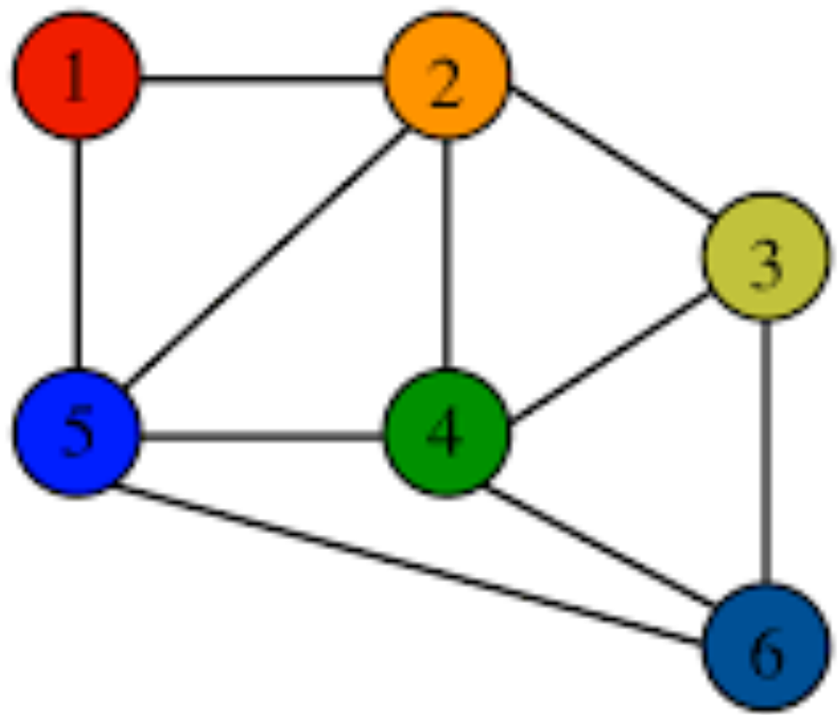



```
{
  first_name: 'Paul',
  surname: 'Miller',
  cell: 447557505611,
  city: 'London',
  location: [45.123,47.232],
  Profession: ['banking', 'finance', 'trader'],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields can contain an array of documents

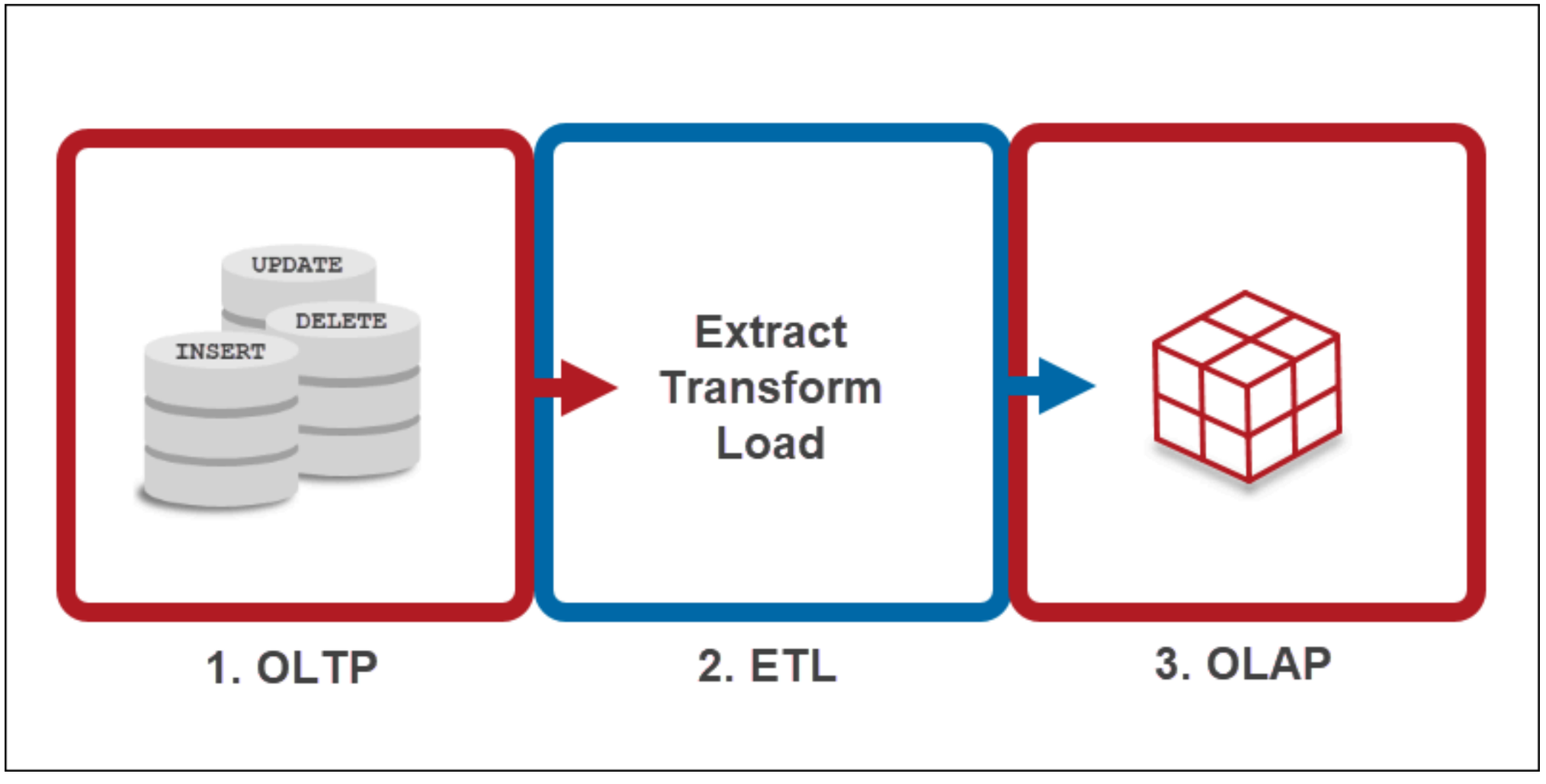
key	value
123	123 Main St.
126	(805) 477-3900





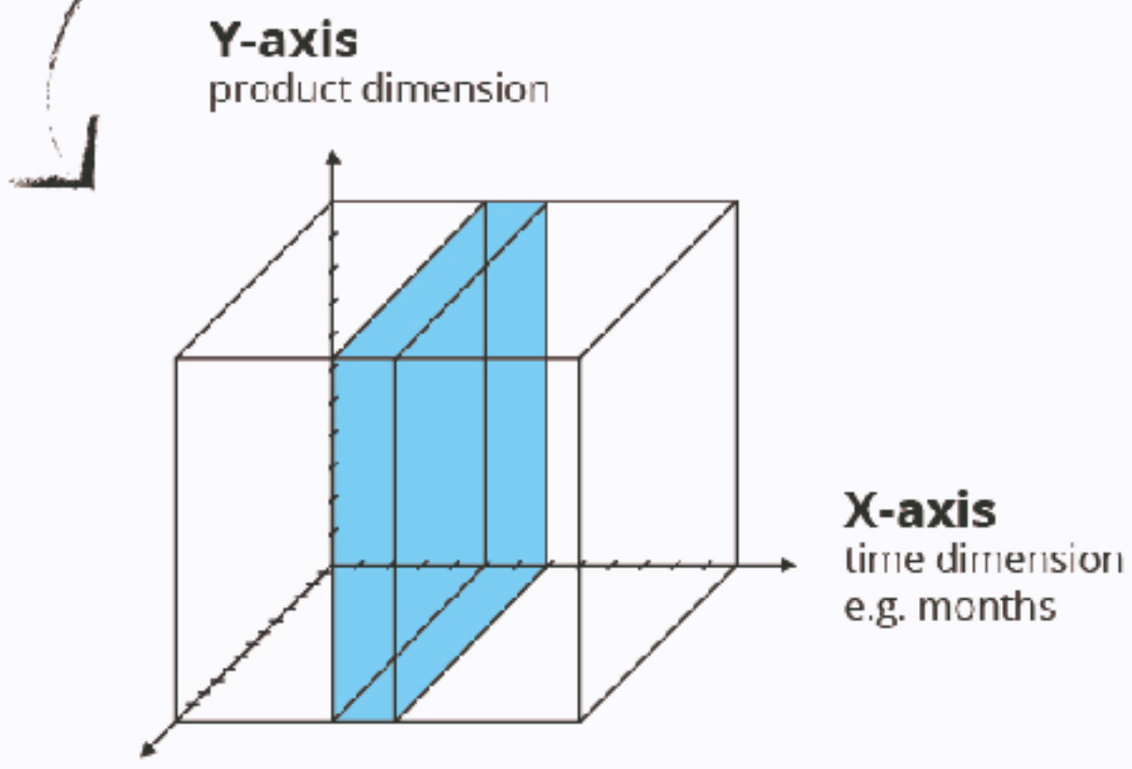
Wide Column

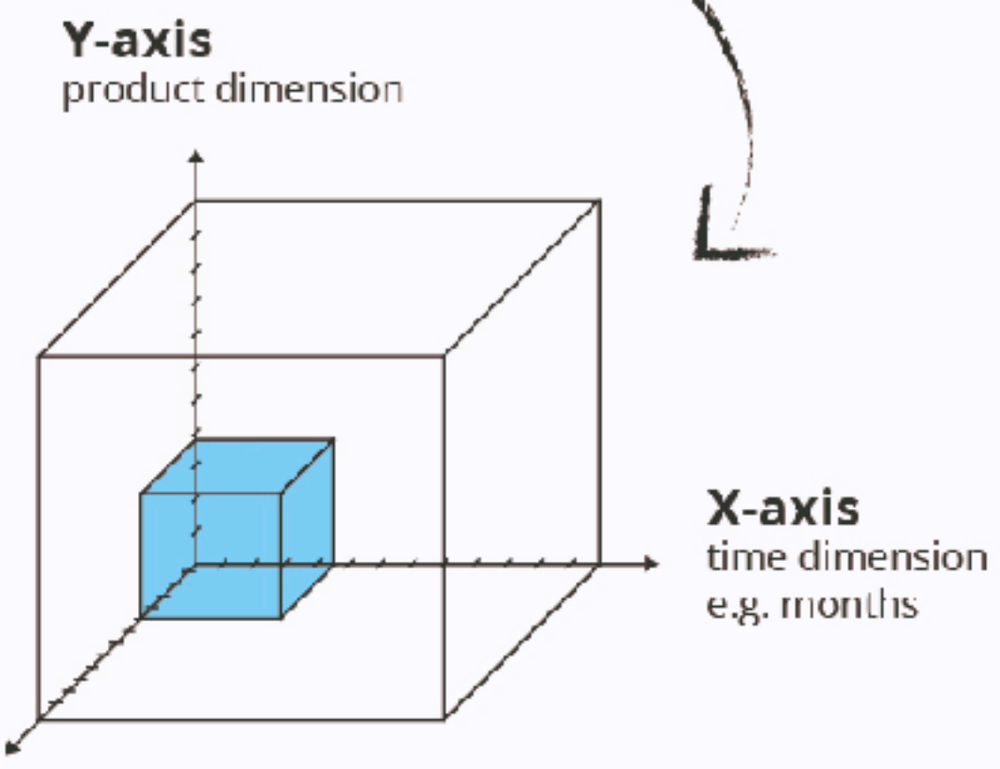
- Spotify uses Cassandra to store user profile attributes and metadata about artists, songs, etc. for better personalization
- Facebook initially built its revamped Messages on top of HBase, but is now also used for other Facebook services like the Nearby Friends feature and search indexing
- Outbrain uses Cassandra to serve over 190 billion personalized content recommendations each month

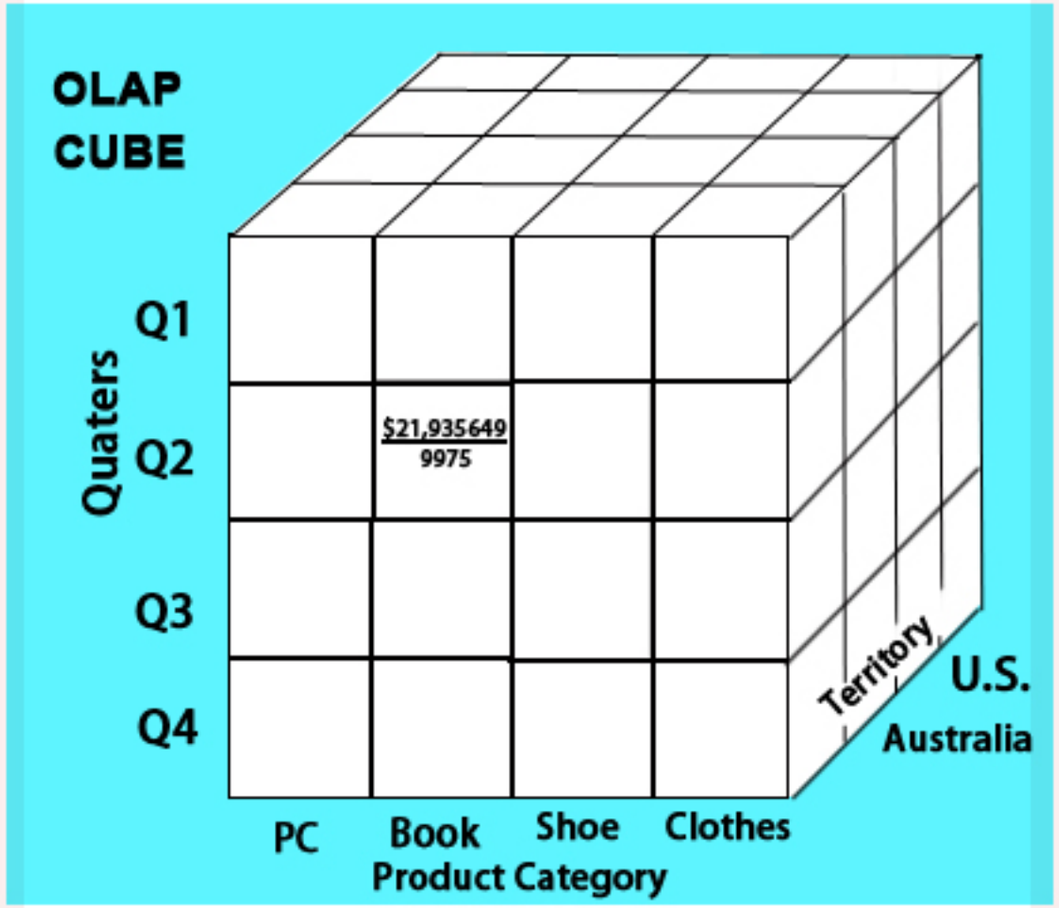

UPDATE
DELETE
INSERT
Extract
Transform
Load
1. OLTP
2. ETL
3. OLAP

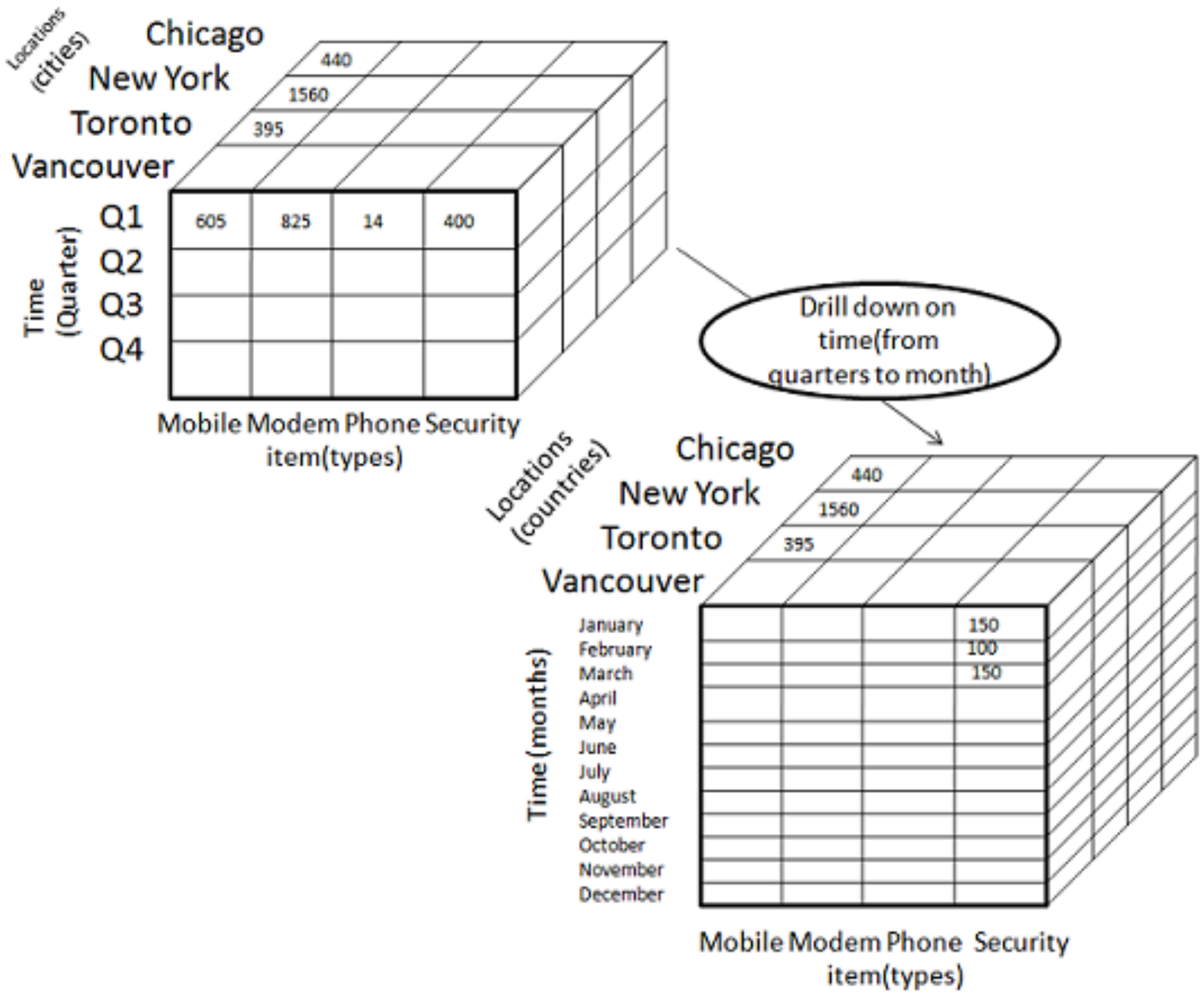
Slicing & Dicing

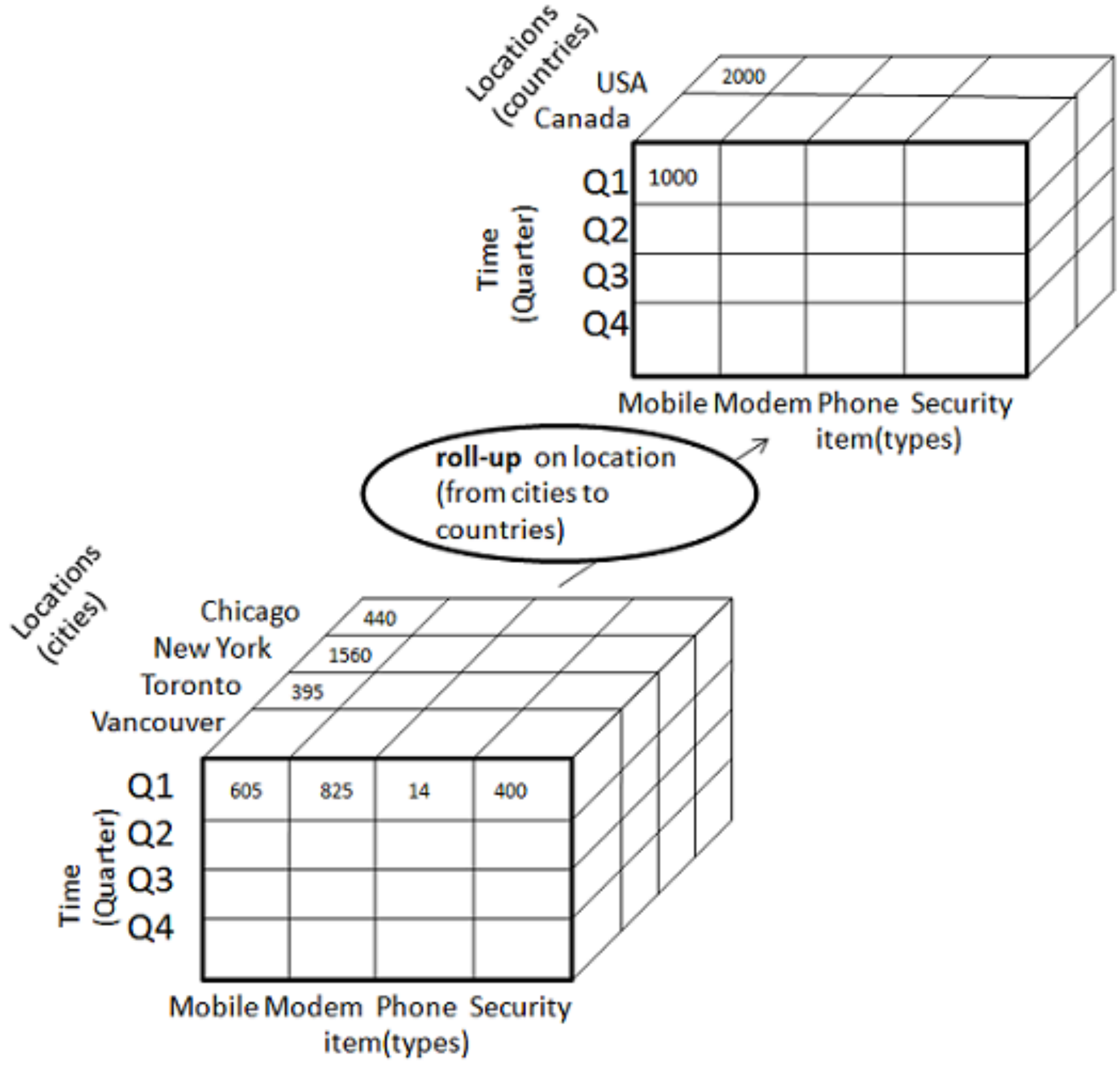
OLAP cube operations

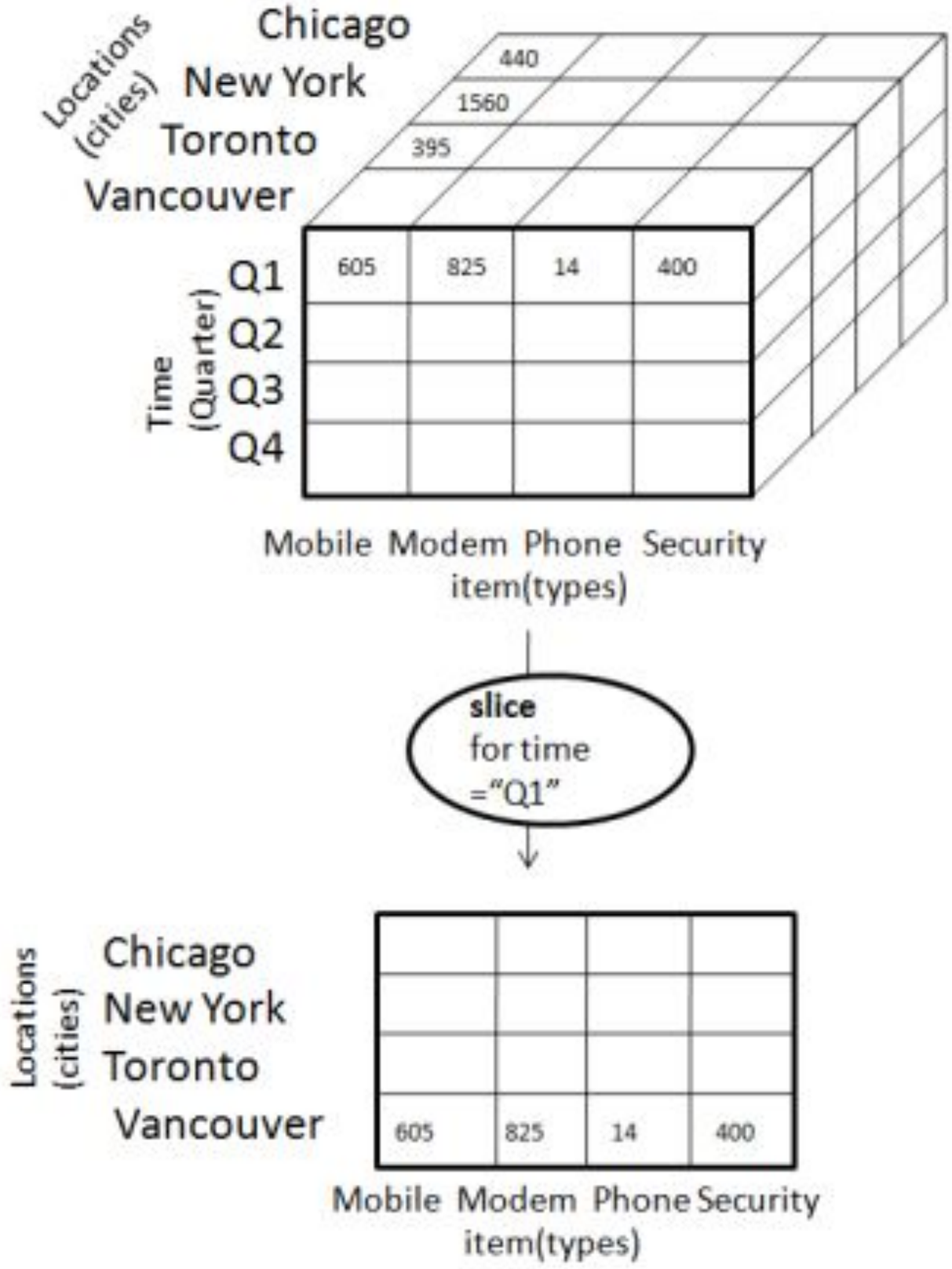


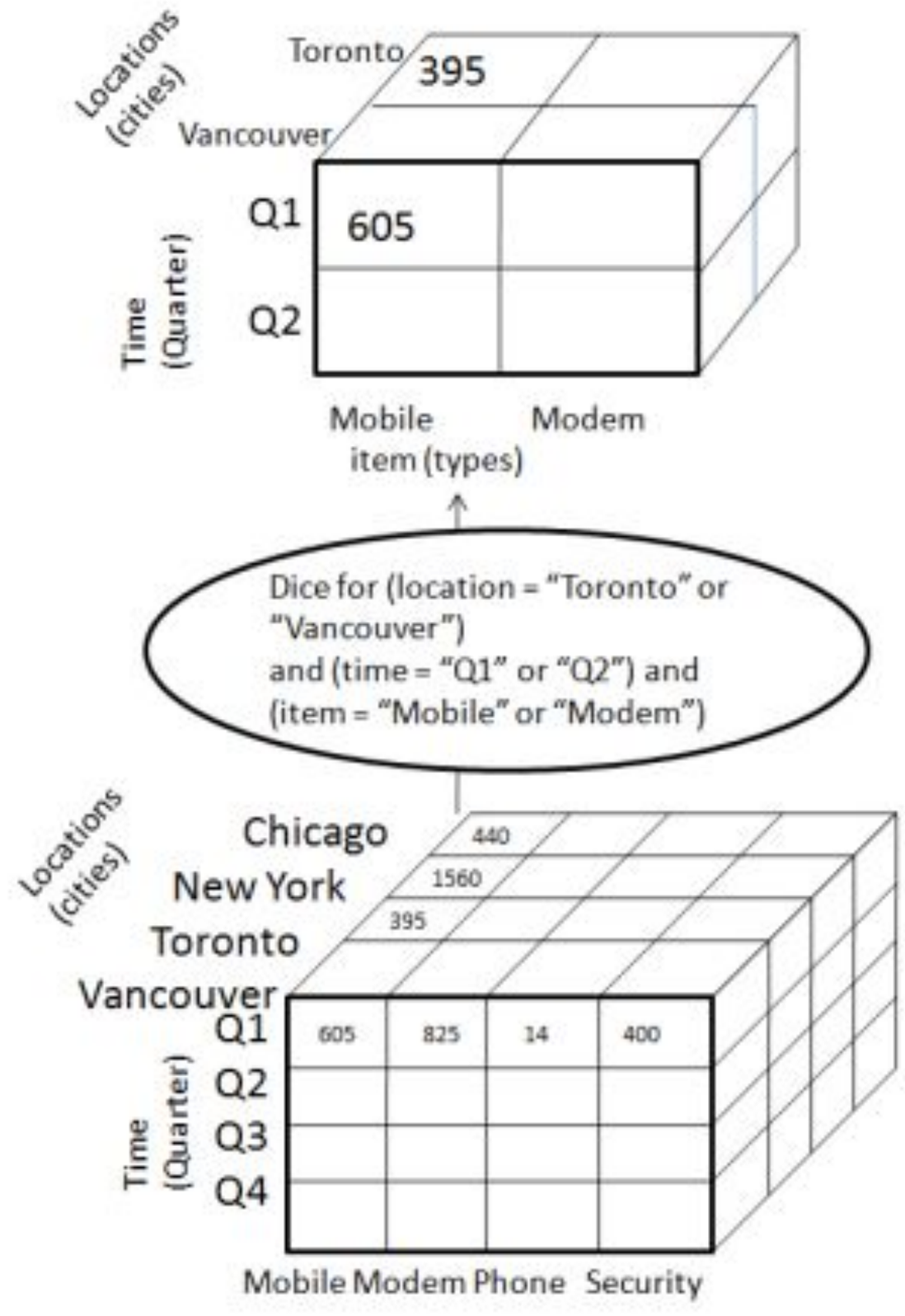


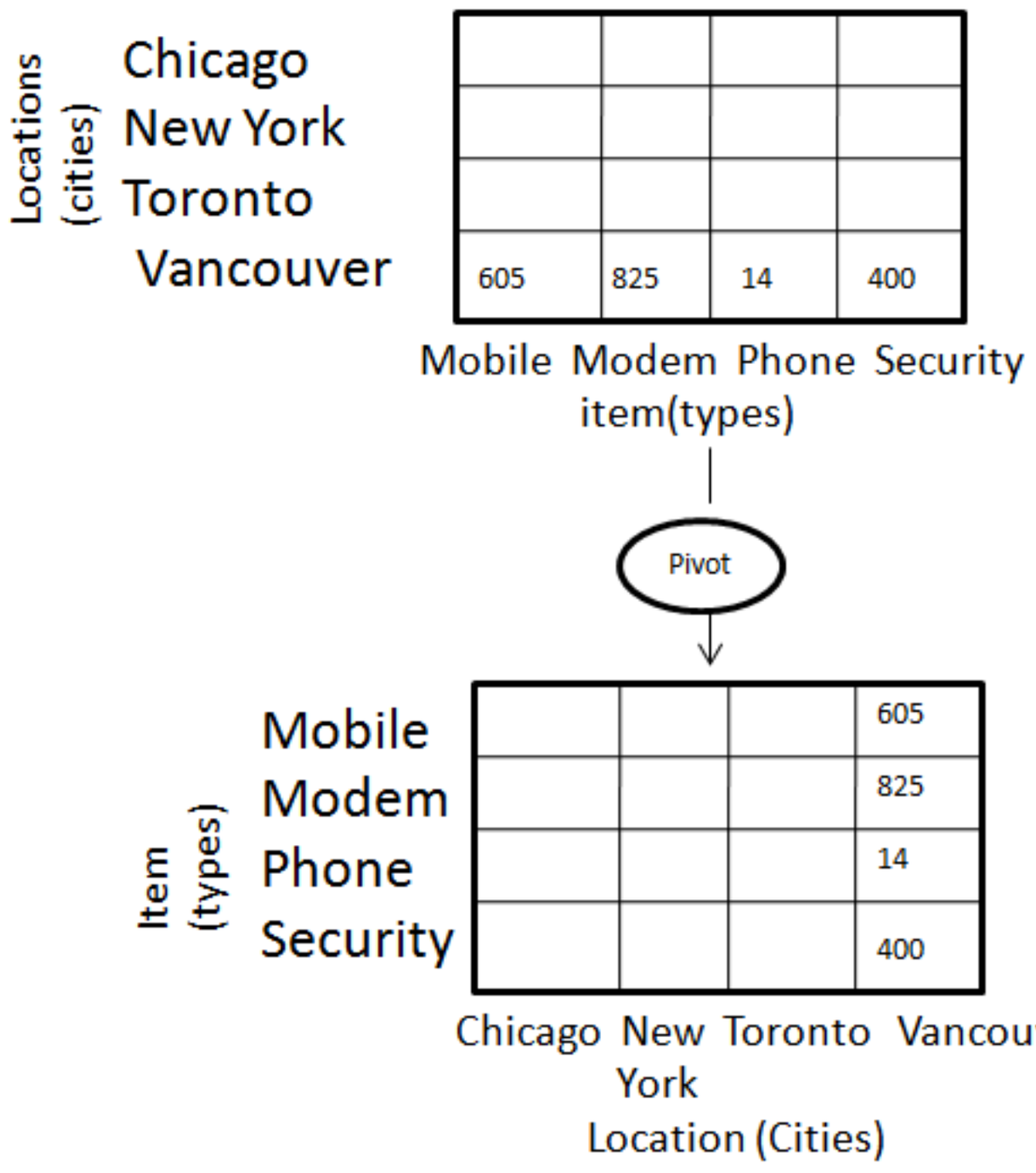

OLAP
CUBE
Quaters
Q1
Q2
Q3
Q4
$21,935649
9975
PC
Book
Shoe
Clothes
Product Category
Territory
U.S.
Australia


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile Modem Phone Security
item(types)
Drill down on time(from quarters to month)
Locations (countries)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (months)
January
February
March
April
May
June
July
August
September
October
November
December
150
100
150
Mobile Modem Phone Security
item(types)

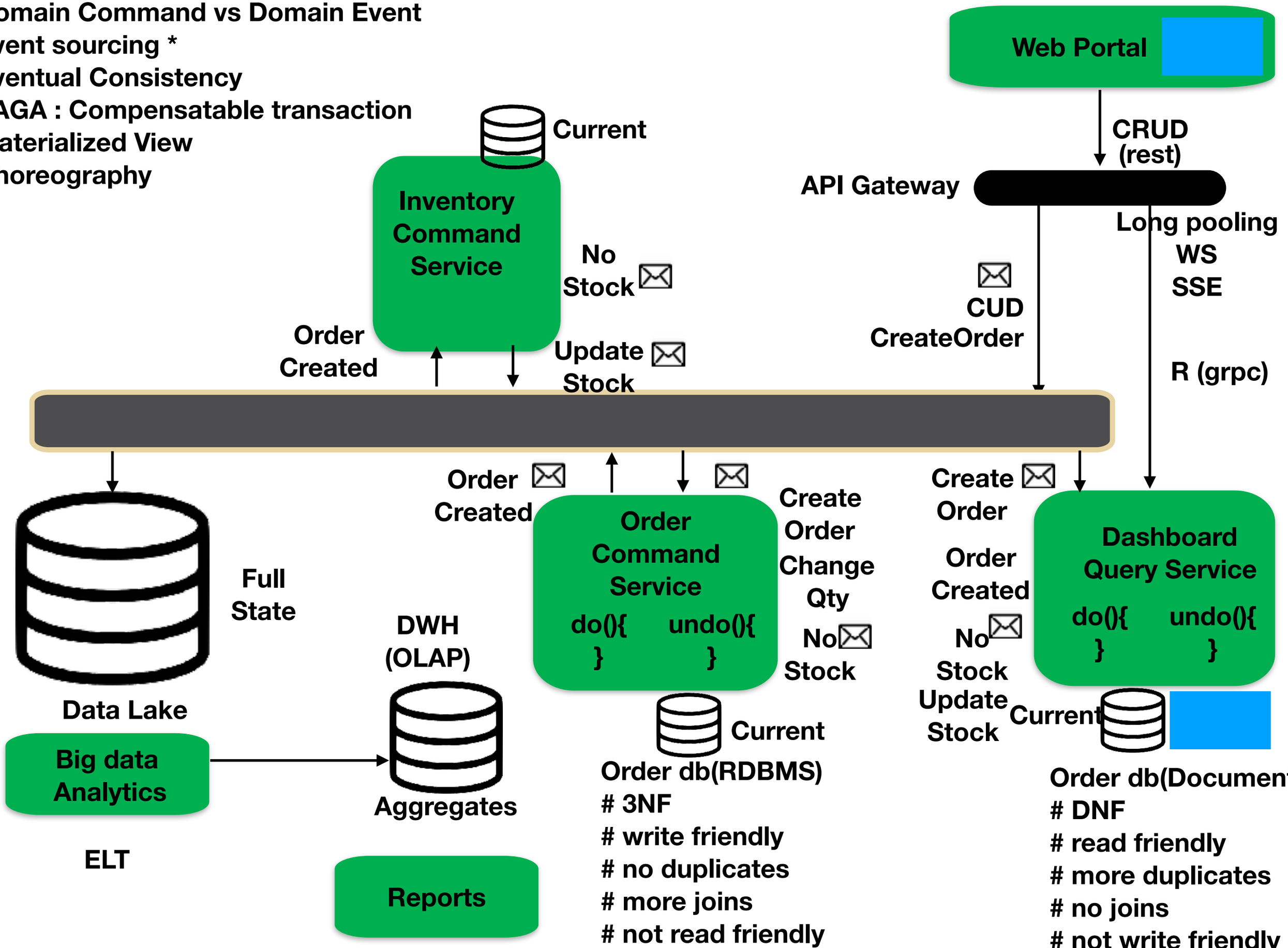

Locations (countries)
USA
Canada
2000
1000
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)
roll-up on location (from cities to countries)
Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
605
825
14
400
Q1
Q2
Q3
Q4
Time (Quarter)
Mobile Modem Phone Security
item(types)


Locations (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
Time (Quarter)
Q1
Q2
Q3
Q4
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
slice
for time
="Q1"
Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)


Locations (cities)
Toronto
395
Vancouver
Q1
605
Q2
Time (Quarter)
Mobile
Modem
item (types)
Dice for (location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "Mobile" or "Modem")
Locations (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
Q1
605
825
14
400
Q2
Q3
Q4
Time (Quarter)
Mobile
Modem
Phone
Security

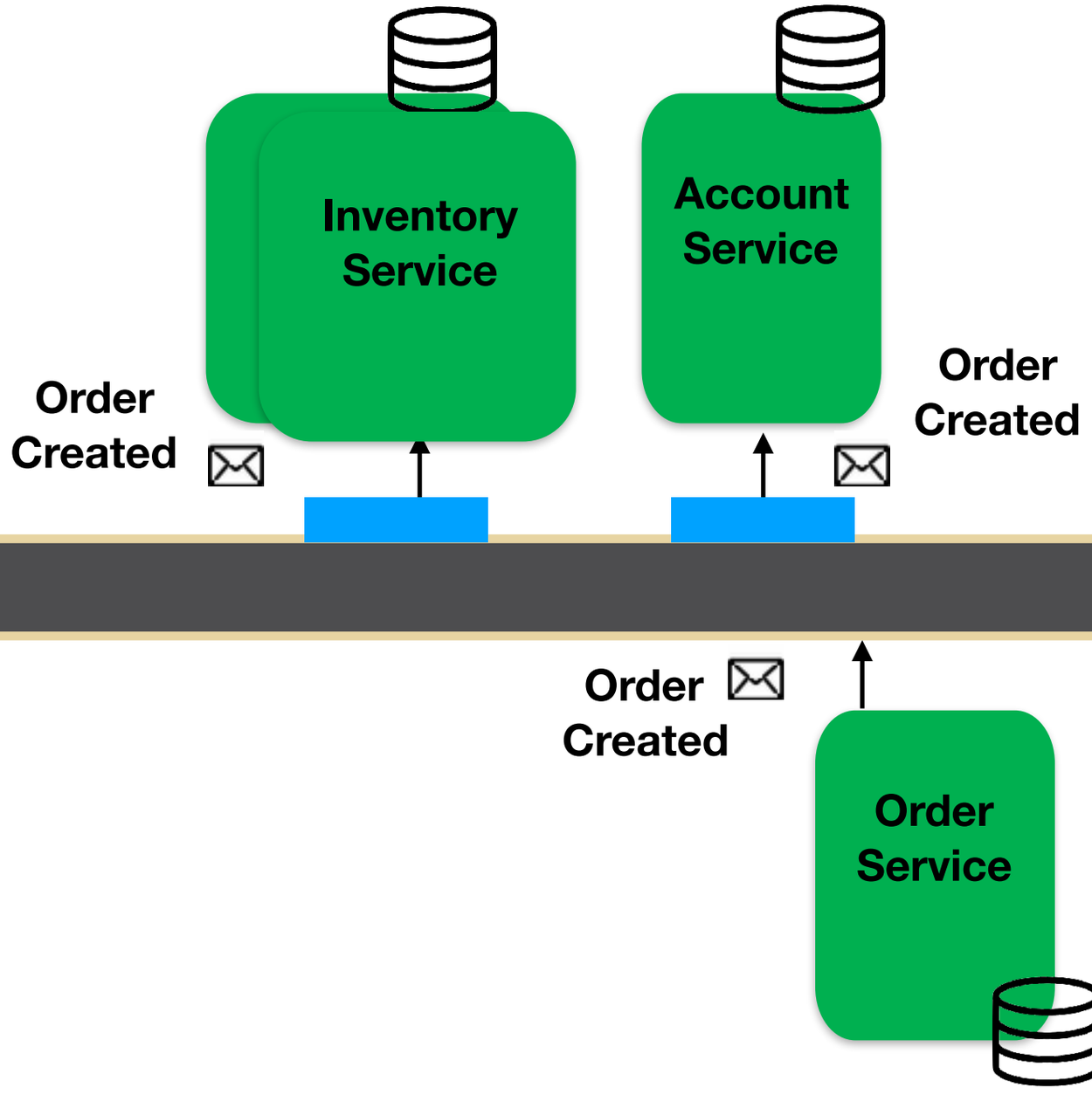

Locations (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
Mobile
Modem
Phone
Security
item(types)
Pivot
Item (types)
Mobile
Modem
Phone
Security
605
825
14
400
Chicago
New York
Toronto
Vancouver
Location (Cities)

CQRS (Command Query Responsibility)
Domain Command vs Domain Event
Event sourcing *
Eventual Consistency
SAGA : Compensatable transaction
Materialized View
Choreography



Queue

Topic


Inventory Service
Account Service
Order Created
Order Created
Order Created
Order Service

Part IV

Addressing Performance Scenarios
Performance Engineering
CPU Performance patterns
I/O Performance patterns
Memory performance patterns
Performance anti patterns
Lab : Building Architecture for a Desktop Application

Commons

Circuit Breaker

Log Library

Cache Library (config)

Exception Handling Library

Web Socket Server

Queue

Control Facade

Config Proxy

Rest APi Client

Cloud Storage Proxy

Queue

Cloud Storage Egress

Web Socket Client

Data Access Layer

Temp Local Storage

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithm

Domain

Performance View

Engineering for Performance

Performance Objectives
(Response Time, Throughput, Resource Utilization, Workload)

Performance Modeling
(Scenarios, Objectives, Workloads, Requirements, Budgets, Metrics)

Architecture and Design Guidelines
(Principles, Practices and Patterns)

Performance and Scalability Frame

Coupling and Cohesion	Resource Management
Communication	Caching, State Management
Concurrency	Data Structures / Algorithms

Measuring, Testing, Tuning

Measuring	**Testing**	**Tuning**
Response Time	Load Testing	Network
Throughput	Stress Testing	System
Resource Utilization	Capacity Testing	Platform
Workload		Application

Roles
(Architects, Developers, Testers, Administrators)

Life Cycle
(Requirements, Design, Develop, Test, Deploy, Maintain)

Performance Modeling

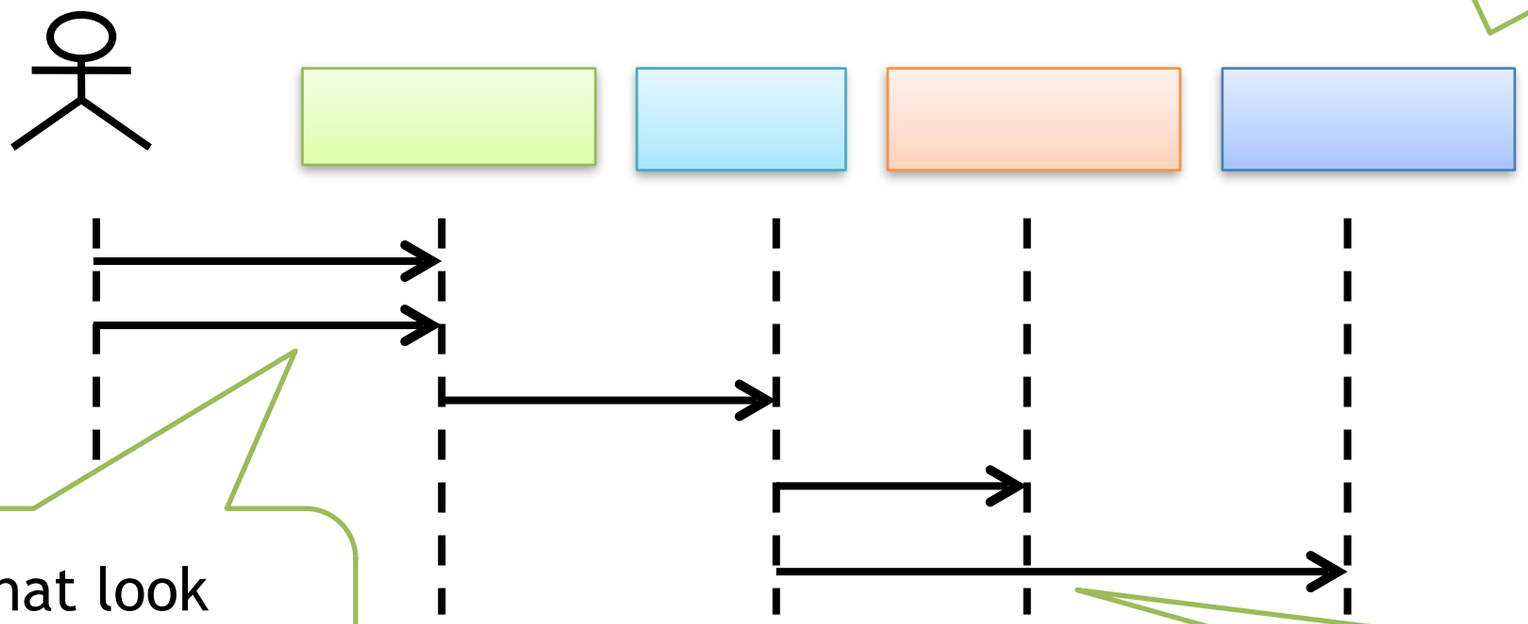
1. Identify Key Scenarios
2. Identify Workloads
3. Identify performance Objectives
4. Identify Budget

5. Identify Processing Steps
6. Allocate Budget
7. Evaluate
8. Validate

Iterate

Allocate Budget

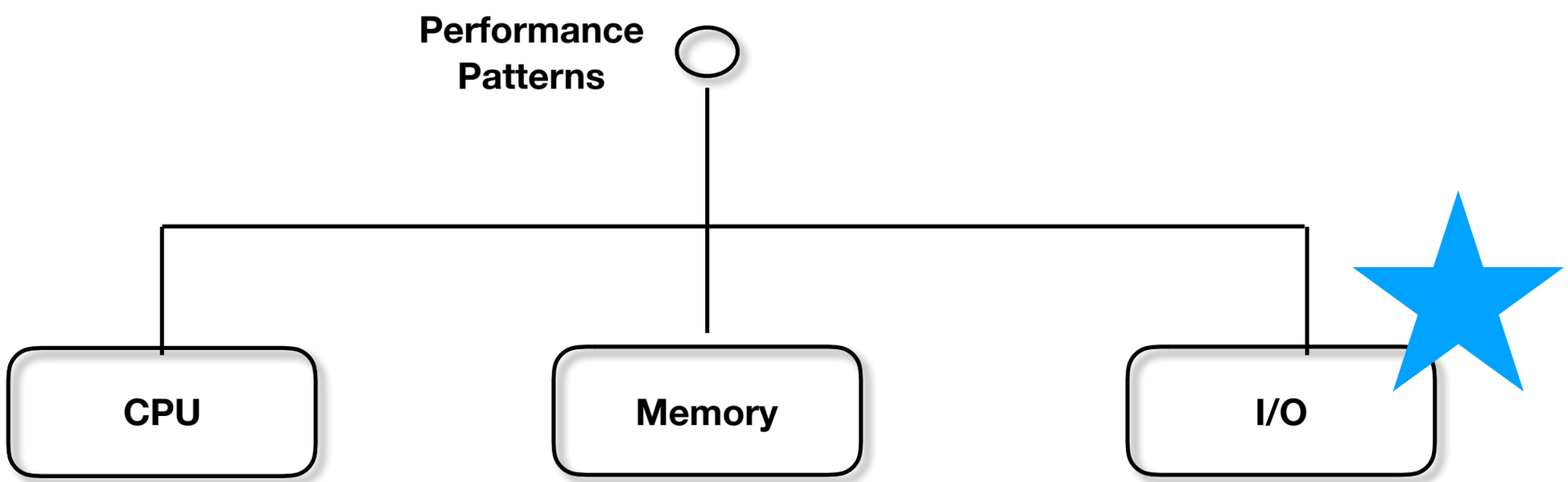
Know the cost of your materials.



For the ones that look risky, conduct some experiments (prototypes)

if you do not know how much time to assign, simply divide the total time equally between the steps.

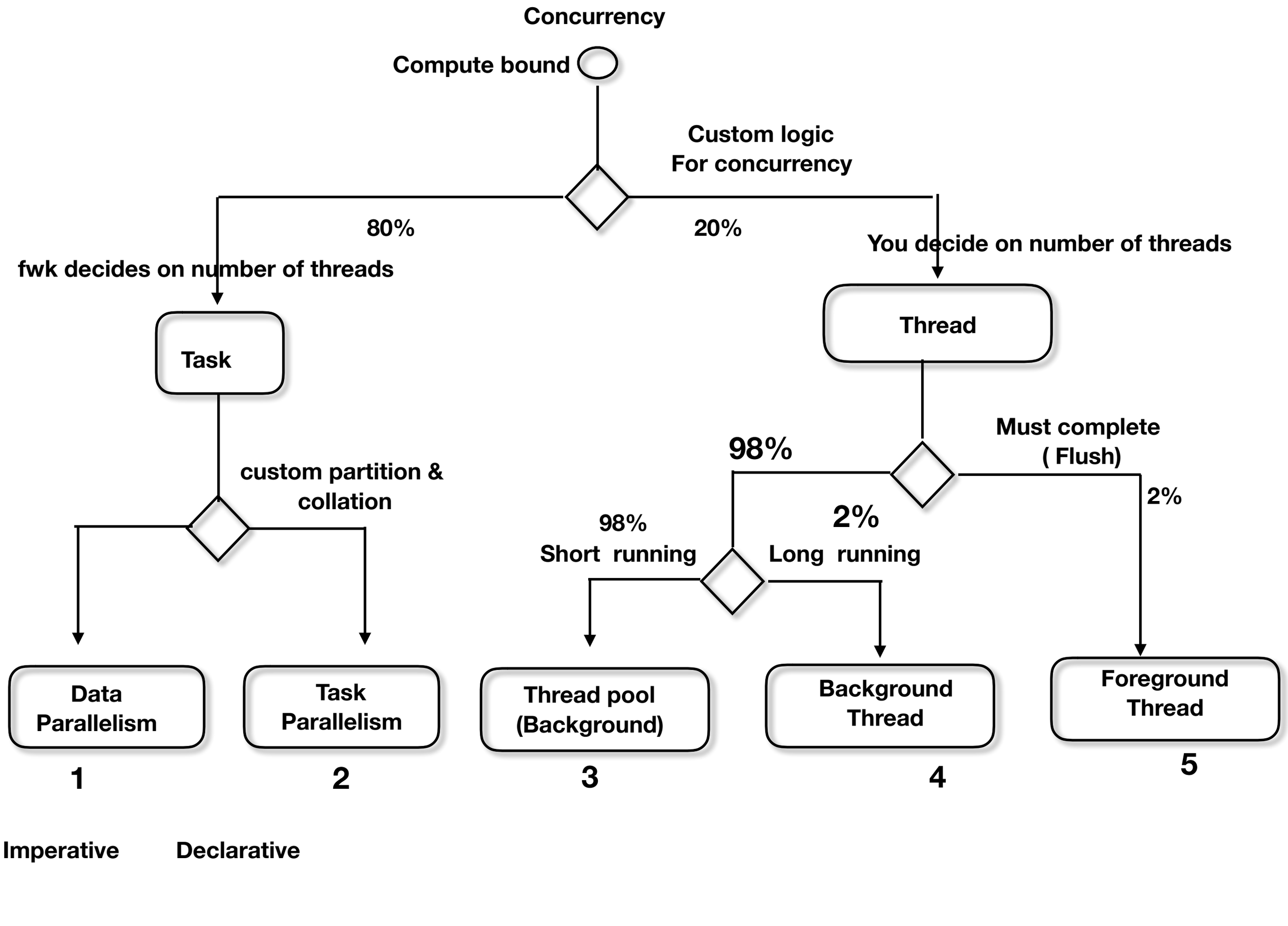
Spread your budget across your processing steps

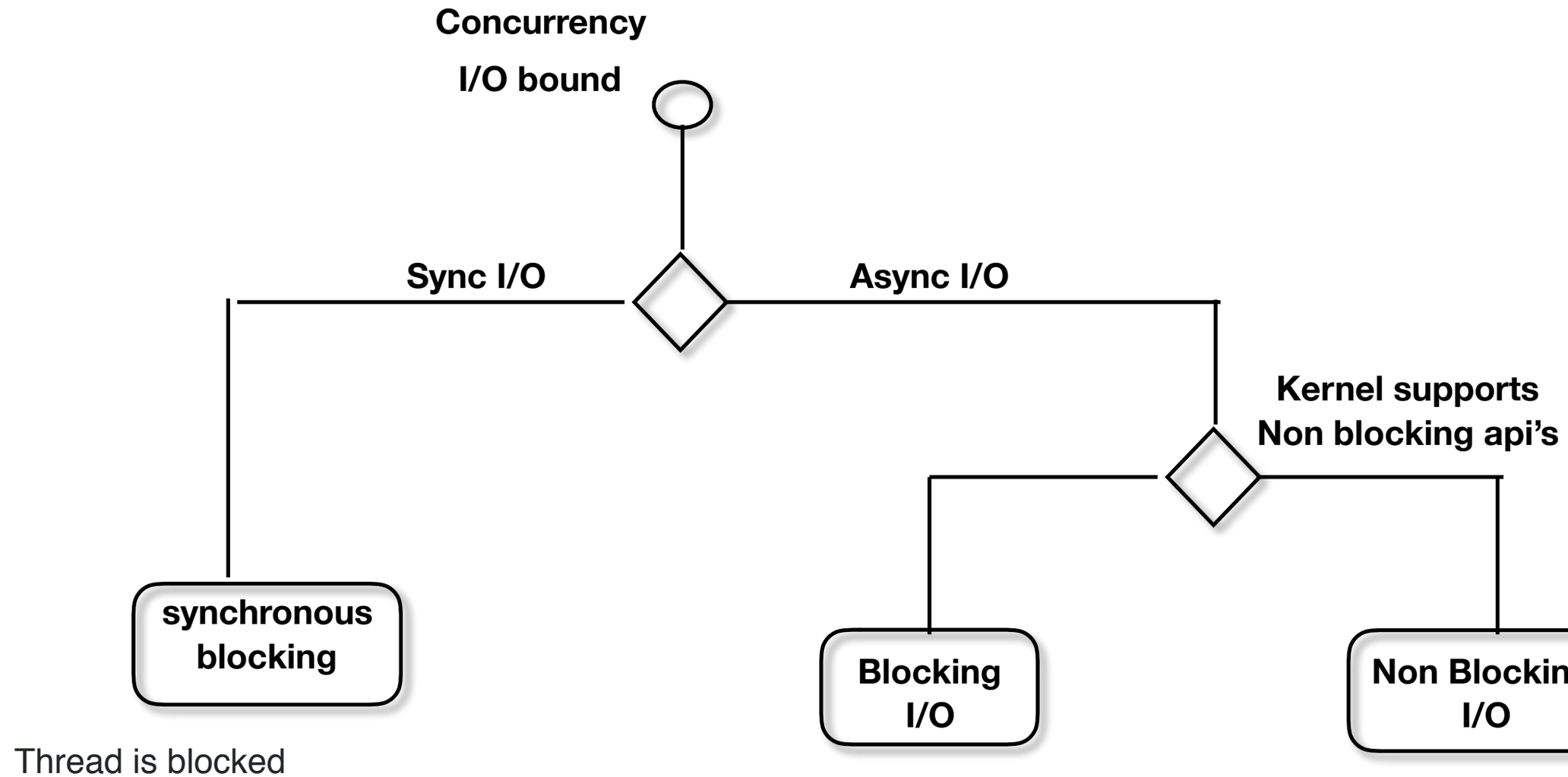


Eager Loading
Caching (Data, Output, Code)
Concurrency
Pooling (Object, Memory)
Queuing (Load Leveling)

Lazy Loading
GC
Fixed Size Memory Pool
Virtualization (Data, UI)
Singleton
Fly weight (singleton for a given state)—> object cache

Chunky / Batch
Compression/ Minify/ ..
Async I/O
Pooling (Connection)
Acquire Late and Release Early

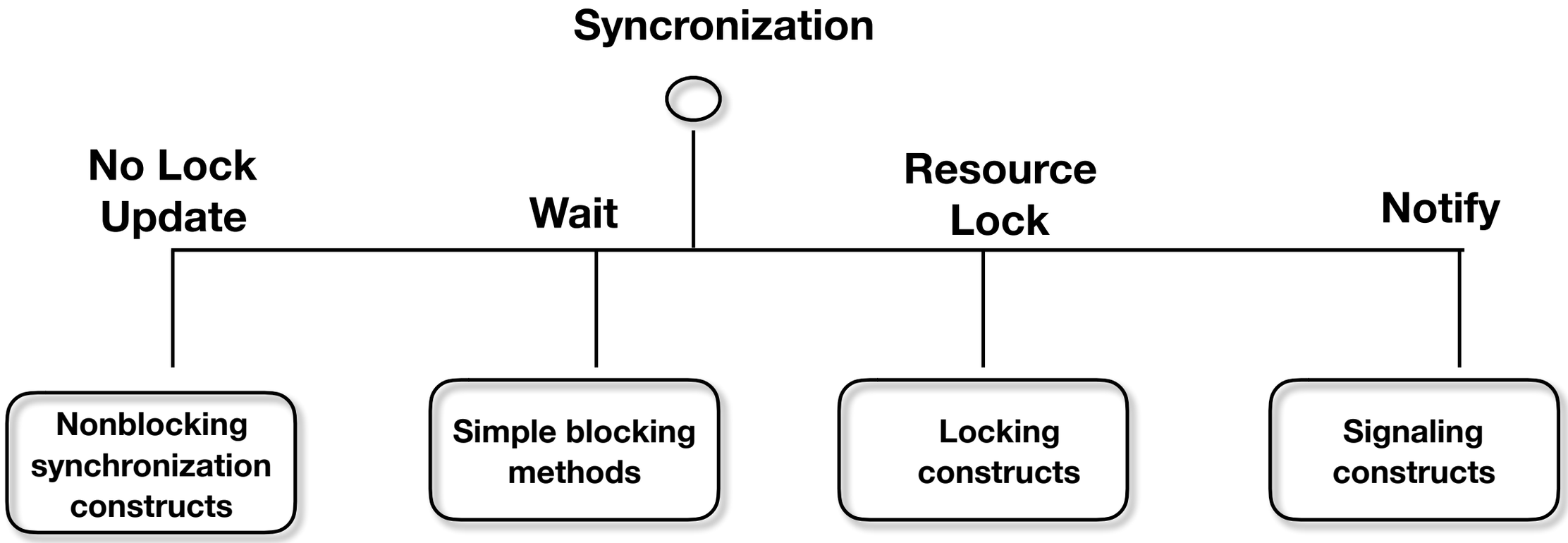

Concurrency
Compute bound
Custom logic
For concurrency
80%
20%
fwk decides on number of threads
You decide on number of threads
Task
Thread
custom partition &
collation
98%
Must complete
(Flush)
2%
98%
Short running
2%
Long running
Data
Parallelism
Task
Parallelism
Thread pool
(Background)
Background
Thread
Foreground
Thread
1
2
3
4
5
Imperative
Declarative

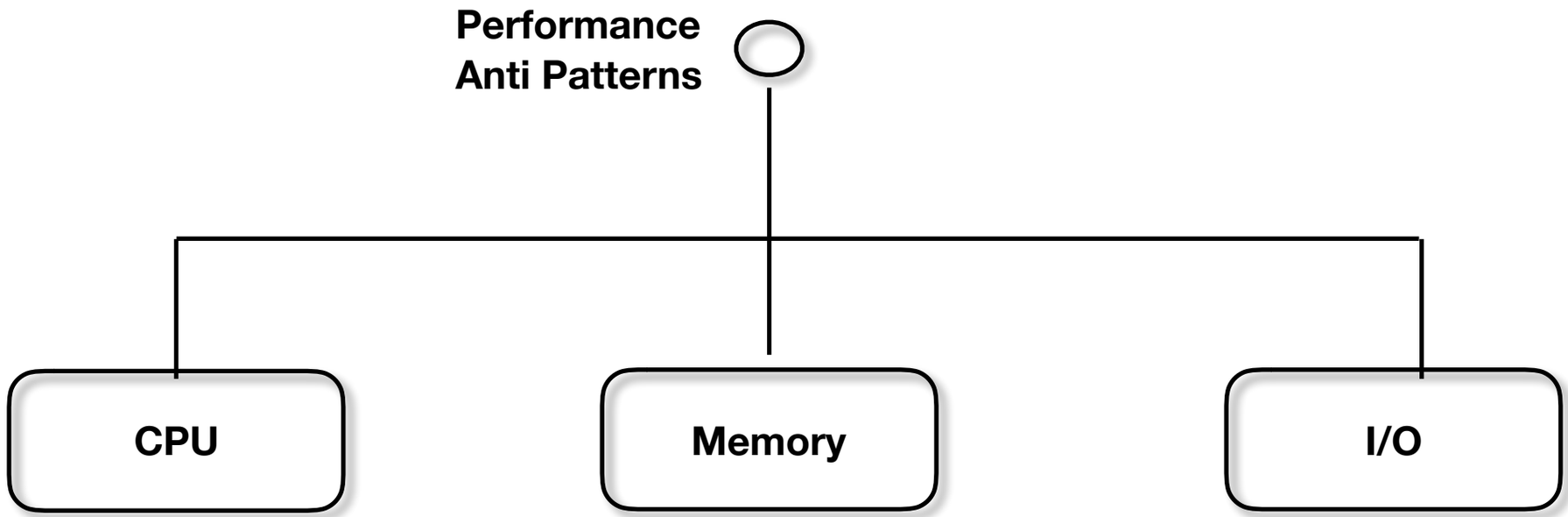


Thread is blocked while hardware performs the work.

Thread is not blocked, But there is still a thread lower down the stack which is blocked on the IO

The kernel doesn't block your thread and returns from the IO call immediately.


Syncronization
No Lock Update
Wait
Resource Lock
Notify
Nonblocking synchronization constructs
Simple blocking methods
Locking constructs
Signaling constructs
– volatile
– Atomic.
– Sleep (bad)
– Join
– Spin
– Sleep (bad)
– Suspend
– Abort
– SetPriority
– Resume
– Exclusive locking
– nonexclusive locking
• Semaphore
• reader/writer locks.
– event wait
– Latch
– Barrier



Single threaded

Lazy Loading
Excessive dynamic allocation
GOF Singleton pattern

Chatty
Sync I/O
Extraneous Fetching (fetching more)
Not Pooling shared resource
Acquire early and Release Late

Profilers
-> CPU
-> Memory
->Thread

Part V

Addressing Cross cutting concerns -
logging
exception handling
Addressing Operational concerns -
Health monitoring Patterns
Alerts

- Context View (black box)
- Functional View
- Quality View
- Constraints
- Assumption
- Logical View (white box - components)
- Infrastructure view (physical box)
- Deployment View (components on physical box)
- Data View (optional)
- Security View (optional)
- Operational View (Monitoring logs, Alert, …)

- Context View (black box)
- Functional View
- Quality View
- Constraints
- Assumption (optional)
- Logical View (white box - components)
- Infrastructure view + Deployment View (components on physical box)
- Operational View (Monitoring logs, Alert)
-

Network.Commons

Queue Receiver <<Adapter>>

Queue Publisher <<Adapter>>

RestClient <<Adapter>>

Circuit Breaker

CrossCutting.Commons

Log Library

Cache Library (config)

Exception Handling Library

Data Collection Service

<<write>>

Distributed Queue

Defect Detection Service

<<read>>

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Config Service

Distributed Queue

<<read>>

Materialized View Service

<<CRUD>>

<<read>>

UI Portal

UI Portal

API App

Report Generation App

Quality Control user

?

http: Rest

Getconfig

?

Quality Check results

Admin

?

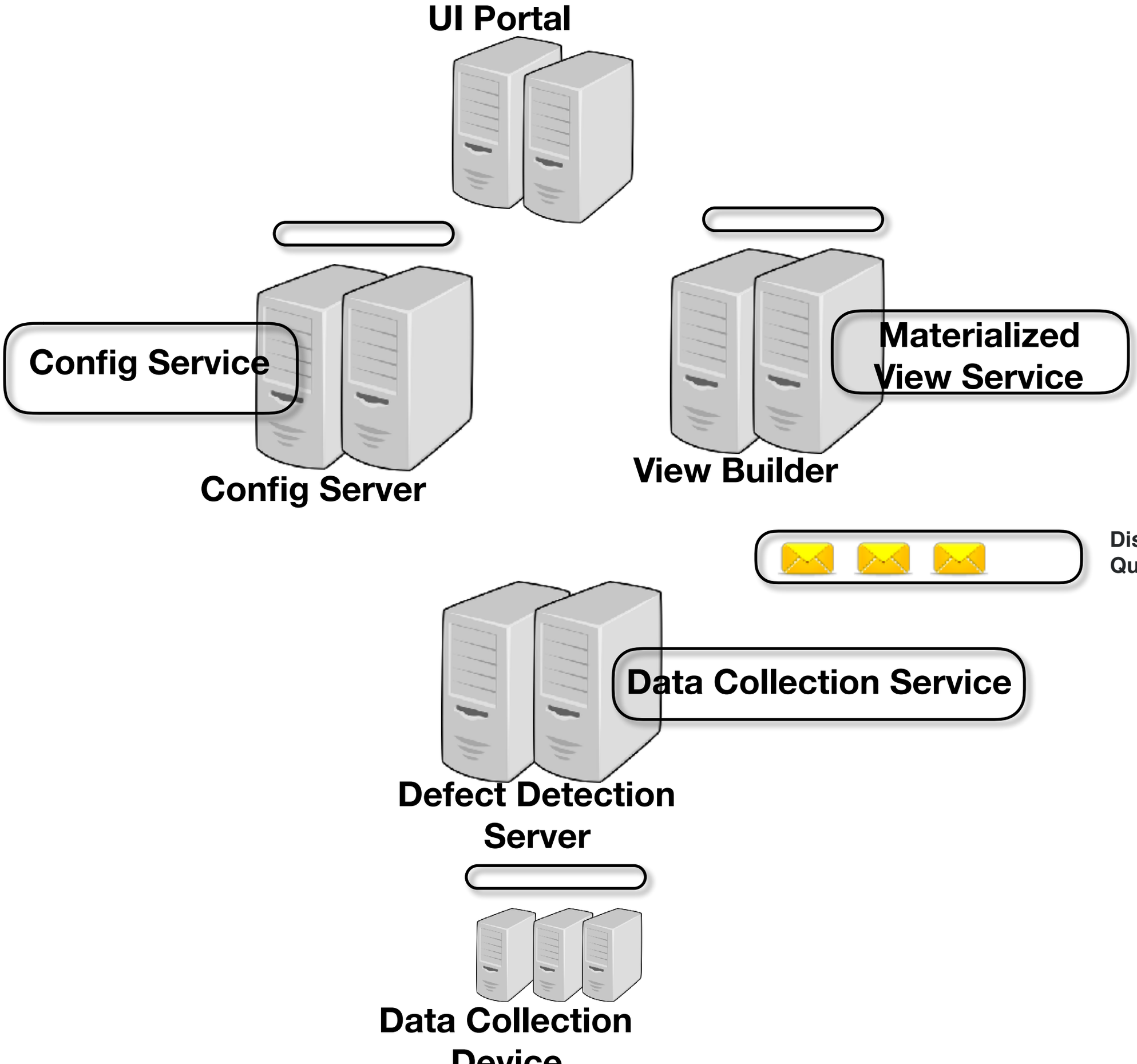
Camera

Camera

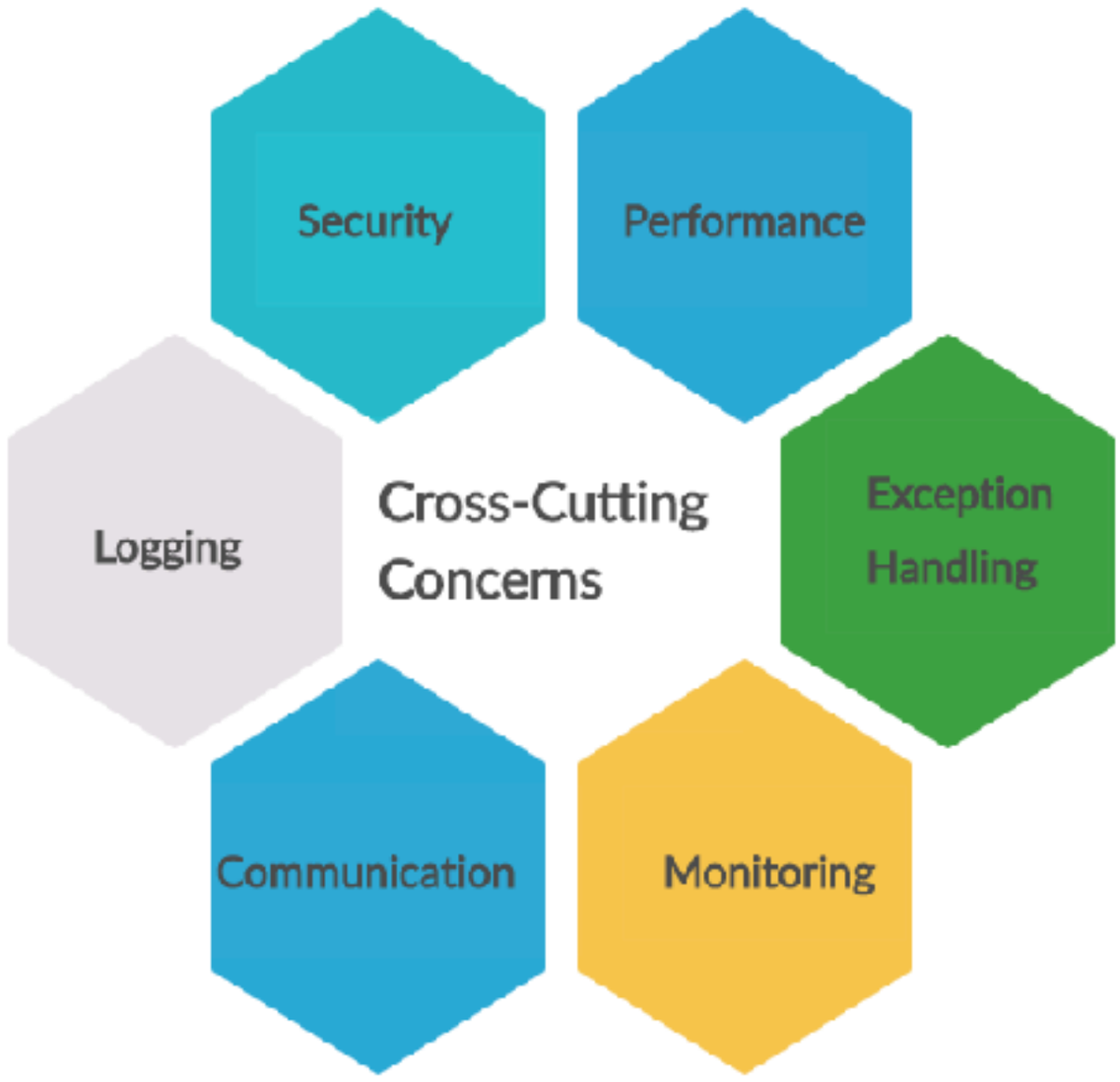
Camera

Defect Detection App

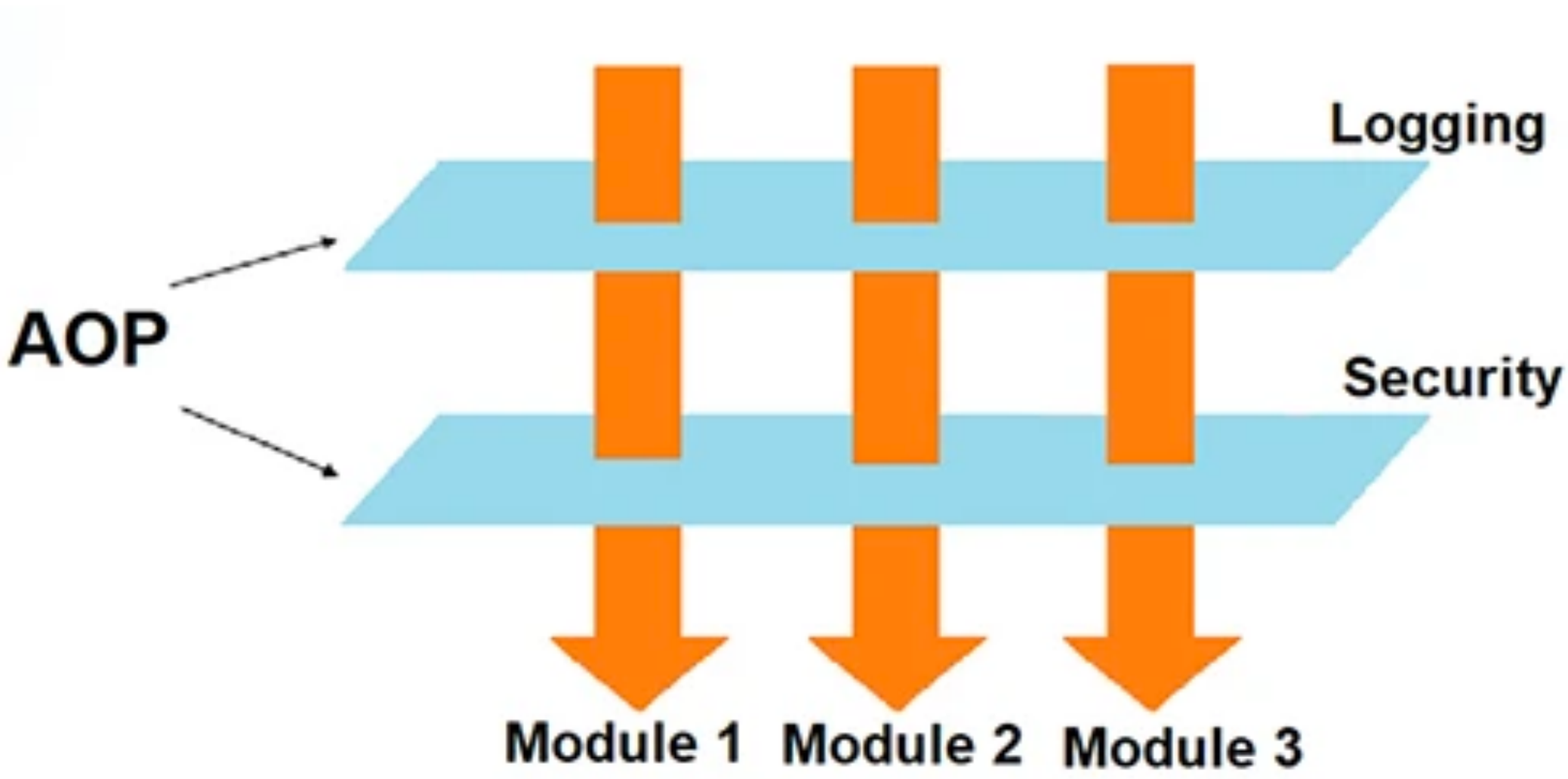
Data Collection App


UI Portal
Config Service
Config Server
Materialized
View Service
View Builder
Distributed
Queue
Data Collection Service
Defect Detection
Server
Data Collection
Device

3. Cross cutting concerns



The **crosscutting concern** is a concern which is needed in almost every module of an application.


AOP
Logging
Security
Module 1
Module 2
Module 3

Part VI

Evaluate Architecture

Architecture Trade of Analysis Method (ATAM)
Scenario generation using Utility Tree
Mapping Quality attributes with Architectural Decisions
Documenting Risks, Non Risks and sensitivity points
Lab : Evaluate Architecture using ATAM

Part 3

Arch Risks

	Risk	I	L	R	C	Notes
1	Performance of the View Todo is significantly impacted by large data volumes.	8	4	32	High	Testing to be undertaken to gauge impact.
2	Limitations of JPA API result in complex workarounds in repository implementation or workarounds in other architectural layers.	6	4	24	Medium	Can fall back to Hibernate which provides richer ORM functionality.
3	The Open Source tools and components selected lack the capabilities of equivalent commercial products limiting the features that can be implemented (e.g. GIS components).	6	3	18	High	Investigate alternative products.

- Risk – The description of the architectural risk.
- Impact to Project – The impact the risk will have on the project
    - (1 = Minor impact, 10 = Showstopper)
- Likelihood – The likelihood of the risk eventuating (1 = Low, 953 = Highly likely)
- Rating – The overall rating for the risk based on Impact * Likelihood (1 = Minor, 90 = Critical)
- Confidence – The confidence in resolving the issue should it eventuate (Low, Medium, High)

- ATAM
- ARID
- SAAM
-

Evaluation Team

Quality Attributes

Architectural Approaches

Analyze
{ Map Quality attributes with Architectural Decisions }

Trade Offs, Sensitive Points, Risks, Non Risks.

Offering mitigation strategies is *not an integral part of* the ATAM.
ATAM is about locating architectural risks.

Evaluate Architecture (ATAM)

 # identify all Architectural approaches (design)

 # A1 (CQRS)
 # A2(Cube)
 # A3 (Caching)
…

 # identify all quality requirements (requirements)

 # s1 (< 5 sec)
 # s2 (99.99%)
 # s3 (…)
 # …

 # analyse Scenario -> Approach

S1 -> A1, A2
S2 -> A6,A8, A9
S3-> ?
S4 -> A6, ?

=> Risk & trade off's

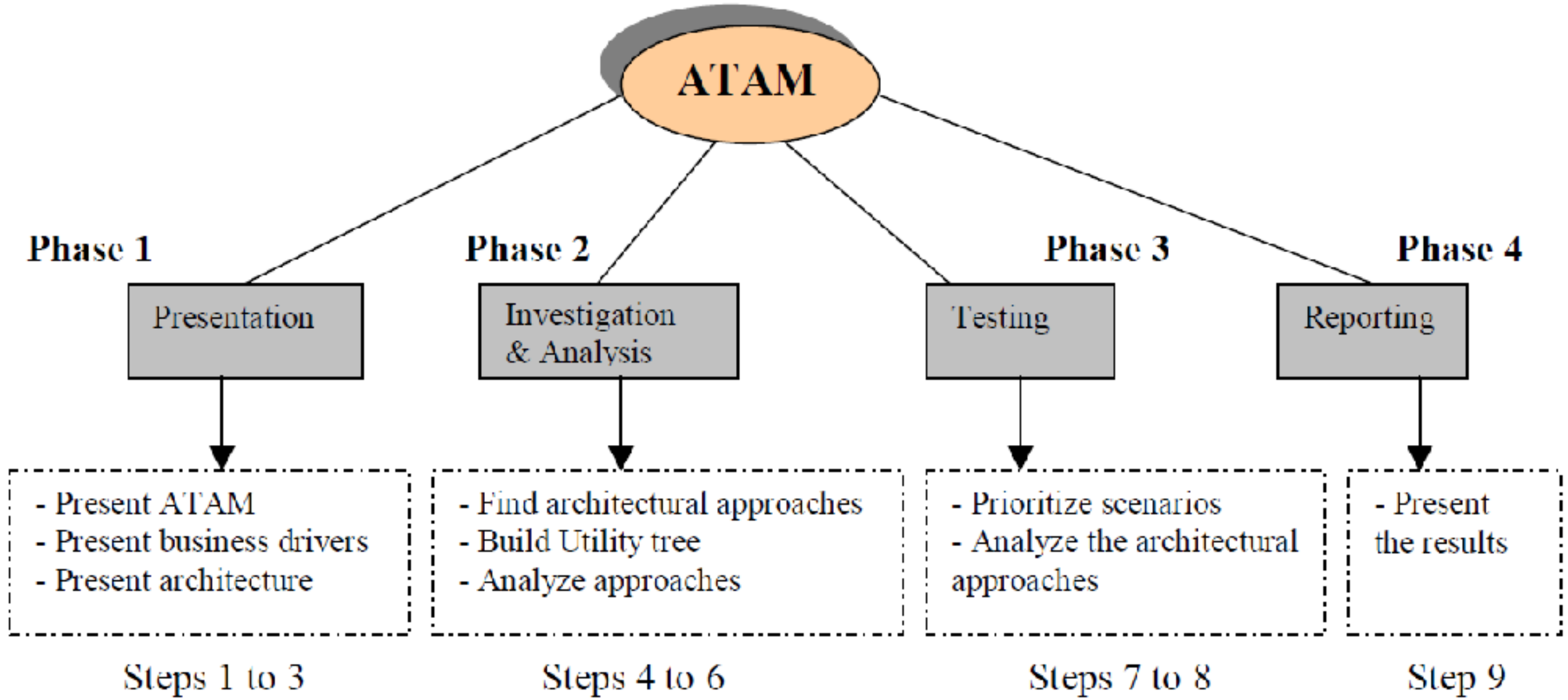
 # brainstorm for scenarios

 # s8
 # s9
 # s10
 # …

 # analyse Scenario -> Approach

S8 -> A1, A2
S9 -> A6,A8, A9
S10-> ?

=> Risk & trade off's


ATAM
Phase 1
Phase 2
Phase 3
Phase 4
Presentation
Investigation & Analysis
Testing
Reporting
- Present ATAM
- Present business drivers
- Present architecture
- Find architectural approaches
- Build Utility tree
- Analyze approaches
- Prioritize scenarios
- Analyze the architectural approaches
- Present the results
Steps 1 to 3
Steps 4 to 6
Steps 7 to 8
Step 9

Network.Commons

Queue Receiver <<Adapter>>

Queue Publisher <<Adapter>>

RestClient <<Adapter>>

Circuit Breaker

CrossCutting.Commons

Log Library

Cache Library (config)

Exception Handling Library

Data Collection Service

<<write>>

Distributed Queue

Defect Detection Service

<<read>>

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Config Service

Distributed Queue

<<read>>

Materialized View Service

<<CRUD>>

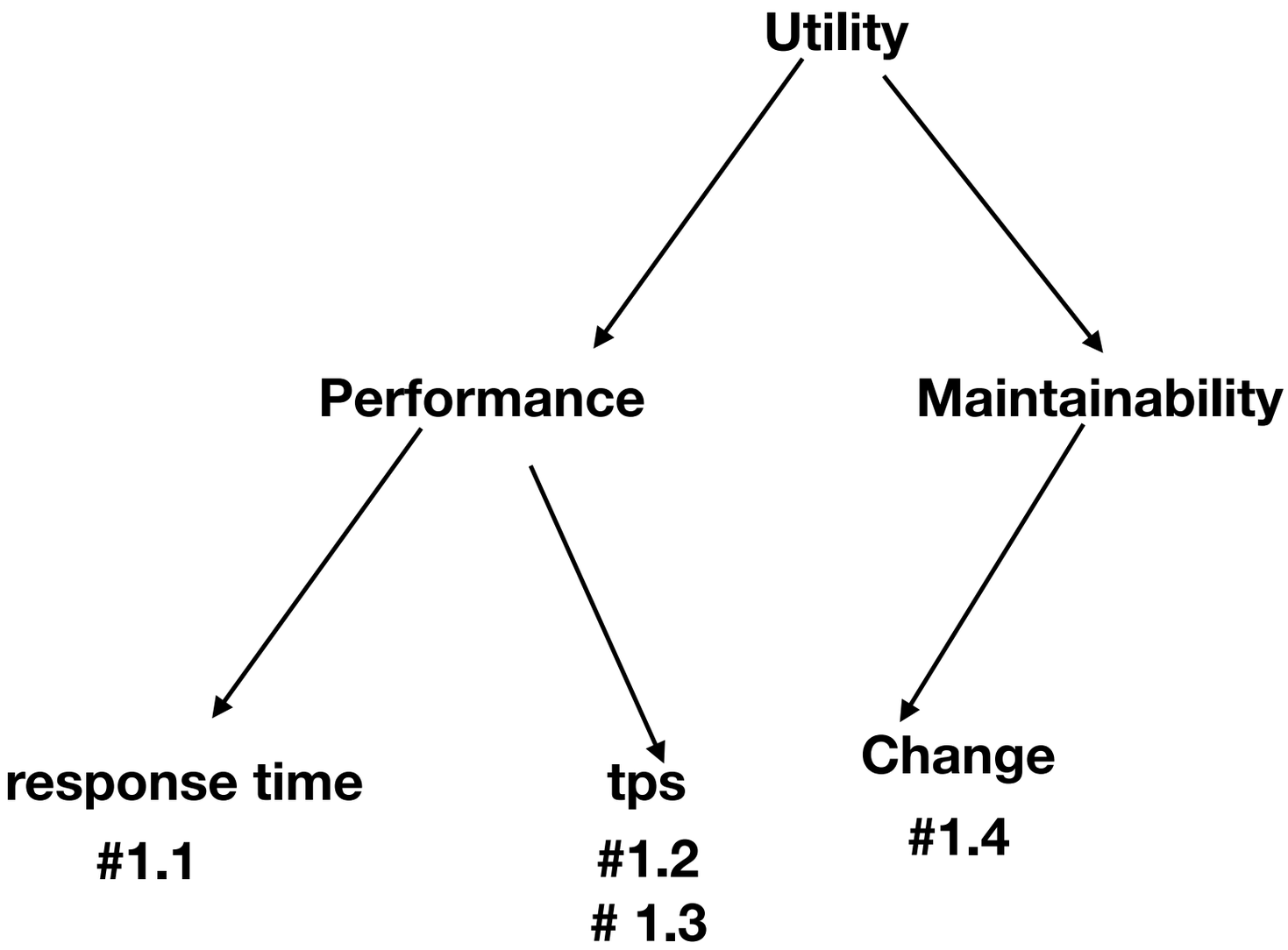
<<read>>

UI Portal

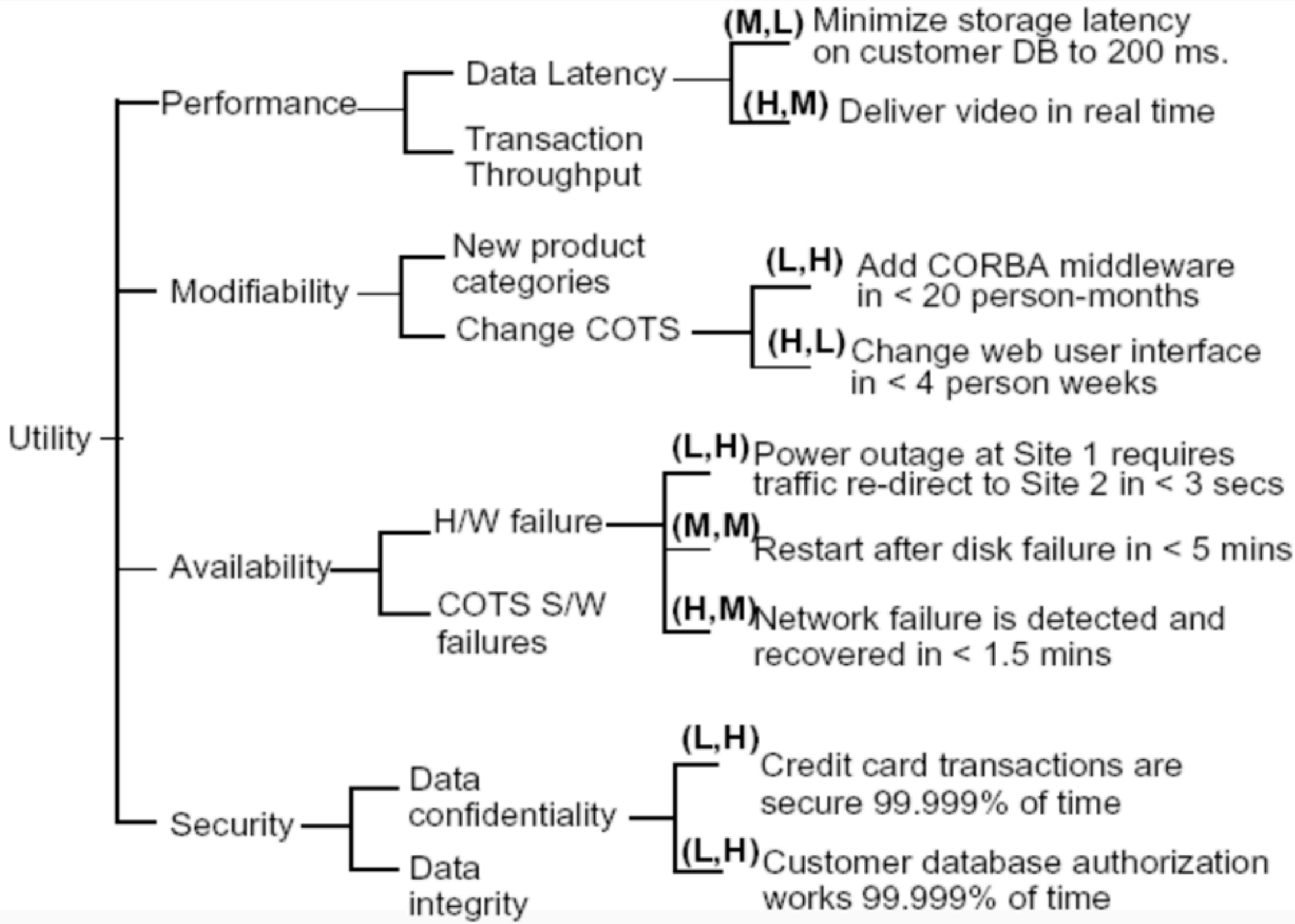
1. Used pipes and filter in image pre and post processing
2. Used Message Queue to scale input
3. Used cache to hold config data
4. Used adapters to remove coupling with vendor libraries
5. Debug and exceptions log
6. Circuit breaker for fail fast

1.1 The data collection device should be able capture images of CB at 2 images/sec
1.2 The system should be able to capture images from at least 10 belts during peak load
1.3 the defect detection system should be able to match CB for 100 images during peak load
1.4 Developer should be able to add additional pre processing logic to image processing in less than 3 man days.

SC#	A#	Trade off	Risks
1.1	A2		
1.2	A2		
1.3			May need GPU
1.4	A1		



1.1 The data collection device should be able capture images of CB at 2 images/sec
1.2 The system should be able to capture images from at least 10 belts during peak load
1.3 the defect detection system should be able to match CB for 100 images during peak load
1.4 Developer should be able to add additional pre processing logic to image processing in less than 3 man days.


Utility
Performance
Data Latency
(M,L) Minimize storage latency on customer DB to 200 ms.
Transaction Throughput
(H,M) Deliver video in real time
Modifiability
New product categories
Change COTS
(L,H) Add CORBA middleware in < 20 person-months
(H,L) Change web user interface in < 4 person weeks
Availability
H/W failure
(L,H) Power outage at Site 1 requires traffic re-direct to Site 2 in < 3 secs
(M,M) Restart after disk failure in < 5 mins
COTS S/W failures
(H,M) Network failure is detected and recovered in < 1.5 mins
Security
Data confidentiality
(L,H) Credit card transactions are secure 99.999% of time
Data integrity
(L,H) Customer database authorization works 99.999% of time

Scenario Utility Tree

- Utility
  - Performance
    - Data latency
      - — Minimize storage latency on customer DB to 200 ms
      - — Deliver video in real time
    - Transaction throughput
      - — Maximize average throughput to authentication server
  - Modifiability
    - New Product Categories
    - Change COTS
      - — change web user interface in < 4 person weeks
  - Availability
    - Hardware Failure
      - — power output at site 1 requires traffic redirect to site 3 in < 3 s
      - — network failure is detected and recovered in < 1,5 min
  - Security
    - Data confidentiality
      - — customer database authorisation works 99,999% of time

Scenario Brain Storming

Sc#	Description	Quality Att.	Votes
4	Dynamically replan a dispatched mission within 10 minutes.	Performance	28
27	Split the management of a set of vehicles across multiple control sites.	Performance, Modifiability, Availability	26
10	Change vendor analysis tools after mission has commenced without restarting system.	Integrability	23
12	Retarget a collection of diverse vehicles to handle an emergency situation in less than 10 seconds after commands are issued.	Performance	13
14	Change the data distribution mechanism from CORBA to a new emerging standard with less than six person-months' effort.	Modifiability	12

Business Drivers

Phase - 1

Architect Centric

Concentrates on eliciting and analyzing architectural information.

Phase - 2

Stakeholder Centric

Elicits points of view from a more diverse group of stakeholders, and verifies the results of the first phase

Scenario:	E-connector looses connection with SAP
Attribute:	Availability
Stimulus:	Temporary network fault
Response:	The system has an overall availability of 99,25% (max 2 hour down/ month)

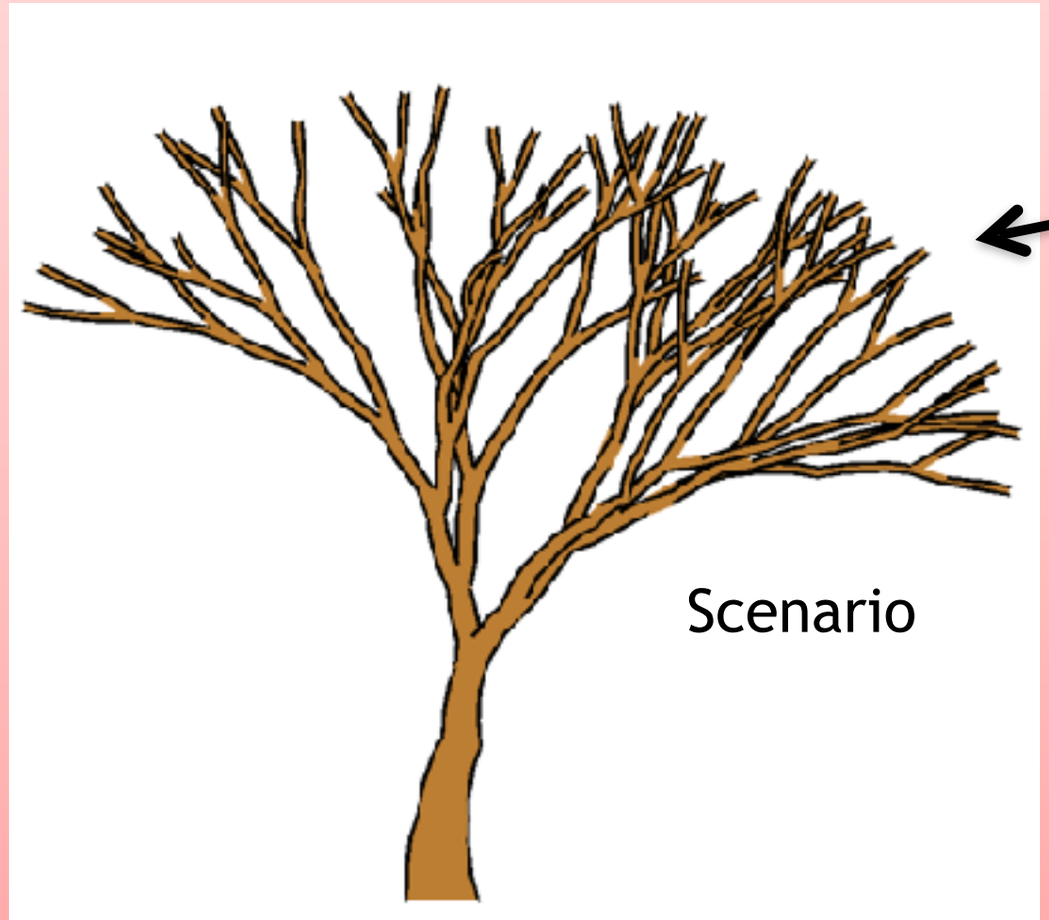
Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
DCTM content server runs on a clustered environment with 2 nodes	Common mode failure can not be handled			Probability of common mode failure is low
Integration relationship between SAP and Documentum is 'data consistency‘ and is not protected	Human user must report malfunction		From complaint to resolution > 2 hours	

Scenario:	Invoice poster needs e-document for data entry in SAP/R3
Attribute:	Perfomance-Latency
Stimulus:	Document request to Documentum
Response:	Document is available for processing in less than 10 s

Architectural decision	Sensitivity	Trade-off	Risk	Non-risk
E-documents are scanned in color at 200 dpi	Size of document is sensitive to quality of scanning	Usability vs Performance	Document too large for roundtrip in 10 s.	
E-documents are not cached	Every document must be fetched from DMTM	Development cost vs bandwidth cost	Document roundtrip time exceeds 10 s.	

Phase - 1

Architect Centric



Utility Tree

Business Drivers

Phase - 2

Brain Storming

Scenarios

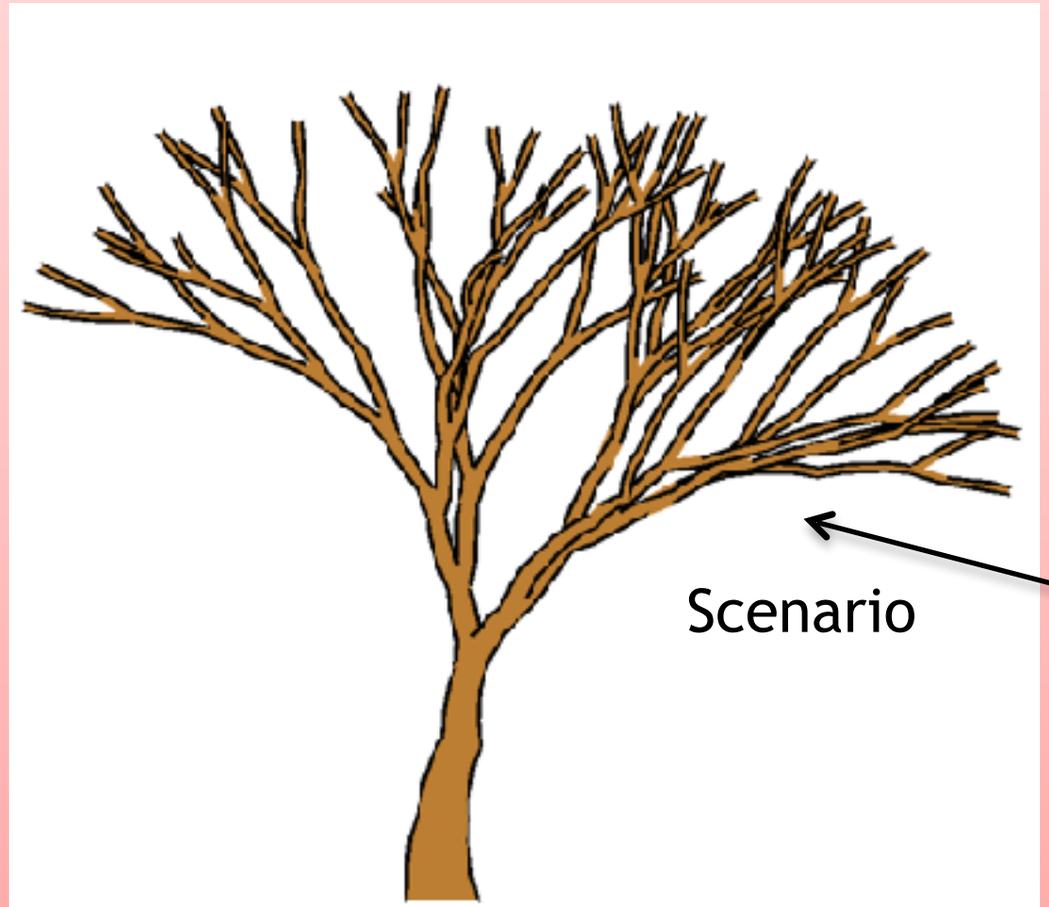
Stakeholder Centric

Comparing Scenarios with utility tree reveals disconnects between what the architects and stakeholders believe to be important system qualities.

Three things may happen when a scenario is placed in the utility tree

Phase - 1

Architect Centric



Utility Tree

Business Drivers

Phase - 2

Brain Storming

Stakeholder Centric

Scenarios

Each High Priority Scenario is inserted to an appropriate leaf node in the utility tree.

Phase - 1

Architec

Scenario

Utility Tree

Phase - 2

The Scenarios matches well to an existing leaf node.

1. Scenario has been already considered in utility tree exercise. (Duplicate)

Phase - 1

Architec

Scenario

Utility Tree

Phase - 2

The Scenarios dose not match to any existing leaf node in utility tree, but it can be associated with an existing branch of the utility tree .

1. The Quality attribute addressed by the scenario has been covered by other scenarios in the utility tree.

2. A scenario may be associated with multiple quality attributes. In that case the scenario is placed into the leaves of several branches.

Phase - 1

Architec

Scenario

Utility Tree

Phase - 2

The Scenario cannot be associated to any branch of the utility tree

1. Scenario expresses a quality requirement that has not been addressed previously in the utility tree exercise.

2. The architect may have failed to consider an important quality requirement.

3. Further Analysis of Architecture on this Scenario is required. (Repeat Phase 1)