# ARCHITECTURE TYPES

G. Molines

2020-2021

Université Nice Sophia Antipolis

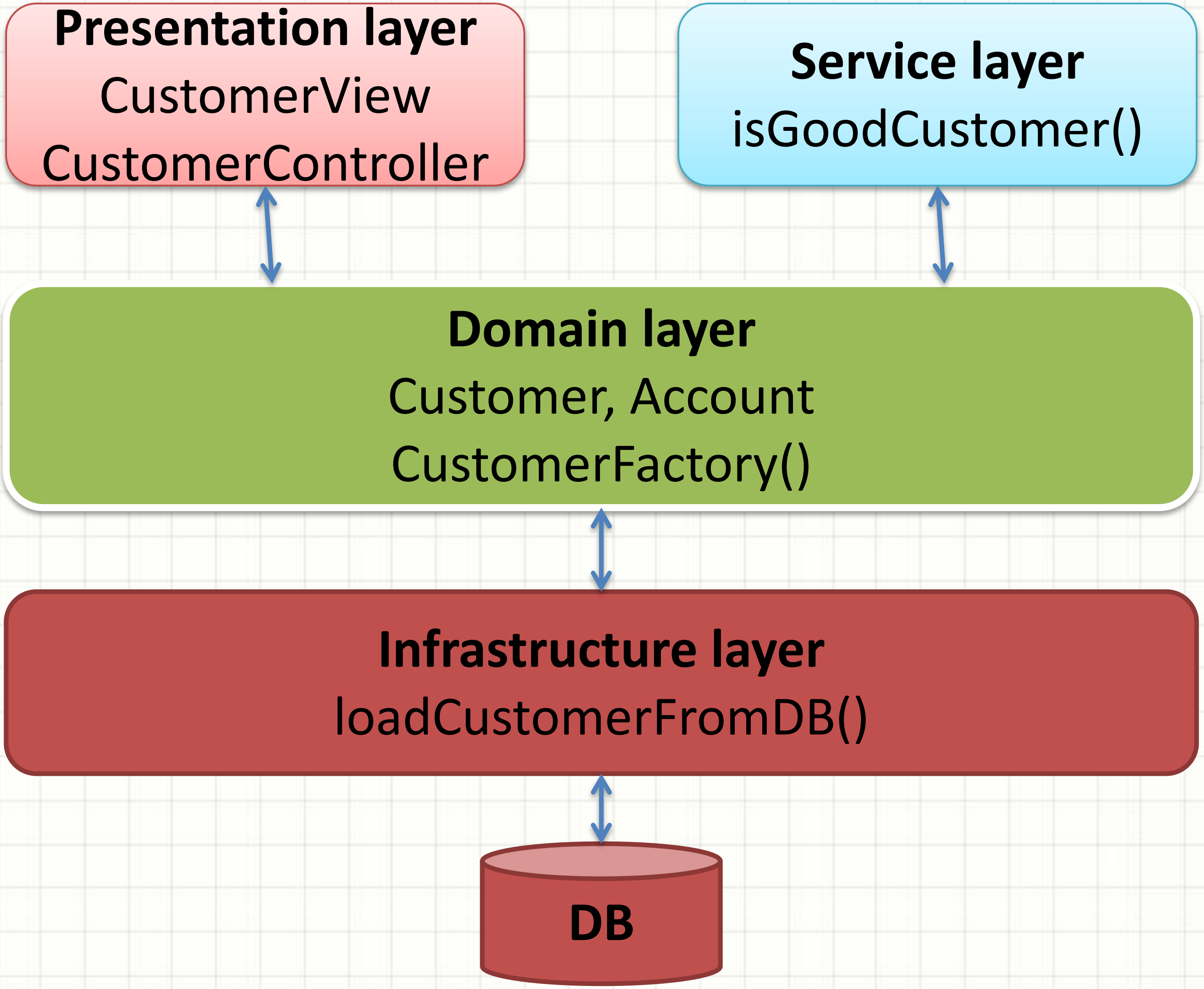
POLYTECH NICE-SOPHIA

# Agenda

- Layered architecture
- SOA
- Event-driven architecture
- Space-based architecture
- Micro service

# Layered Architecture

# Example – banking



**Presentation layer**

CustomerView

CustomerController

**Service layer**

isGoodCustomer()

**Domain layer**

Customer, Account

CustomerFactory()

**Infrastructure layer**

loadCustomerFromDB()

**DB**

# Benefits

- Less code per layer
- Reduced complexity
  - Easier maintenance
  - Easier extensibility
  - Easier to test
- Code reuse

# SERVICE ORIENTED ARCHITECTURE

Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

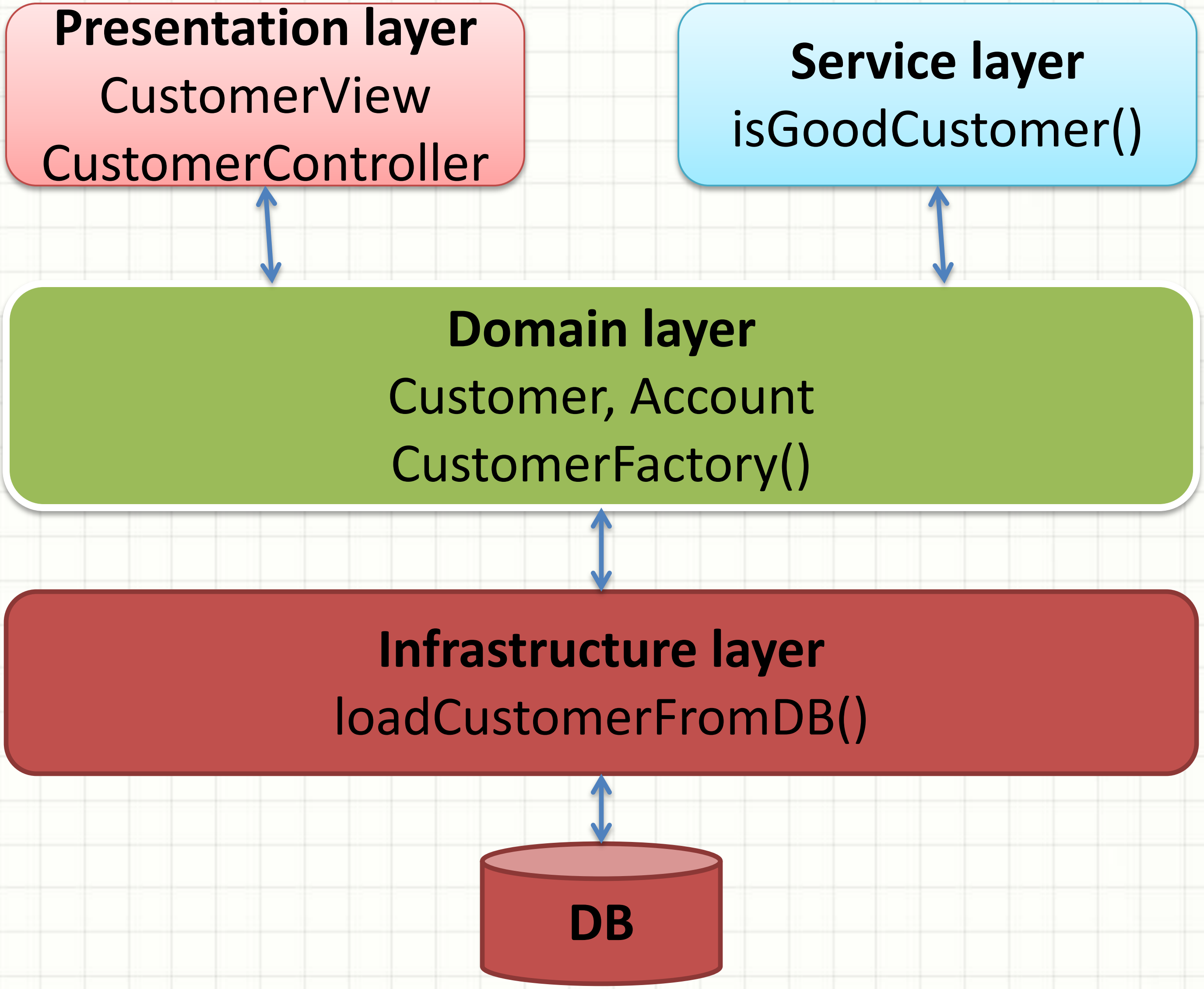
# What is SOA?

- An architecture based on reusable building blocks, “services”, which:
  - Are autonomous, stateless business functions
  - Use a request/responses pattern
  - Use well-defined, standard interfaces

# SOA features

- Communication through standard protocols
  - XML, SOAP, JSON, HTTP, etc.
- OS-, language- and platform-independent
- Discoverability
  - Service registries
- Loosely coupled
- Explicit boundaries

# Example – banking layers



**Presentation layer**
CustomerView
CustomerController

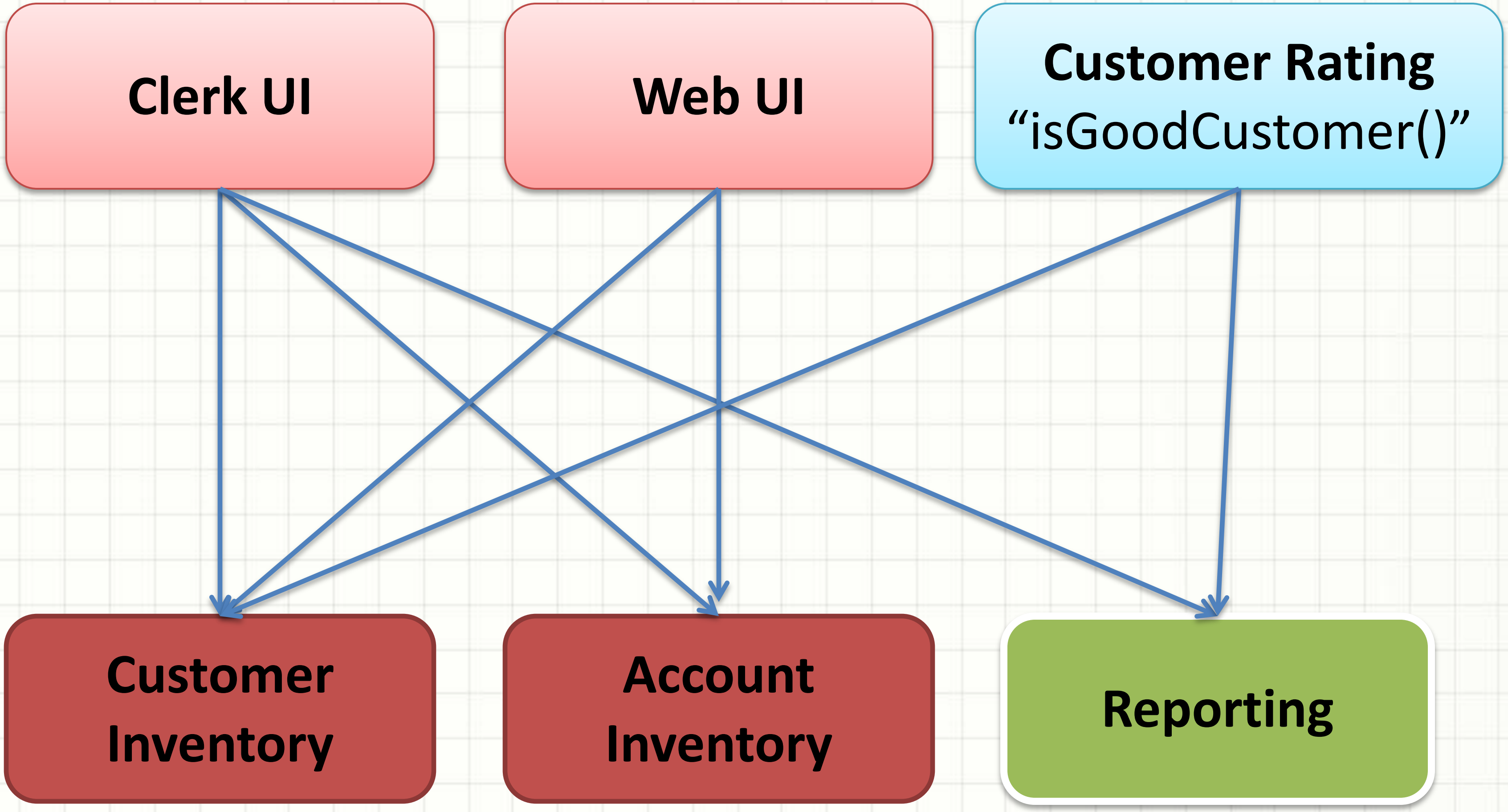
**Service layer**
isGoodCustomer()

**Domain layer**
Customer, Account
CustomerFactory()

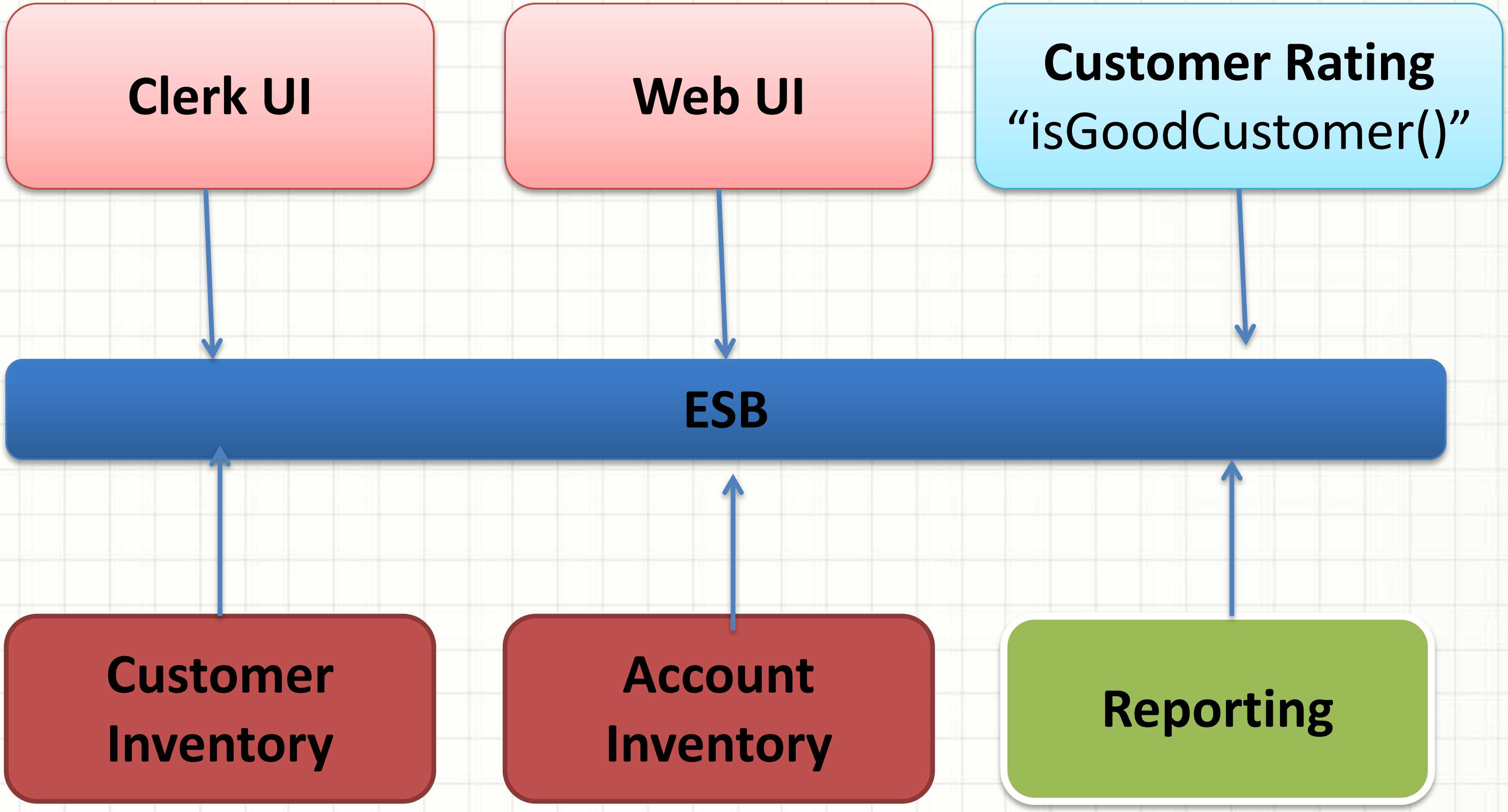
**Infrastructure layer**
loadCustomerFromDB()

**DB**

# Example – banking SOA



# Example – banking SOA



Clerk UI

Web UI

Customer Rating
“isGoodCustomer()”

ESB

Customer Inventory

Account Inventory

Reporting

# Event Driven Architecture

# Interactions

| Type of interaction | Initiator | Details |
|---|---|---|
| Time-driven | Time | Eg: batch processing |
| Request-driven | Client | Client / server |
| Event-driven | Event | Open ended |

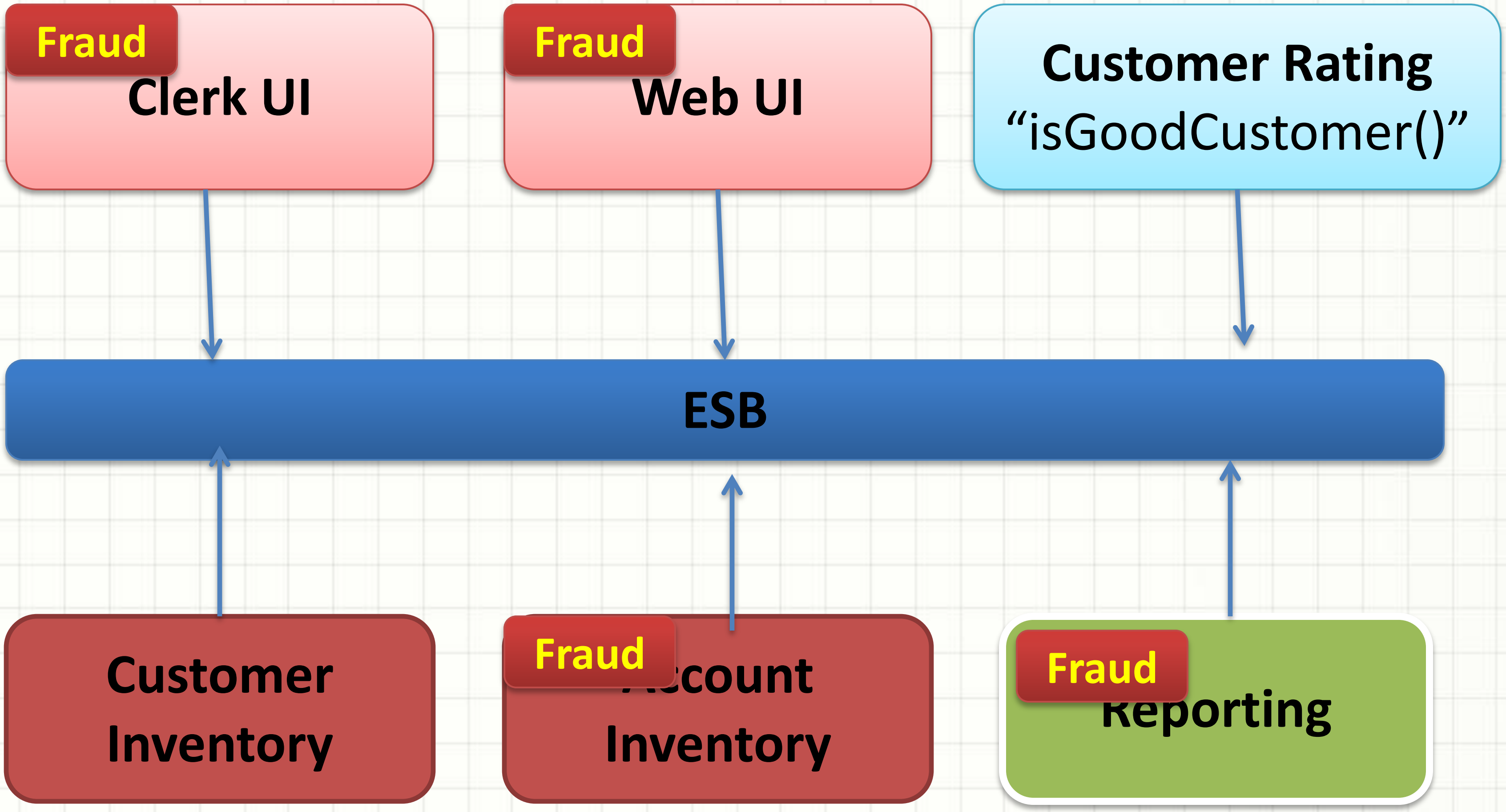
# Event-driven architecture

- Architecture
  - Producer / event bus / consumers
  - No central control
  - Stateless
- Event types
  - Notification
  - Transactional
    - Event reported can change something
    - Reliable transport

# Considerations

- Time ordering of events
- Once and only once delivery
- Payload format – need schema support
- -> choice of message broker
  - Resilience
  - Slow consumers
  - Durable / non durable subscriptions
  - Payload
- Security: event can carry sensitive data

# Let’s add fraud detection



Fraud

Clerk UI

Fraud

Web UI

**Customer Rating**

“isGoodCustomer()”

ESB

Customer Inventory

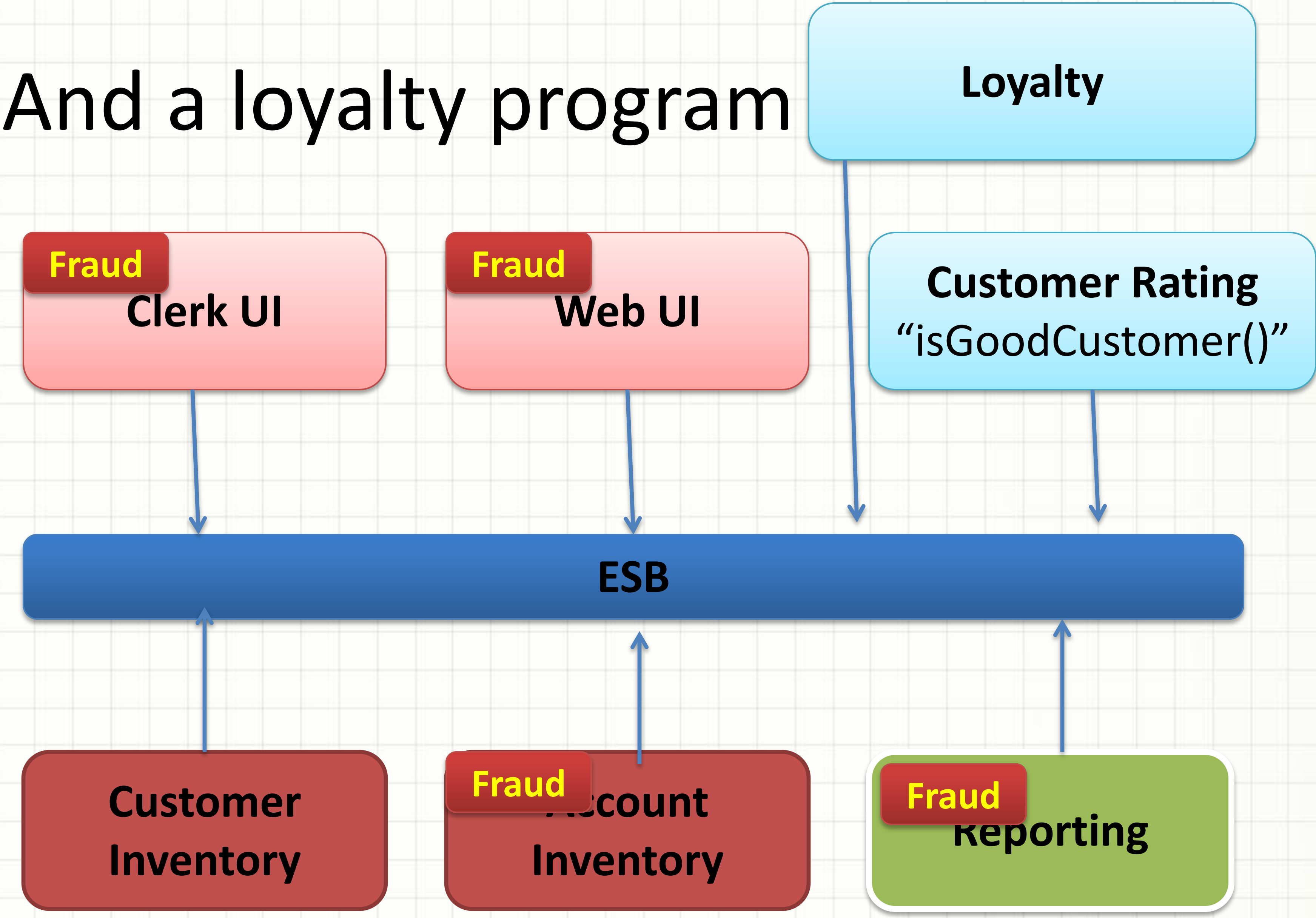
Fraud

Account Inventory

Fraud

Reporting

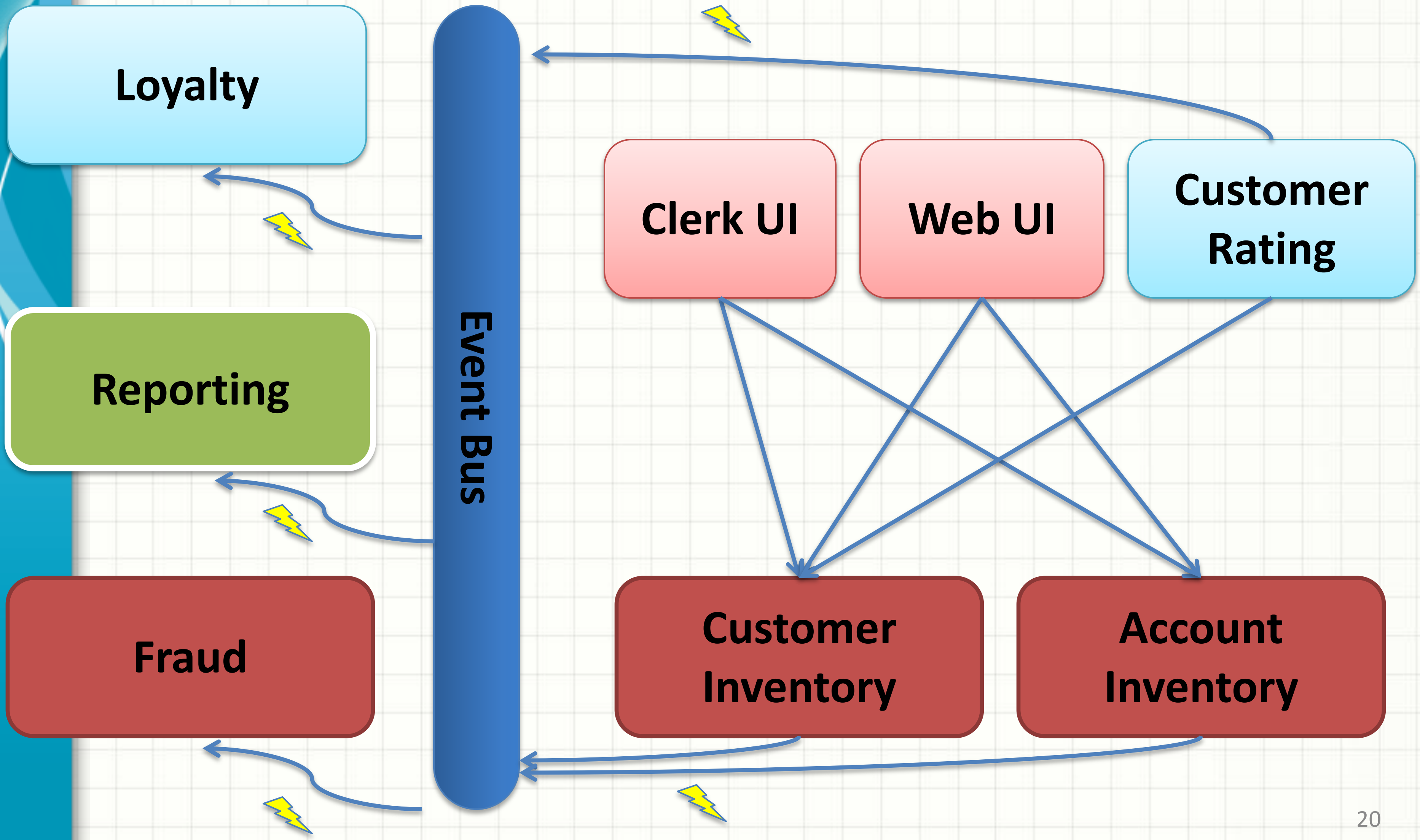
# And a loyalty program



# Benefits

- Less integrations point (not point to point), fire & forget
- Performance, scale
- SOA doesn't handle aspects very well. Some logic spreads all over and domain become polluted by extra logic
- EDA handles cross-cutting aspects better
- Designed for extensibility

# SOA + EDA


Loyalty
Reporting
Fraud
Event Bus
Clerk UI
Web UI
Customer Rating
Customer Inventory
Account Inventory

# EDA use cases

- Government: border control, taxes
- Compliance (Eg: regulatory)
- Track and trace
- Fraud, trading
- Equipment monitoring

# SOA / EDA

- SOA
  - Messages drive computations
  - Interfaces of operations
  - Sync or async
  - Business processes attached to services
  - Sustained performance depends on weakest link
- EDA
  - Events drive computations
  - Interfaces of events
  - Always async
  - Business  processes attached to complex events
  - Harder to test
  - EDA can absorb peaks (but requires tuning of the system)

# Space Based Architecture

# What is Space-based architecture?

- Linear scalability
- Multiple processing units
- Requires problems that can be sliced in parallel chunks
- Computing grid vs. data grid
  - See SETI@Home and other computational grids

# Questions?