Managing Complex Scenarios

Michael L Perry
SOFTWARE MATHEMATICIAN

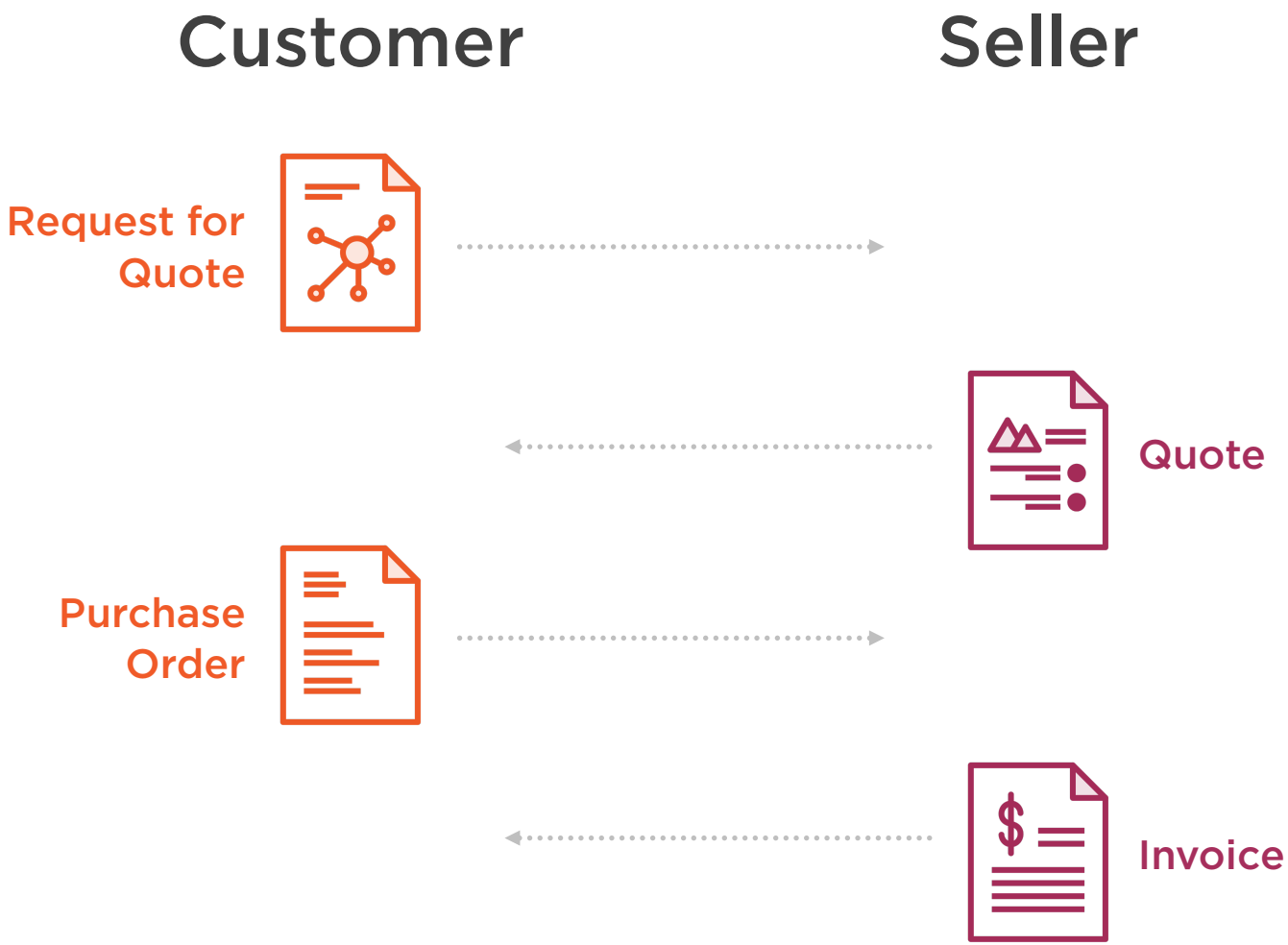
@michaellperry immutablearchitecture.com

Business Processes

Simple

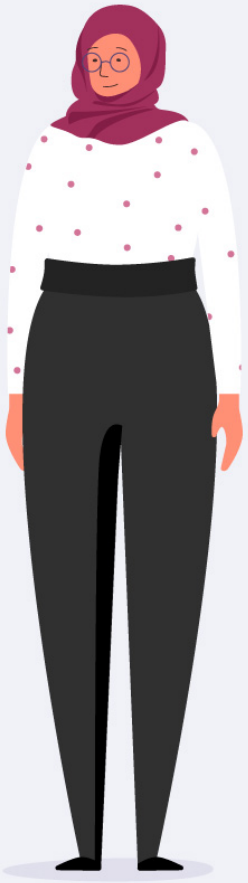
Complex

Transacting Business via Document Exchange



Computerizing Business Processes

- Limitations of computing
- Data structures
- Record locking
- Sequential processing



Learned the sales process

Discovered complexity

Understanding interactions

Invariants

True statements

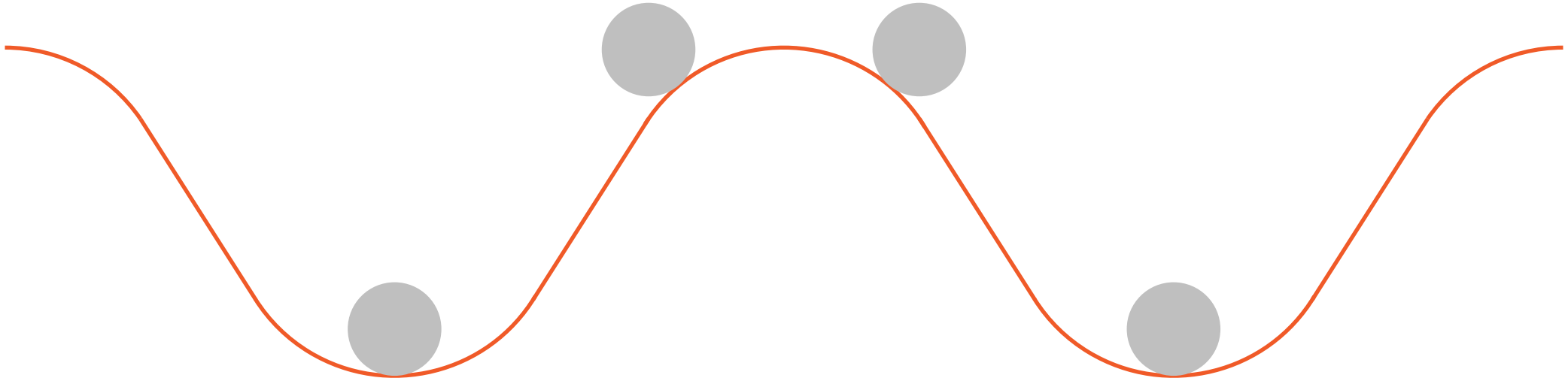
Money is conserved

Inventory is not negative

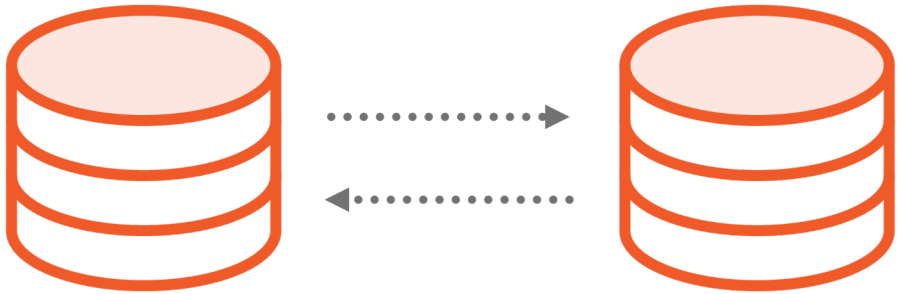
Things can’t be sold twice

Owns product if and only if paid for

Breaking and Restoring Invariants



Consistency



Between components

Invariants restored

Partner Sales

GloboTicket **Partner**

Inventory

Partner Sales

GloboTicket

Partner

Inventory

Accounting

Partner Sales

GloboTicket	Partner
Inventory	
Accounting	

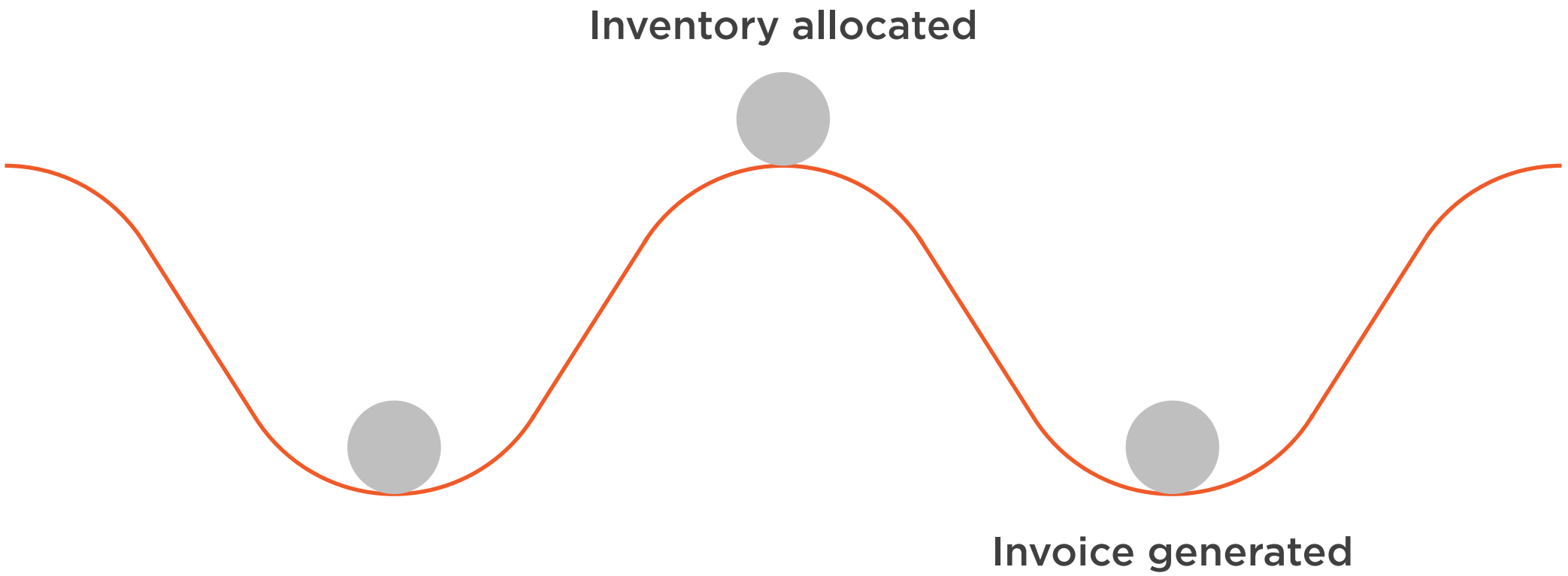
Partner Sales

GloboTicket **Partner**

Inventory

Accounting

Breaking and Restoring Invariants



Database Transactions

Atomic

Can’t see invariants broken afterward

Consistent

Invariants are restored

Isolated

Can’t see invariants broken in real time

Durable

Changes persist

Database Transactions

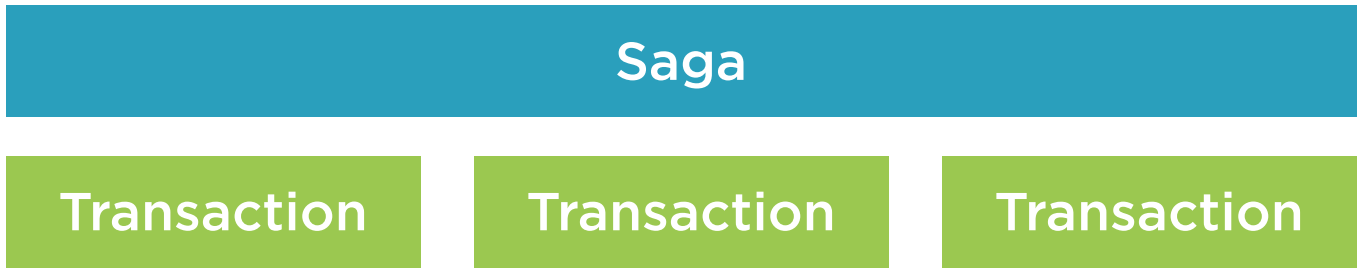
Limited in time and scope

Consumes resources

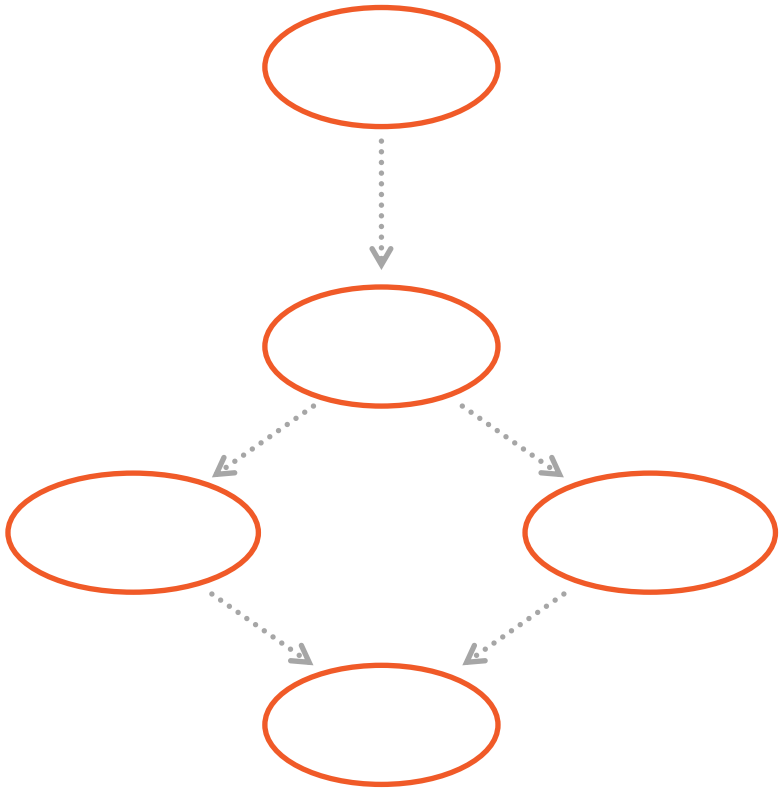
Costly to distribute

Saga Pattern

1987
Hector Garcia-Molina
Kenneth Salem
Princeton Department of Computer Science



State Machine



Messages cause transitions

Correlation ID

- Identify a business process
- From business domain
- Identifies machine instance

Instance holds current state

Process message and transition to next state

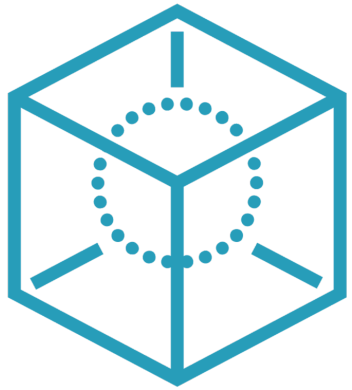
Sales Service

Reserve funds

Lock inventory

Capture funds

Allocate inventory



Sales Service

Purchase Ticket

Reserve funds

Funds Reserved

Lock inventory

Inventory Locked

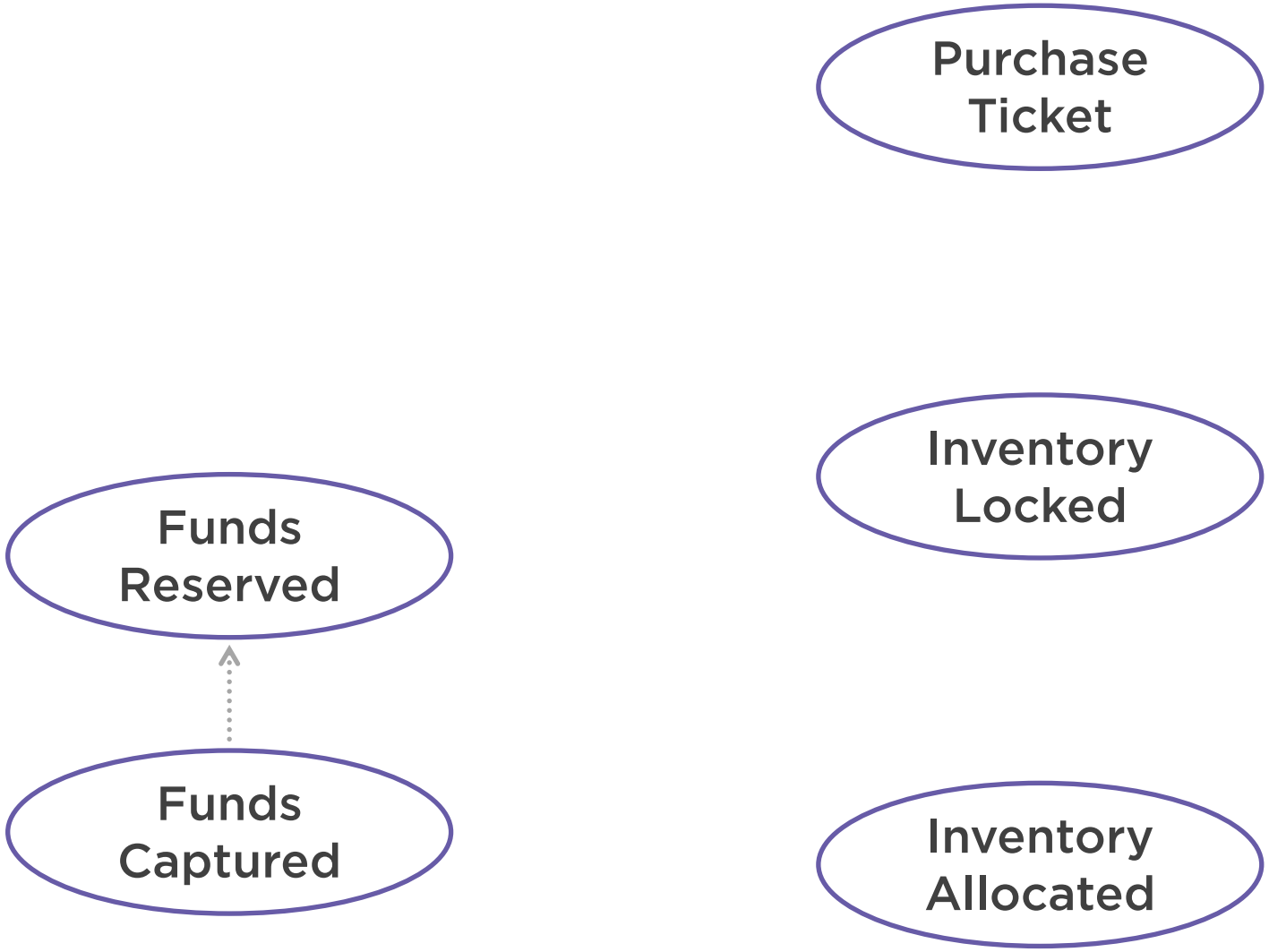
Capture funds

Funds Captured

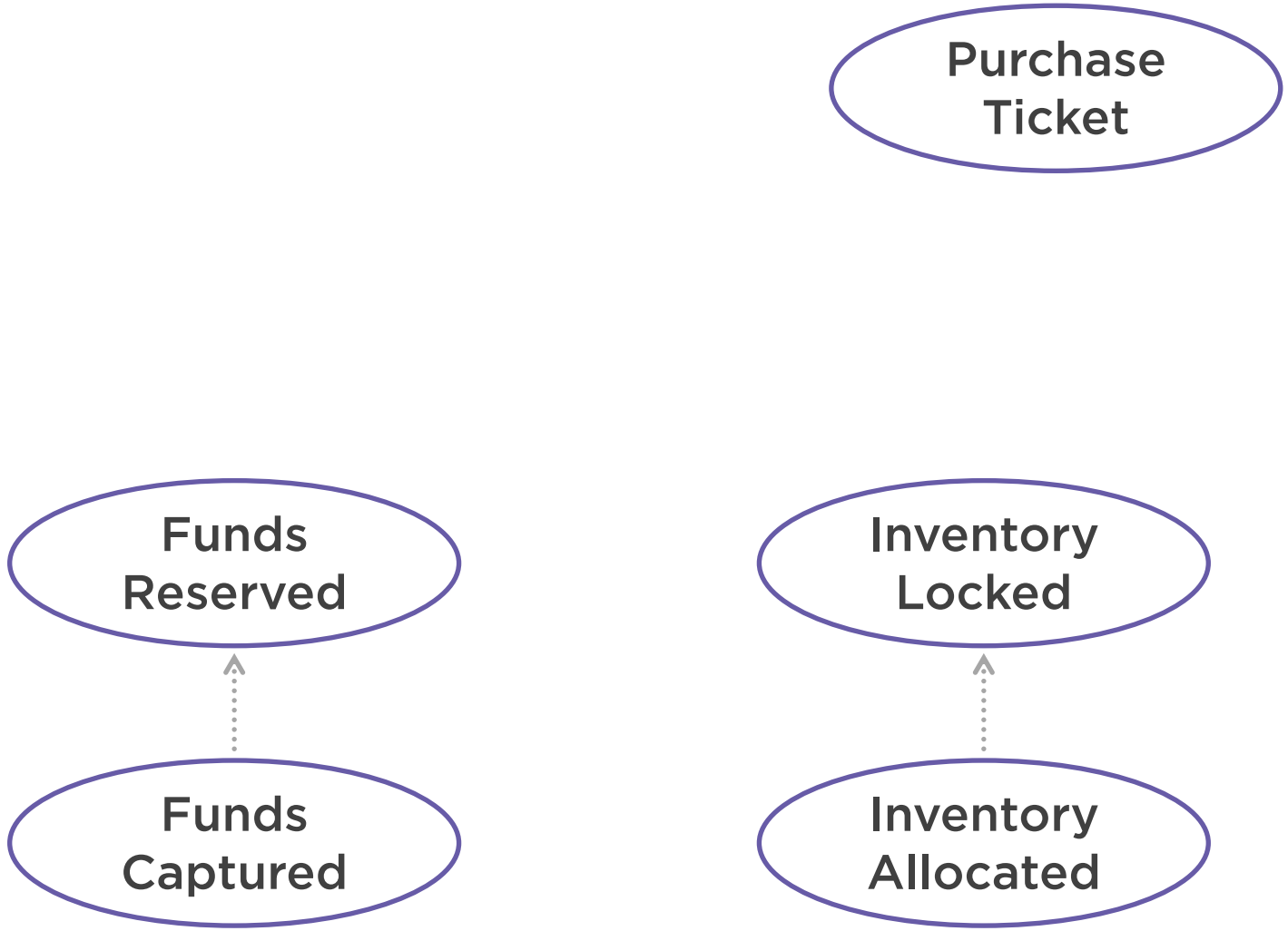
Allocate inventory

Inventory Allocated

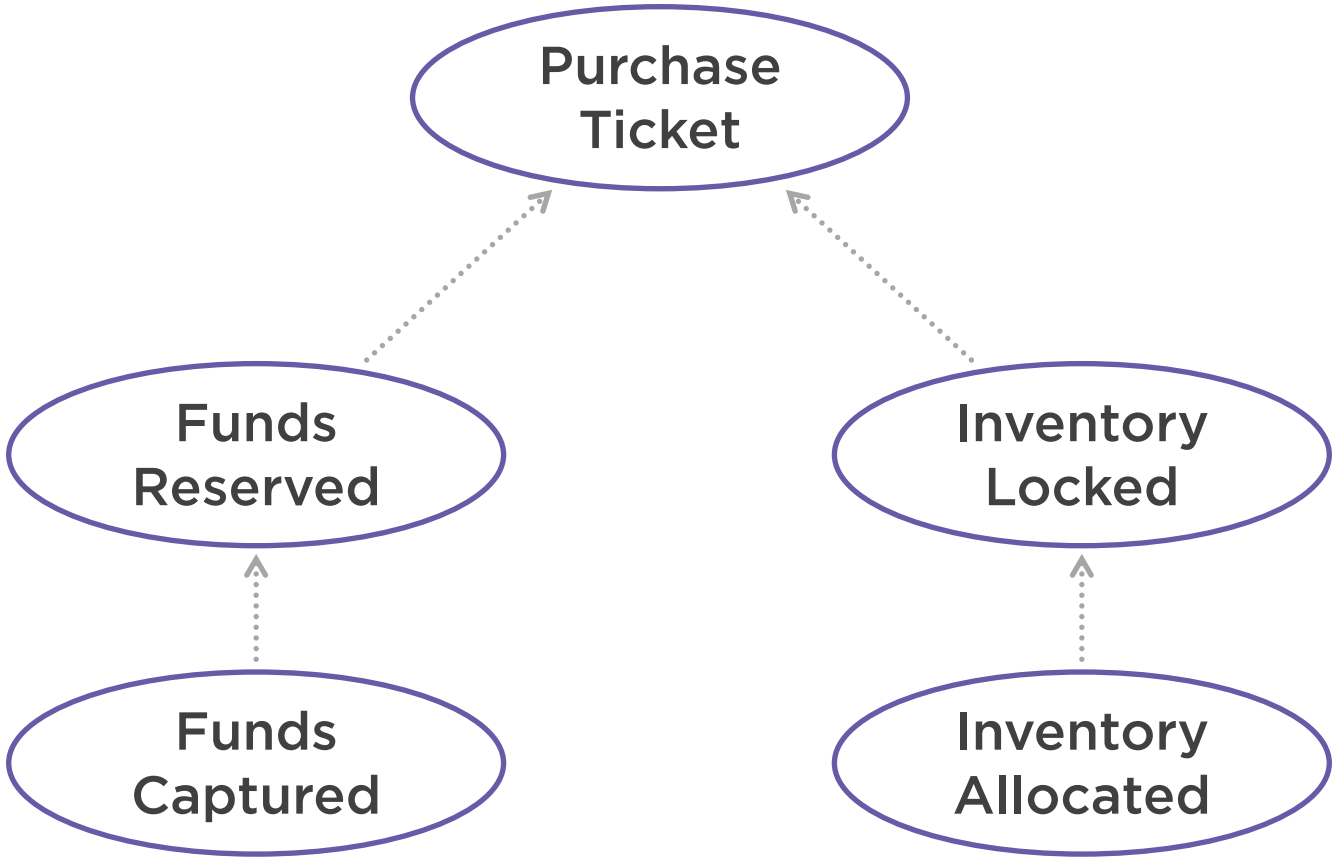
Causal Relationships

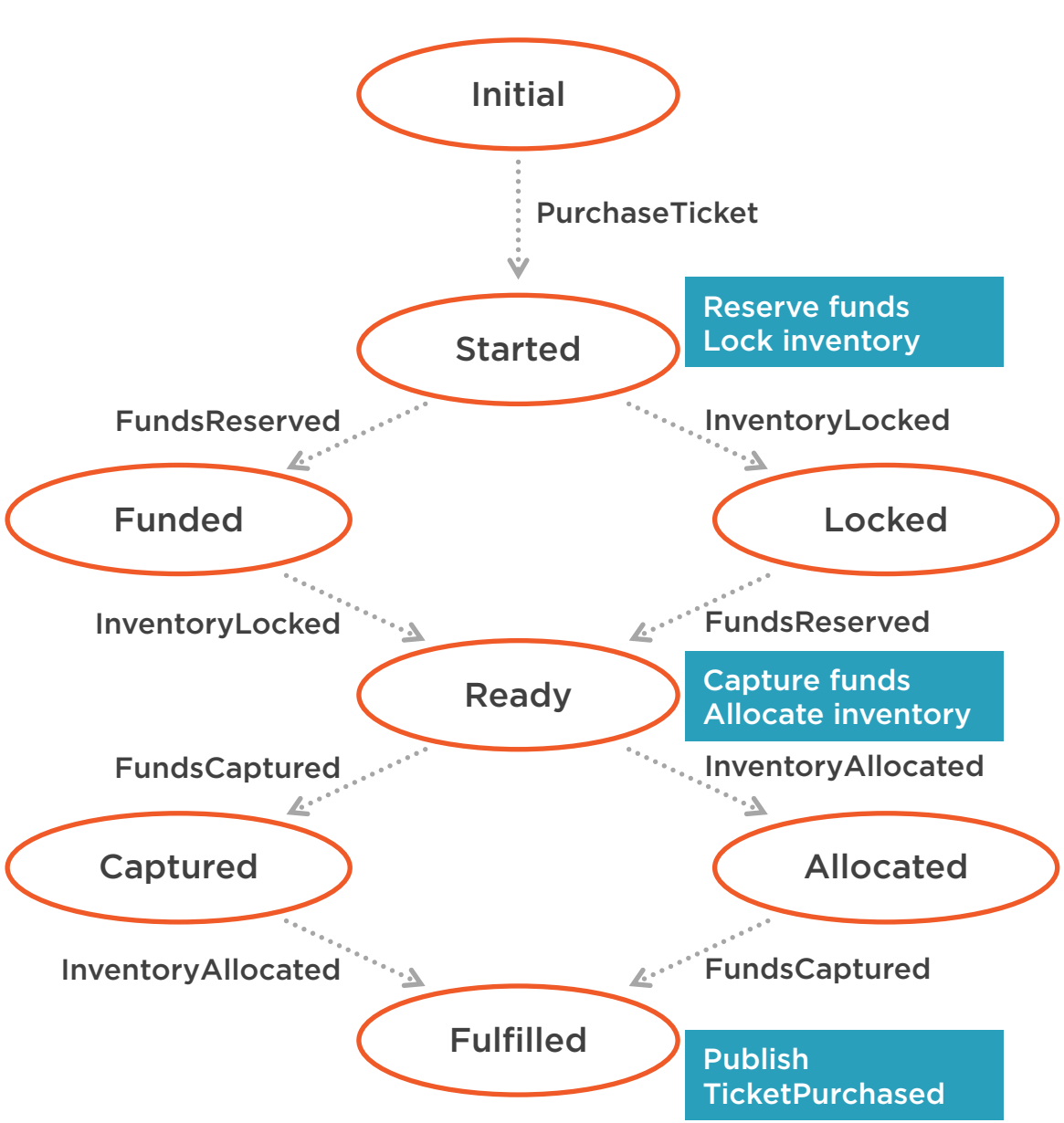


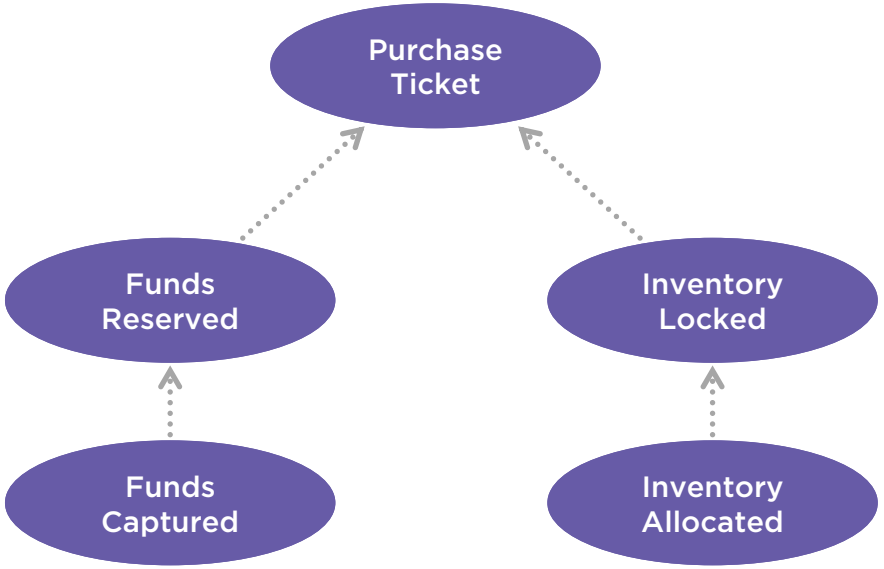
Causal Relationships



Causal Relationships

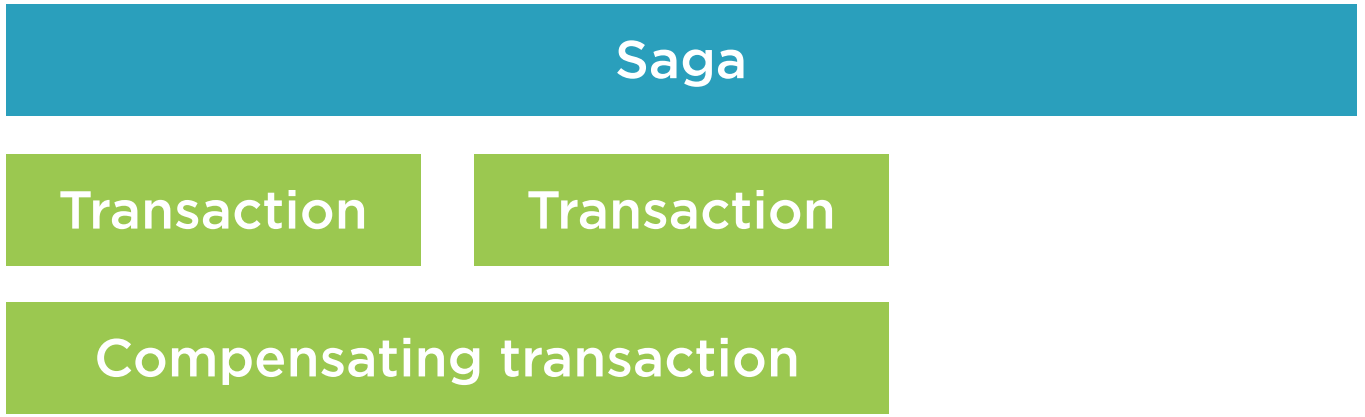


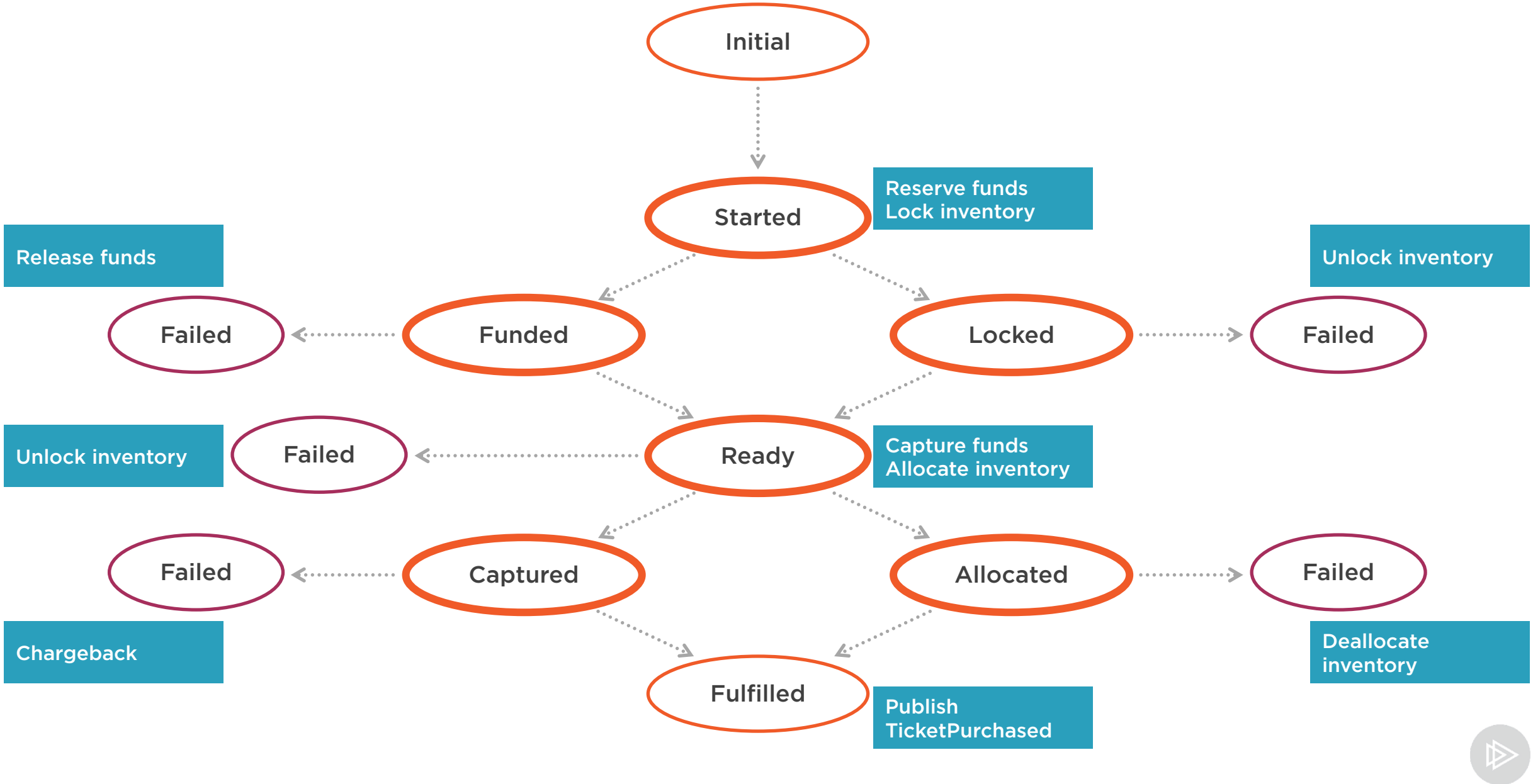

Initial
PurchaseTicket
Started
Reserve funds
Lock inventory
FundsReserved
InventoryLocked
Funded
Locked
InventoryLocked
FundsReserved
Ready
Capture funds
Allocate inventory
FundsCaptured
InventoryAllocated
Captured
Allocated
InventoryAllocated
FundsCaptured
Fulfilled
Publish
TicketPurchased


Purchase
Ticket
Funds
Reserved
Inventory
Locked
Funds
Captured
Inventory
Allocated

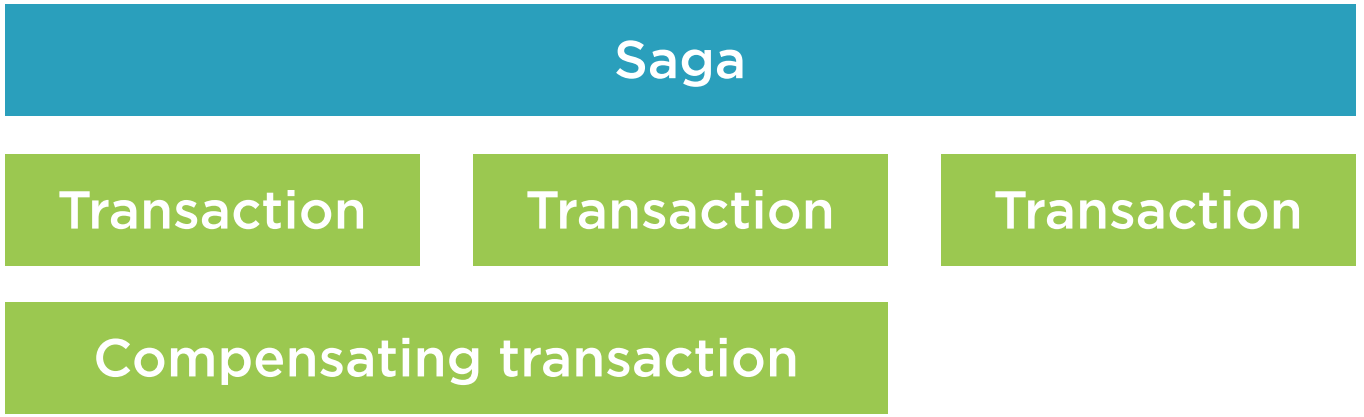
Restoring Invariants

Restoring Invariants




Initial
Started
Reserve funds
Lock inventory
Release funds
Failed
Funded
Locked
Unlock inventory
Failed
Unlock inventory
Failed
Ready
Capture funds
Allocate inventory
Failed
Captured
Allocated
Failed
Chargeback
Deallocate inventory
Fulfilled
Publish TicketPurchased

Restoring Invariants



Detecting and Mitigating Failure

Add early operations to ensure later operations

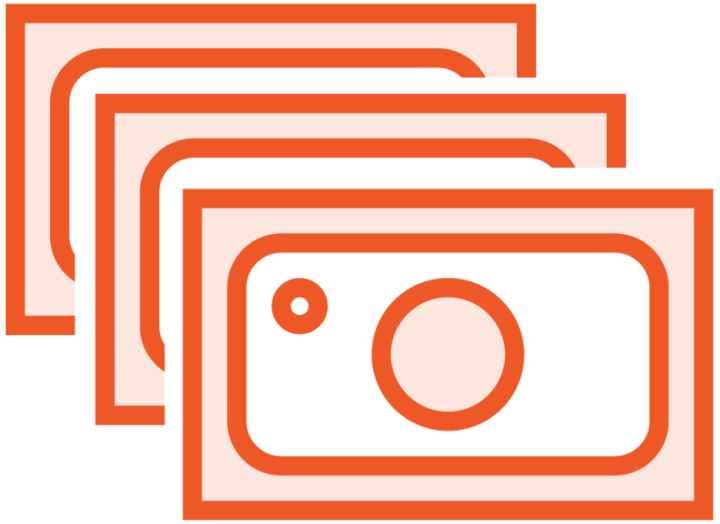
Choose reliable compensating transactions

Query for invariant violations

- Avoid false negatives
- Combine with monitoring

Reducing Contention

Pre-sales



Preferred status

Reduces simultaneous sales

- Reduces likelihood of conflicts

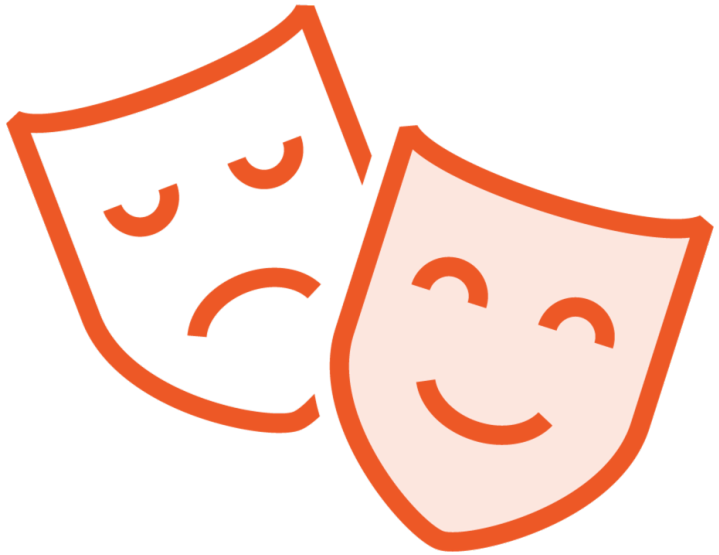
Limits pool of customers

- Increases likelihood of success

Rewards loyalty

Test markets

Season Tickets



Reserved seats

Fewer general customers

Higher chance of success

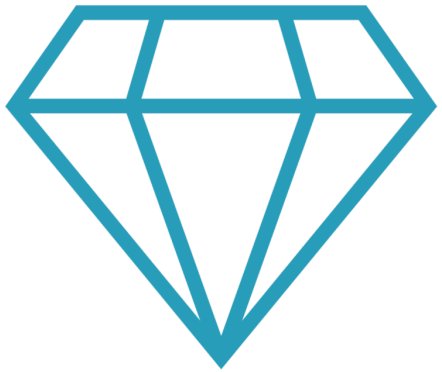
Lock Tickets Sooner

- Timer on web site
- Increases perceived scarcity
- Encourages customer registration
- Longer duration
- False sell-outs

Distributed Systems

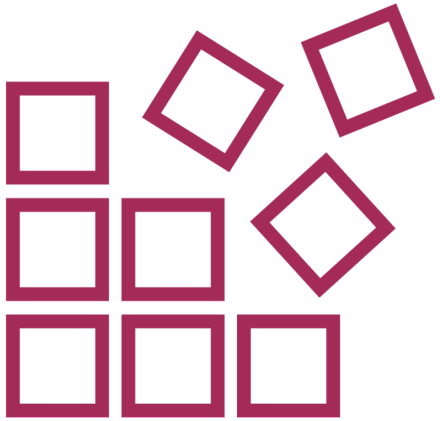
Hard to build
Hard to run
Don’t behave

Distributed Systems



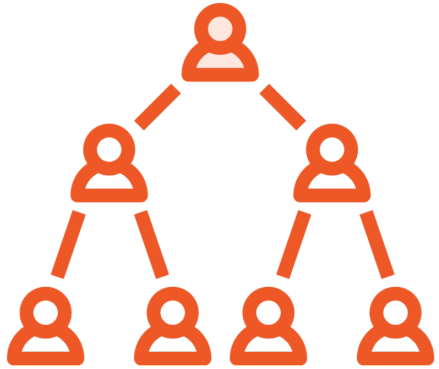
Make Reliable Applications

Immutable database
Hide internal ID
Additive structure



Bring Components Together

Commutative operations
Eventual consistency
Non-homogenous



Build Business Value

Align with business strategy
Build teams
Encourage culture