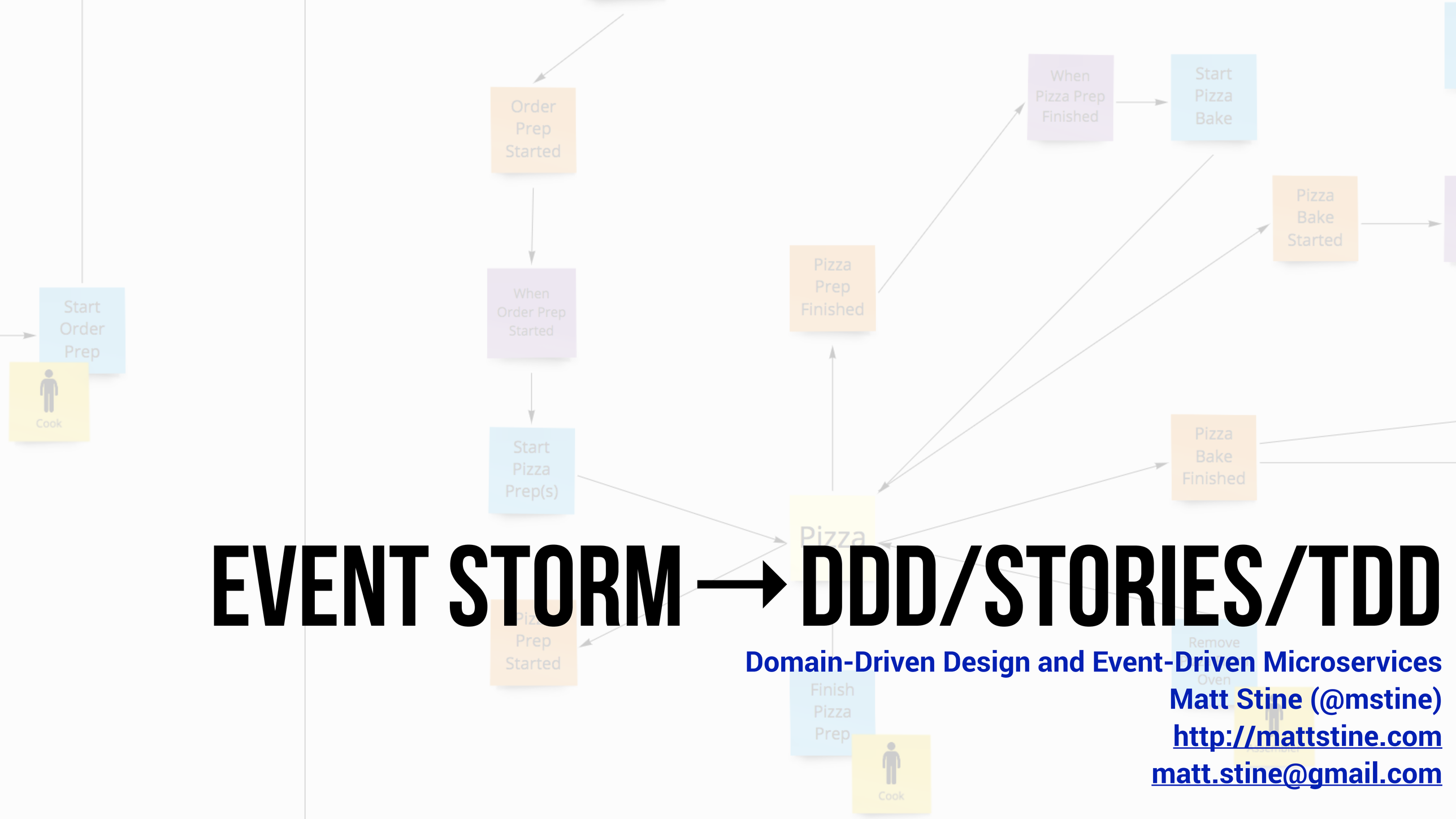


EVENT STORM → DDD/STORIES/TDD

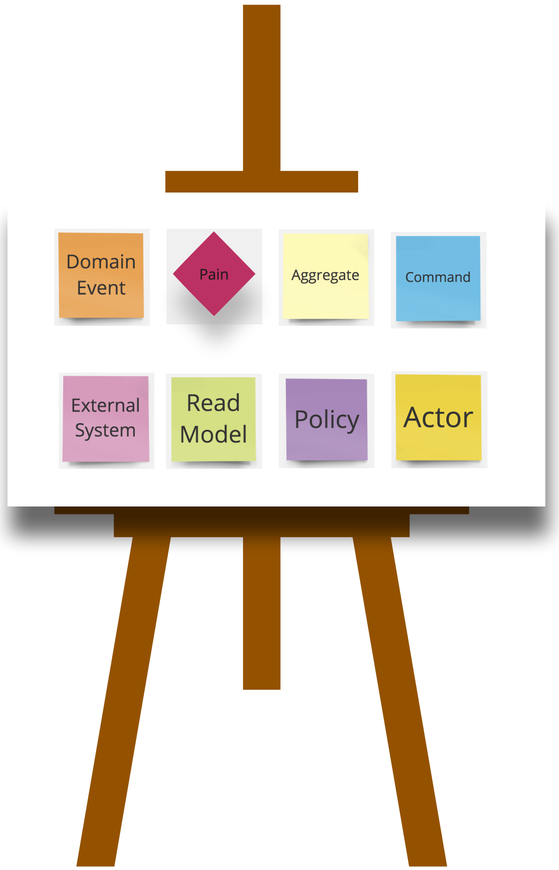
Domain-Driven Design and Event-Driven Microservices

Matt Stine (@mstine)
http://mattstine.com
matt.stine@gmail.com

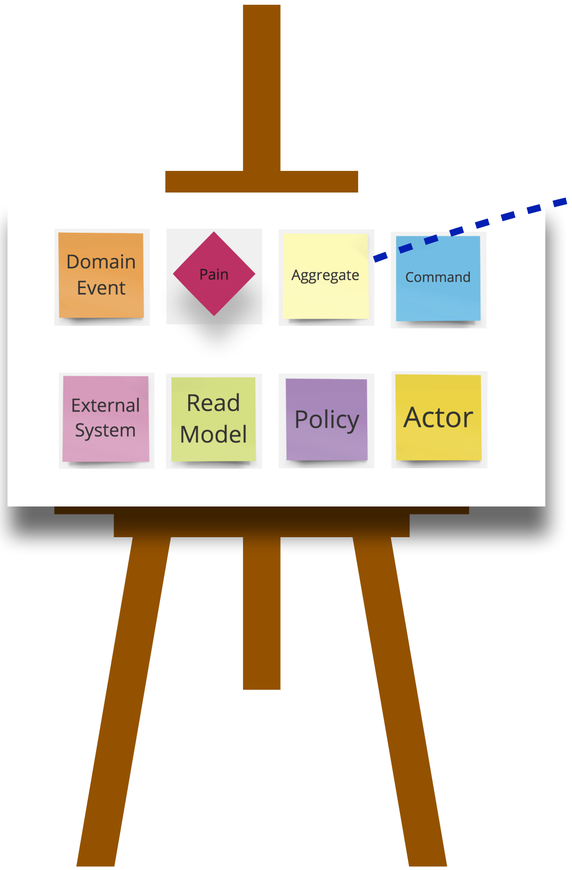
AGENDA

- From Event Storm to DDD Concepts
- From Event Storm to User Stories
- From Event Storm to TDD Tests
- From Event Storm to Code

FROM EVENT STORM TO DDD

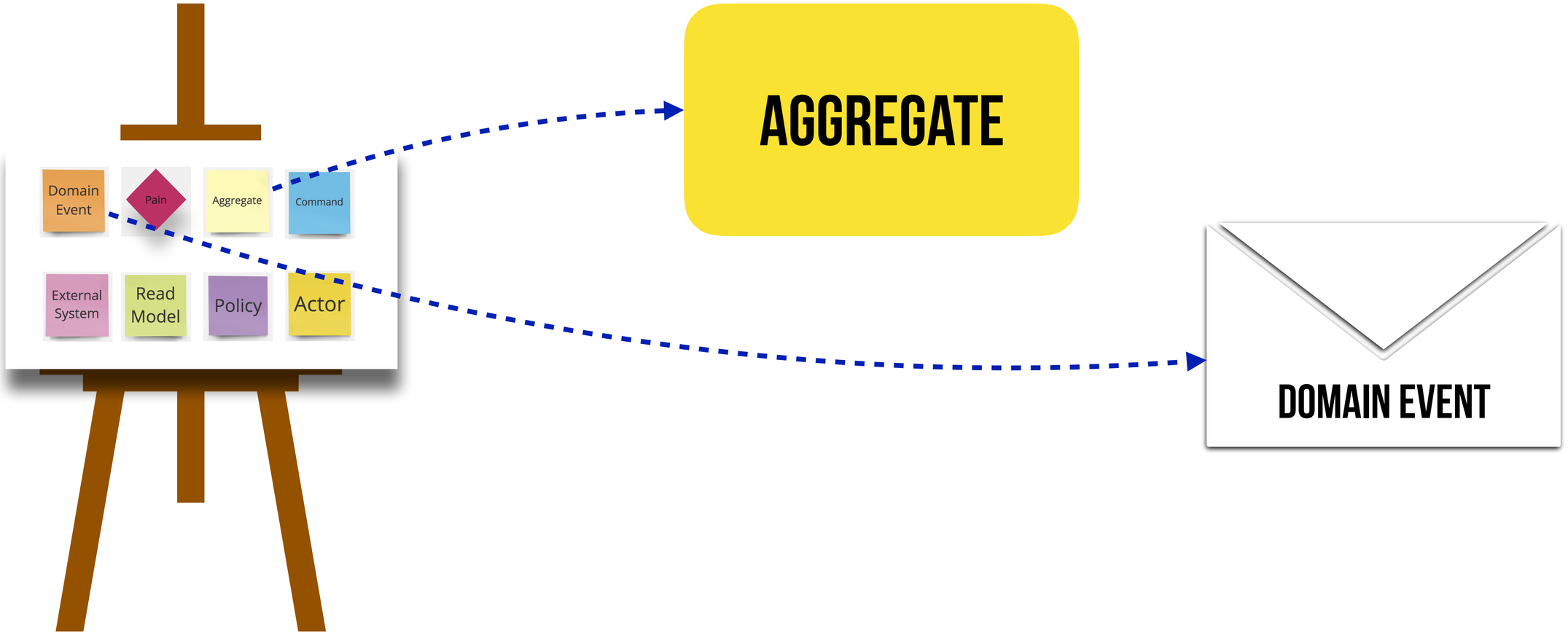


FROM EVENT STORM TO DDD

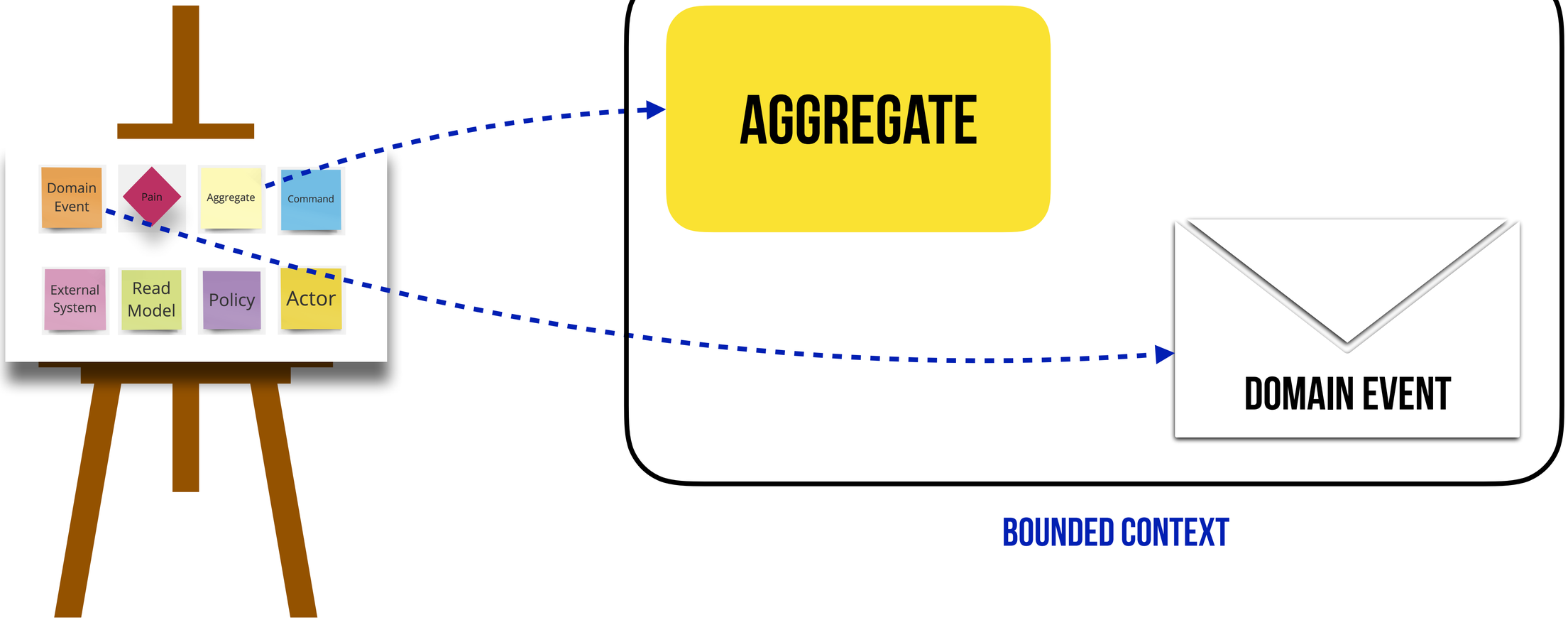


AGGREGATE

FROM EVENT STORM TO DDD



FROM EVENT STORM TO DDD



USER STORIES

USER STORIES

As a ROLE

USER STORIES

As a ROLE

I want to ACTION

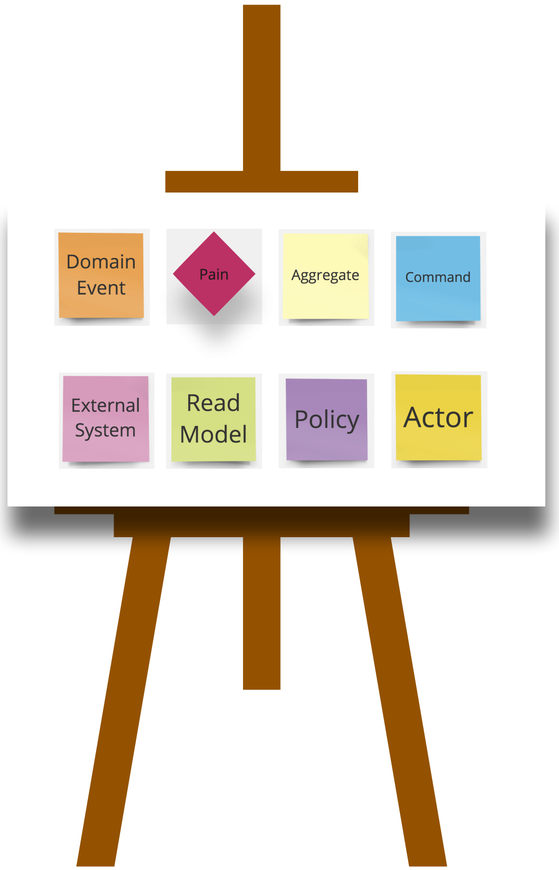
USER STORIES

As a ROLE

I want to ACTION

So that I can GOAL

USER STORIES

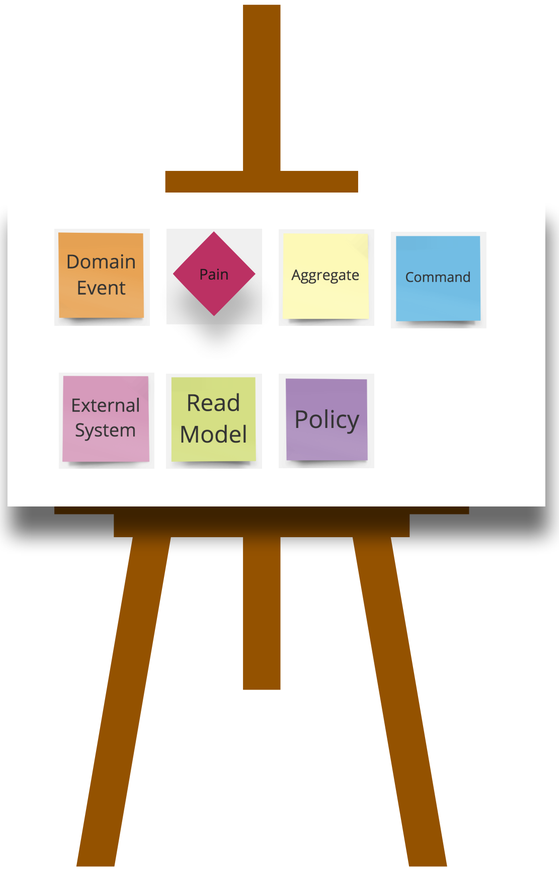


As a ROLE

I want to ACTION

So that I can GOAL

USER STORIES

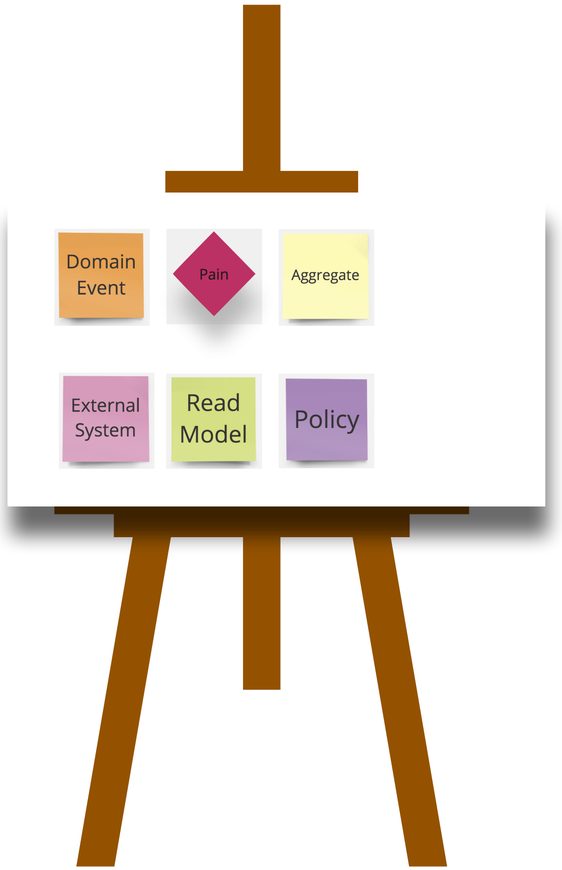


As a ROLE Actor

I want to ACTION

So that I can GOAL

USER STORIES

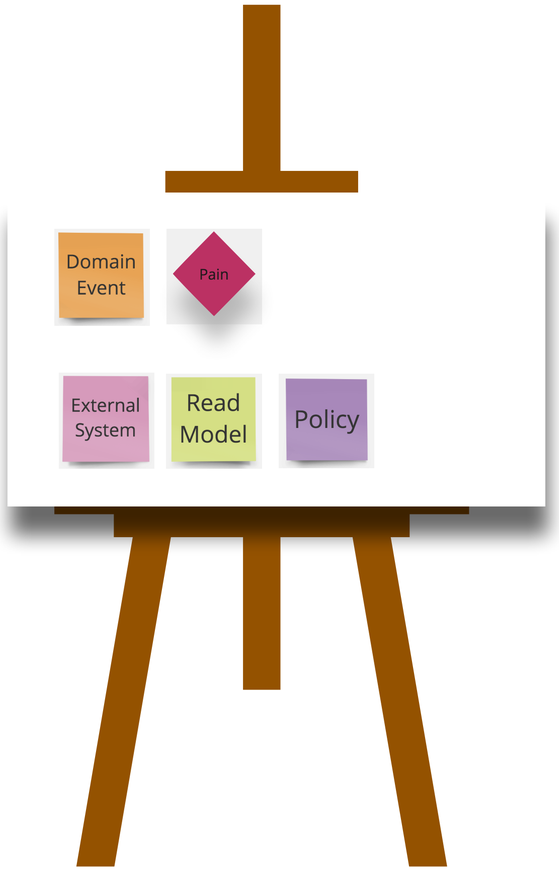


As a ROLE Actor

I want to ACTION Command

So that I can GOAL

USER STORIES

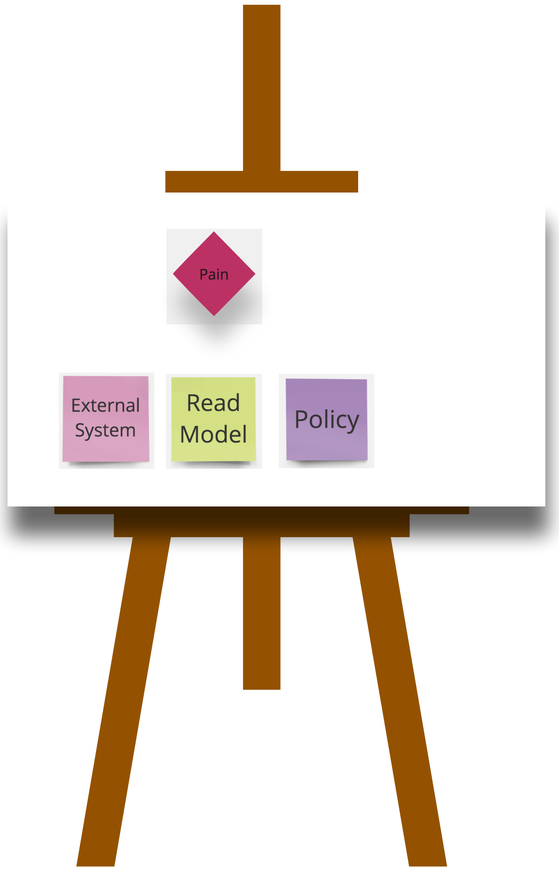


As a ROLE Actor

I want to ACTION Command Aggregate

So that I can GOAL

USER STORIES

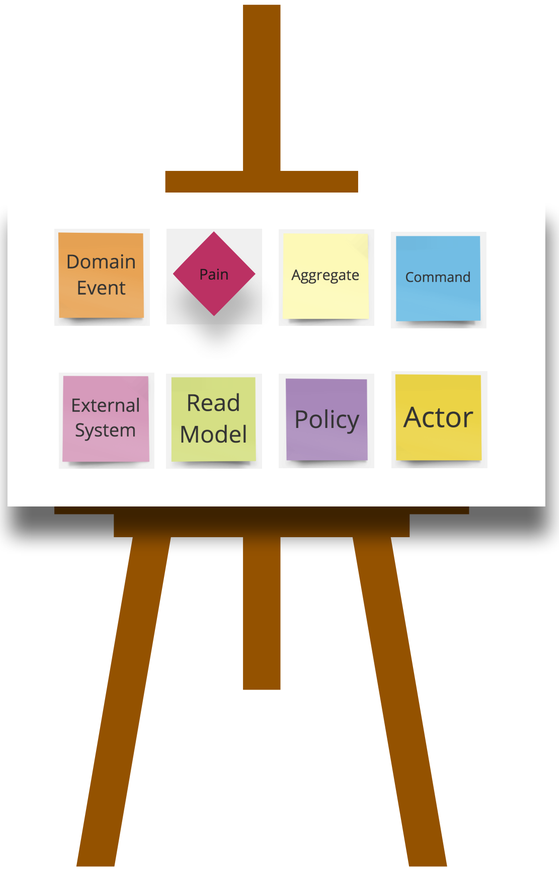


As a ROLE Actor

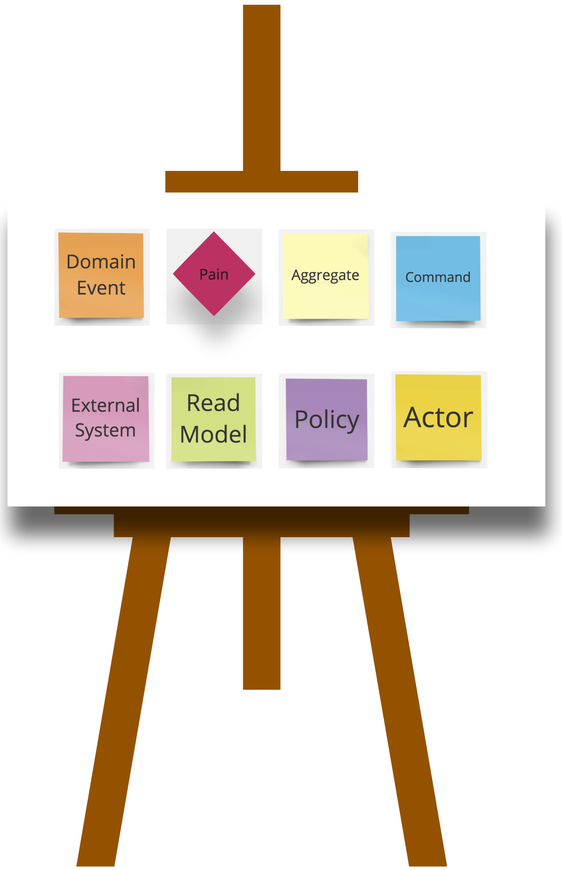
I want to ACTION Command Aggregate

So that I can GOAL Domain Event

WHAT ABOUT TESTS?

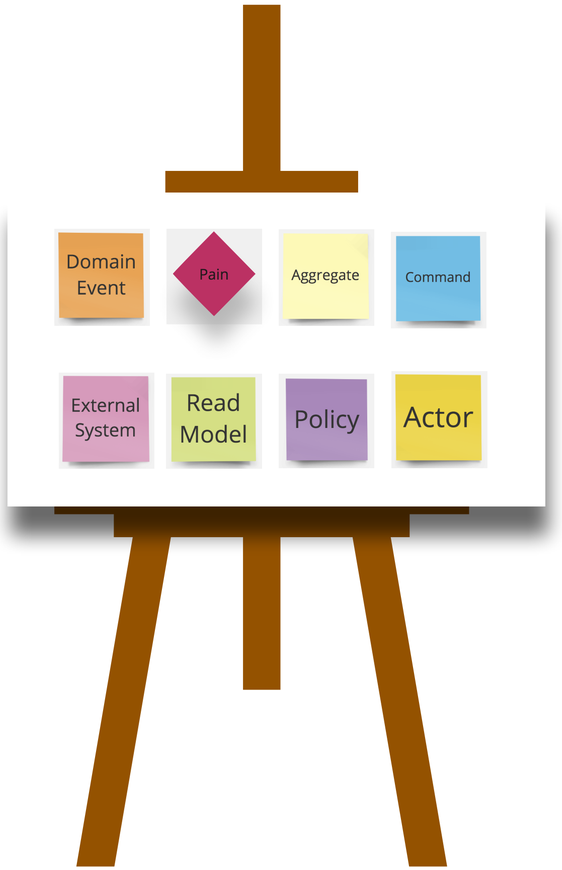


WHAT ABOUT TESTS?



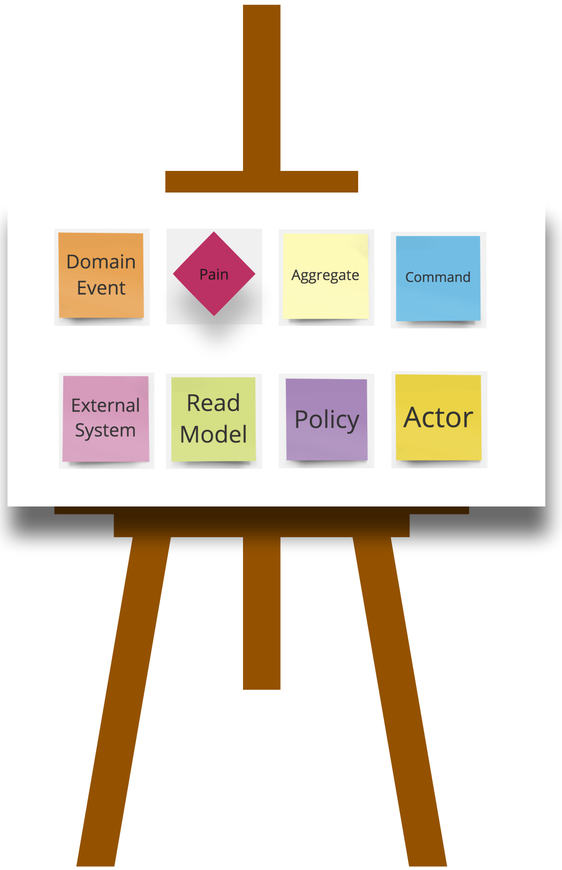
- Aggregate Business Invariants:
  `should_not_allow_placeOrder_without_a_lineItem()`

WHAT ABOUT TESTS?



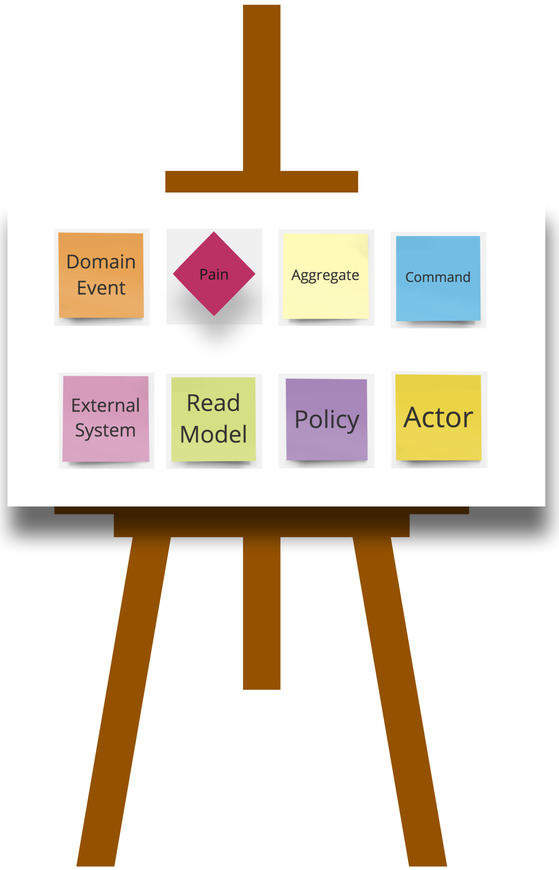
- Aggregate Business Invariants:
  `should_not_allow_placeOrder_without_a_lineItem()`
- Aggregate Domain Events:
  `should_fire_event_when_order_placed()`

WHAT ABOUT TESTS?

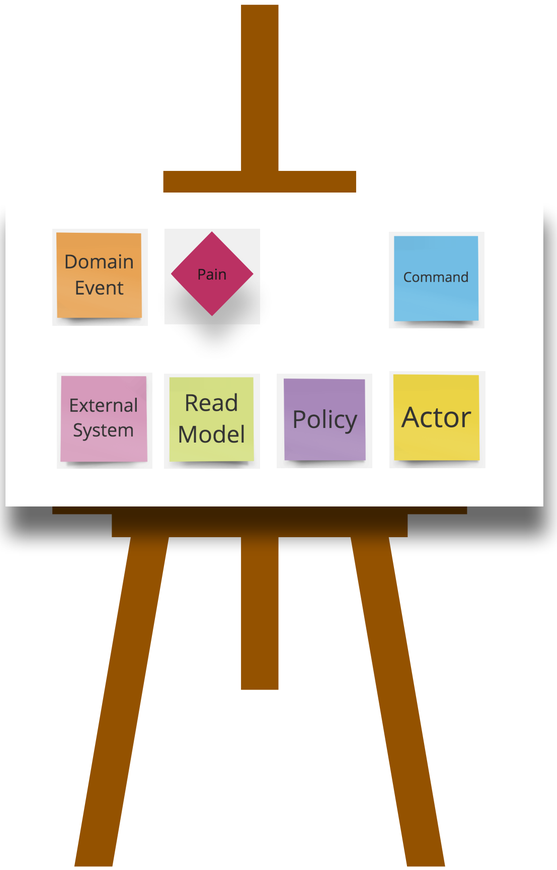


- Aggregate Business Invariants:
  `should_not_allow_placeOrder_without_a_lineItem()`
- Aggregate Domain Events:
  `should_fire_event_when_order_placed()`
- Policies:
  `should_check_inventory_when_order_placed()`

WHAT ABOUT CODE?

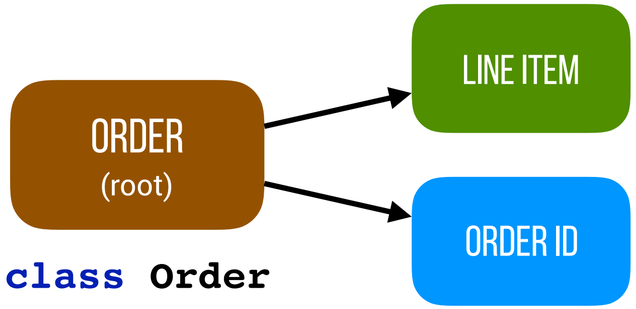


WHAT ABOUT CODE?



Aggregate

```
class OrderLineItem
```



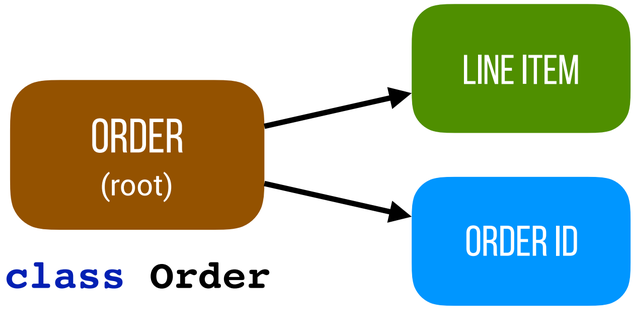

```
class Order
```

```
class OrderId
```

WHAT ABOUT CODE?

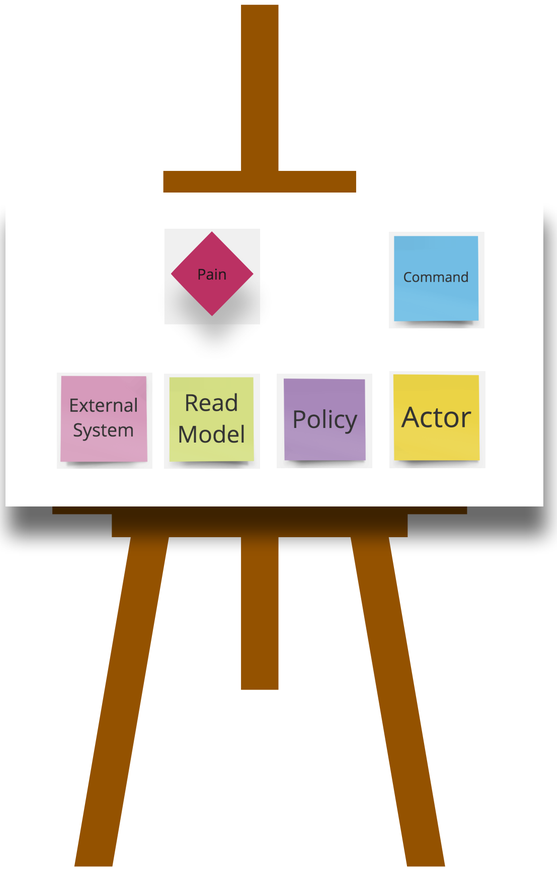
```
class OrderLineItem
```

Aggregate




```
class Order
```

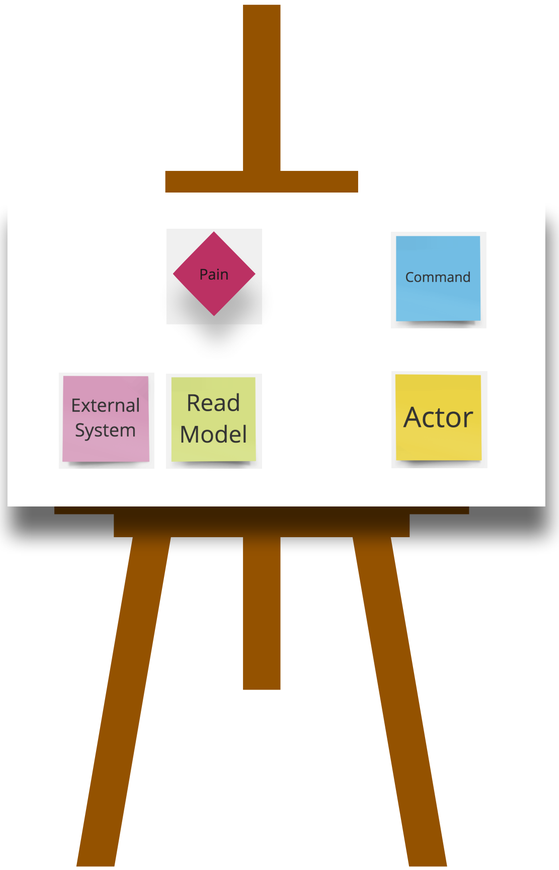
```
class OrderId
```



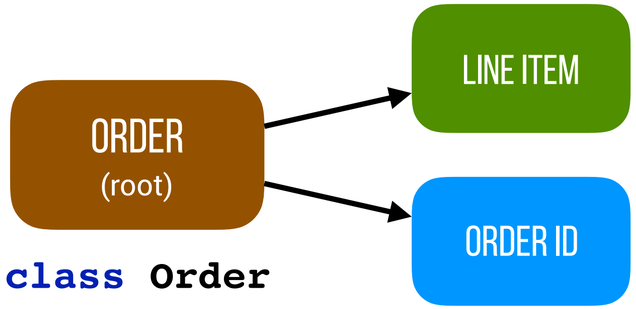
Domain Event


```
class OrderPlacedEvent
```

WHAT ABOUT CODE?



Aggregate




```
class OrderLineItem
```

```
class Order
```

```
class OrderId
```

Policy

APPLICATION SERVICE

```
eventLog.subscribe(OrderPlacedEvent,
e -> {});
```

Domain Event

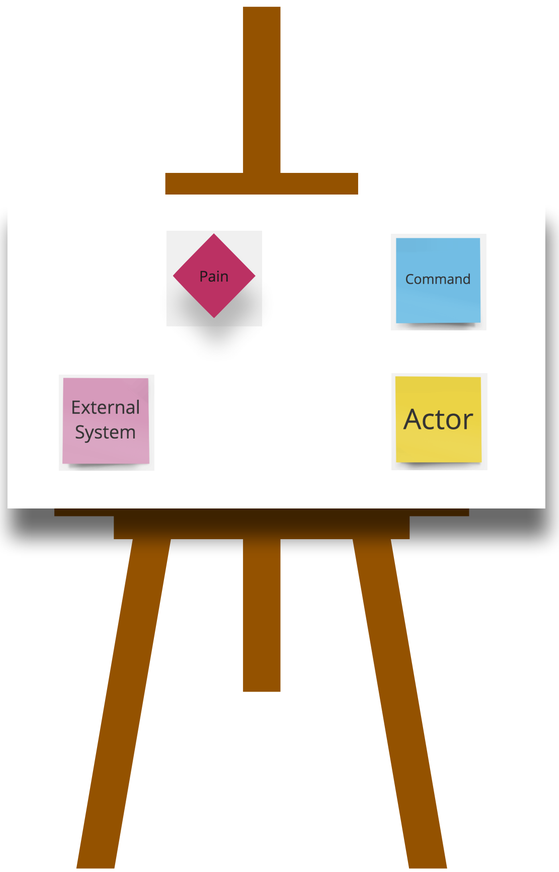

```
class OrderPlacedEvent
```

WHAT ABOUT CODE?

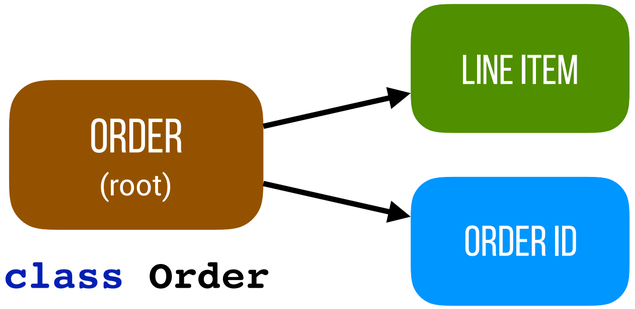
REPOSITORY

```
eventLog.subscribe(ProductAddedEvent,
e -> {});
```

Read Model



Aggregate




```
class OrderLineItem
```

```
class Order
```

```
class OrderId
```

Policy

APPLICATION SERVICE

```
eventLog.subscribe(OrderPlacedEvent,
e -> {});
```

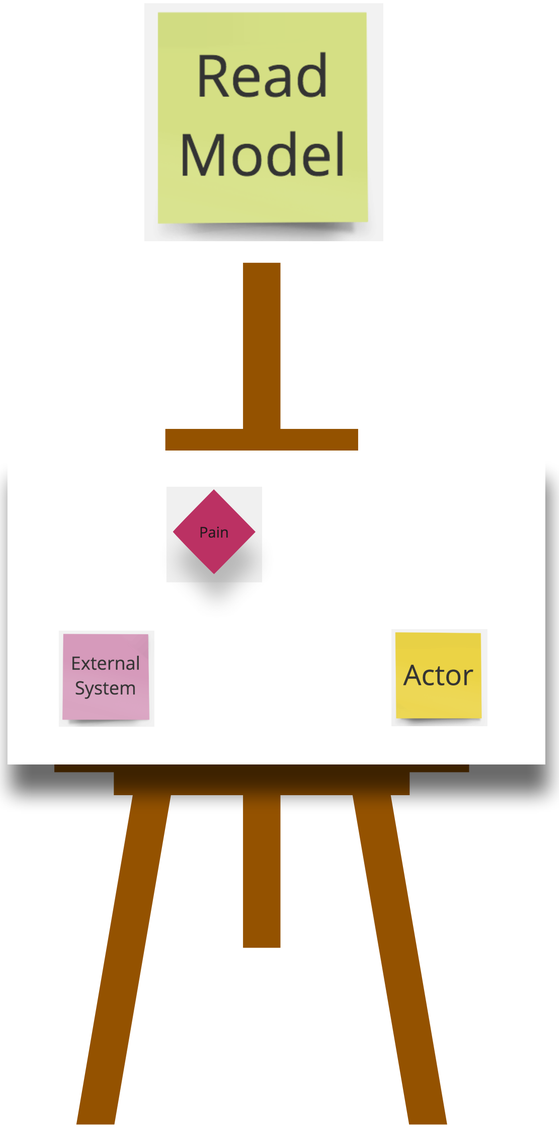
Domain Event


```
class OrderPlacedEvent
```

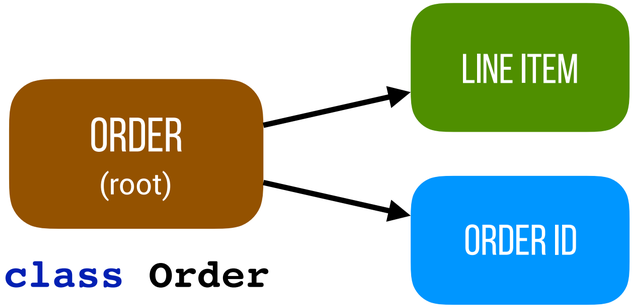
WHAT ABOUT CODE?

REPOSITORY

```
eventLog.subscribe(ProductAddedEvent,
e -> {});
```



Aggregate




```
class OrderLineItem
```

```
class Order
```

```
class OrderId
```

Command

```
void placeOrder()
```

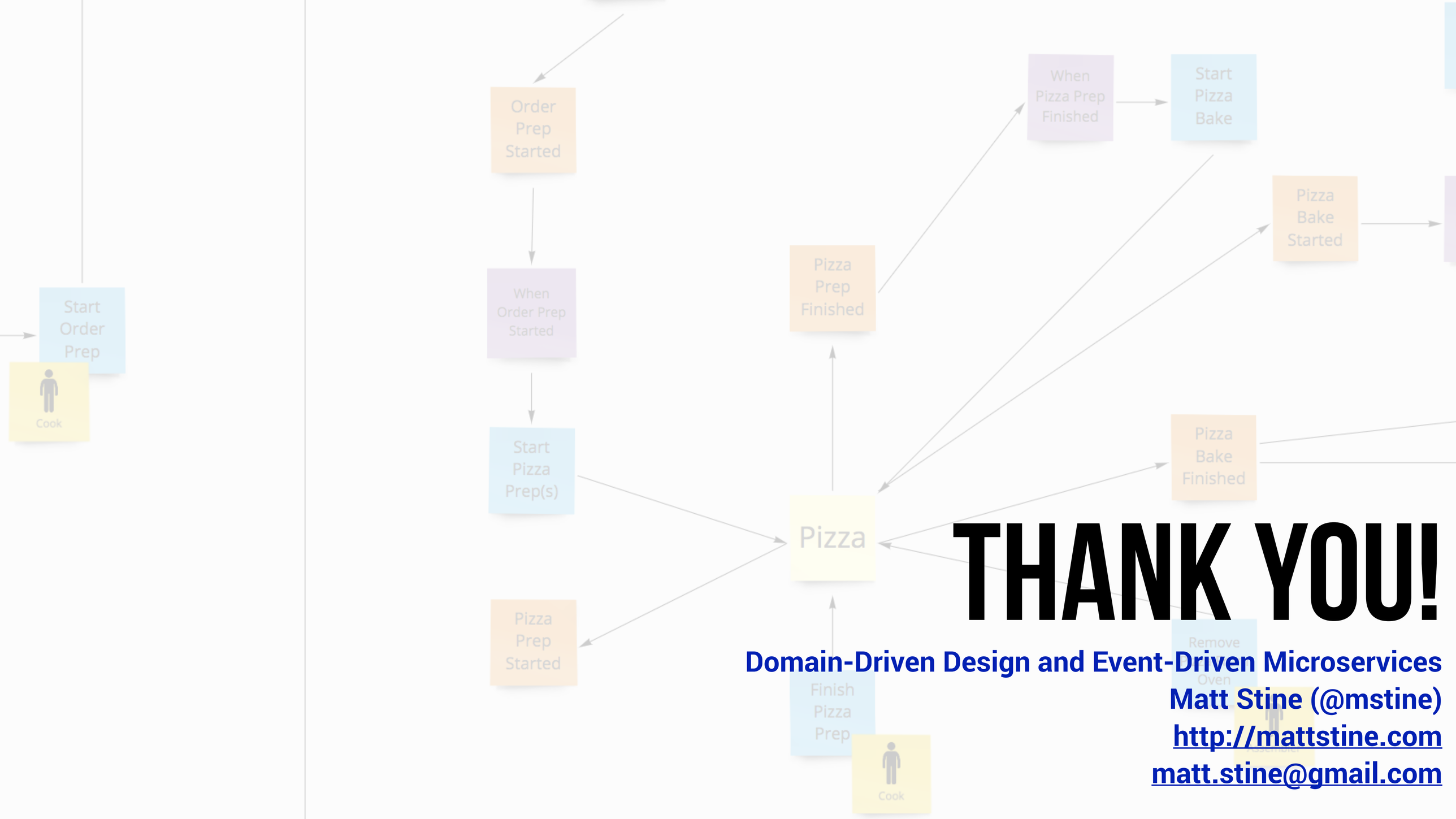
Policy

APPLICATION SERVICE

```
eventLog.subscribe(OrderPlacedEvent,
e -> {});
```

Domain Event


```
class OrderPlacedEvent
```



THANK YOU!

Domain-Driven Design and Event-Driven Microservices

Matt Stine (@mstine)

http://mattstine.com

matt.stine@gmail.com