Working with Domain Events

Vladimir Khorikov

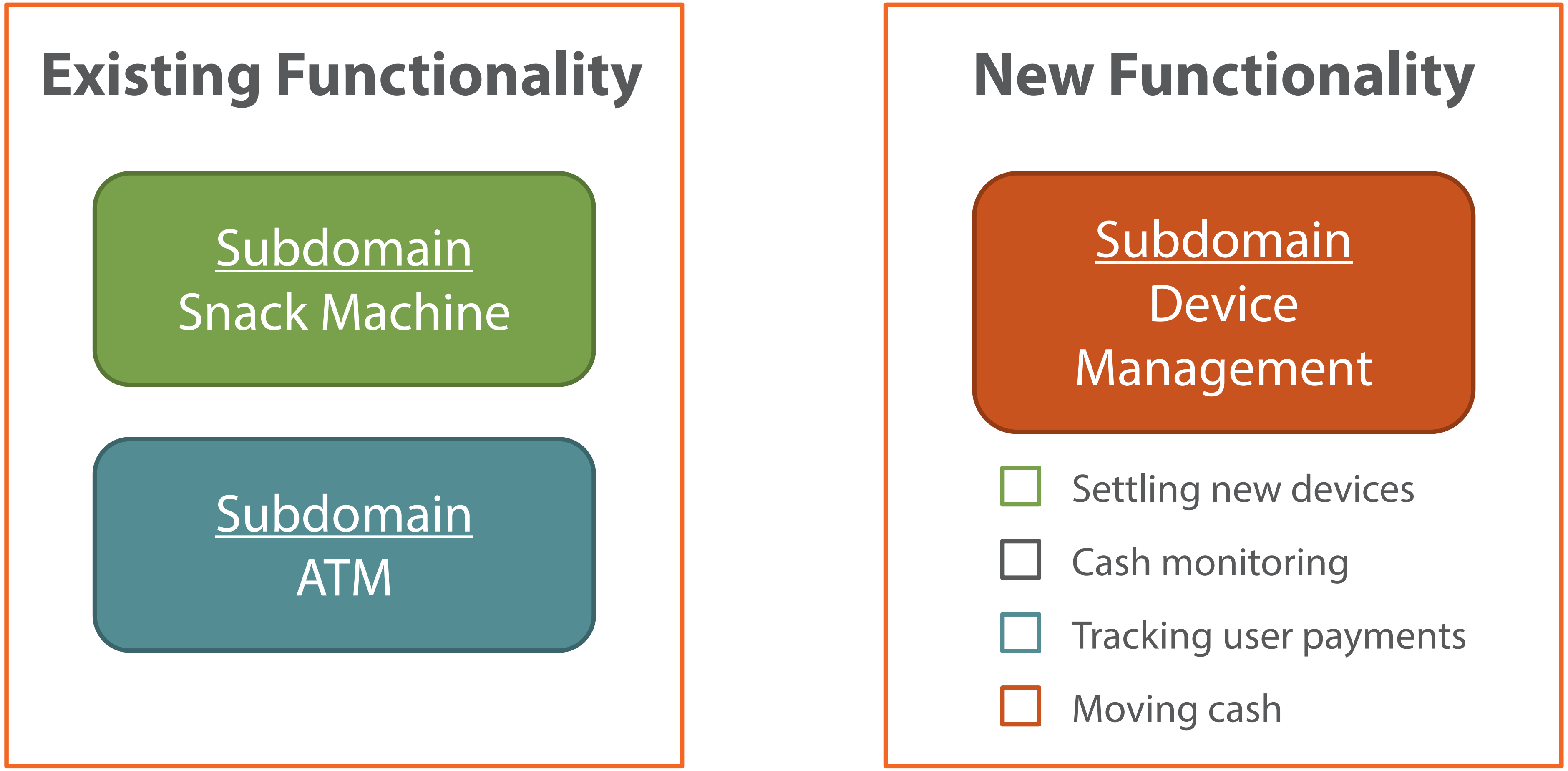
@vkhorikov | www.enterprisecraftsmanship.com

In This Module

Domain Events

Two ways to handle domain events in code

New Requirements



Existing Functionality

Subdomain
Snack Machine

Subdomain
ATM

New Functionality

Subdomain
Device
Management

- □ Settling new devices
- □ Cash monitoring
- □ Tracking user payments
- □ Moving cash

New Requirements

Total amount of all users payments

New Requirements

Cash producer

$$

Cash consumer

Introducing a New Bounded Context

New sub-domain

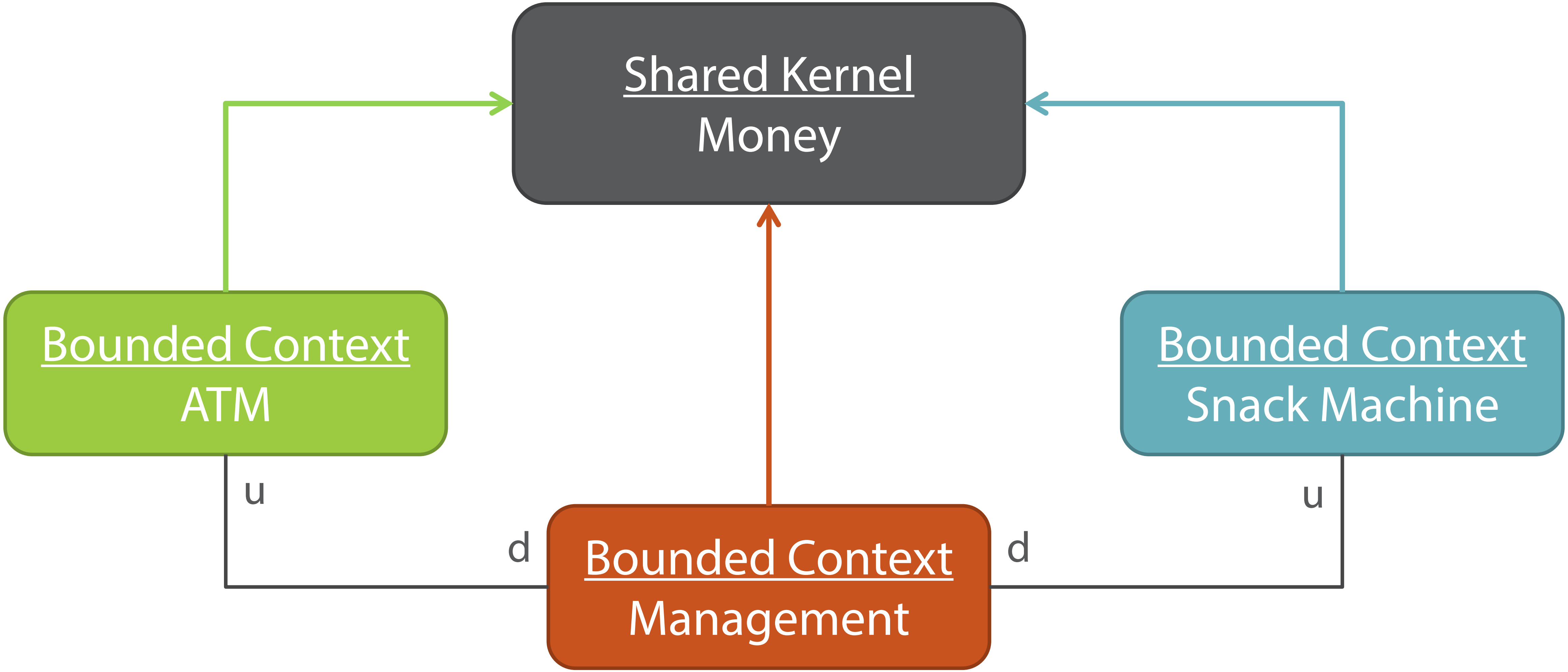
Management

New bounded context

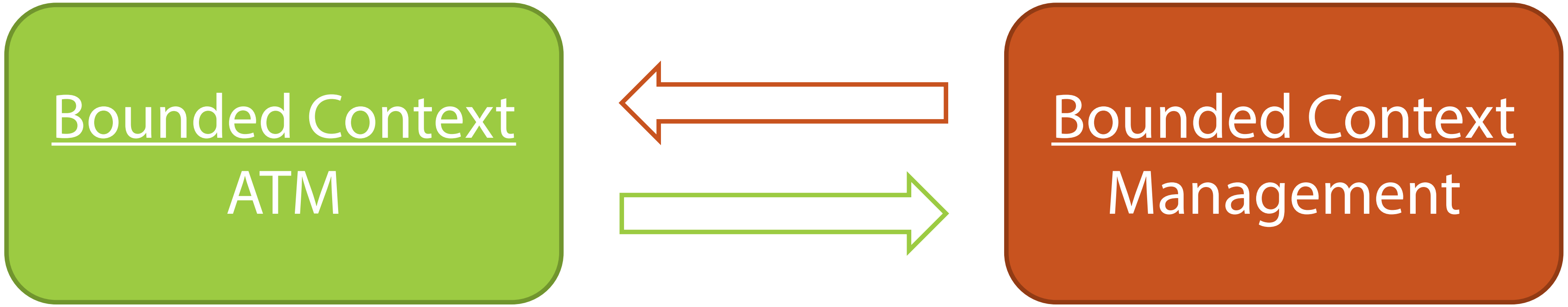
Management

- Where the payments go?
- How cash flows from snack machines to ATMs?

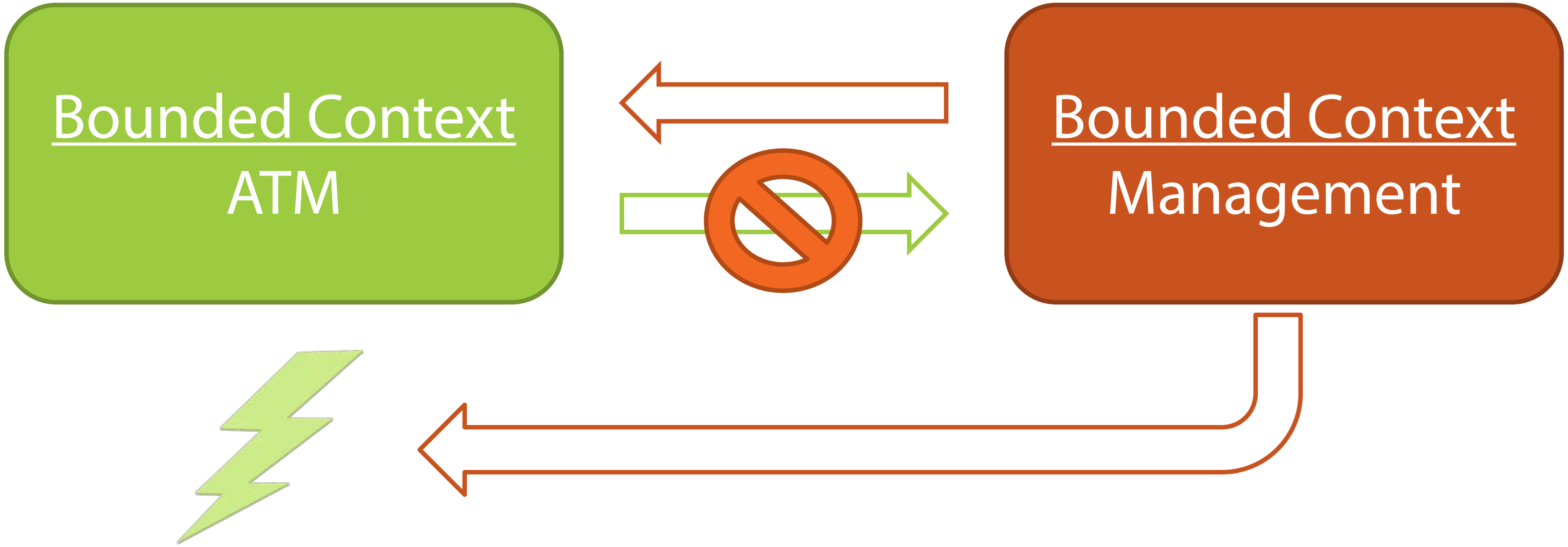
Introducing a New Bounded Context



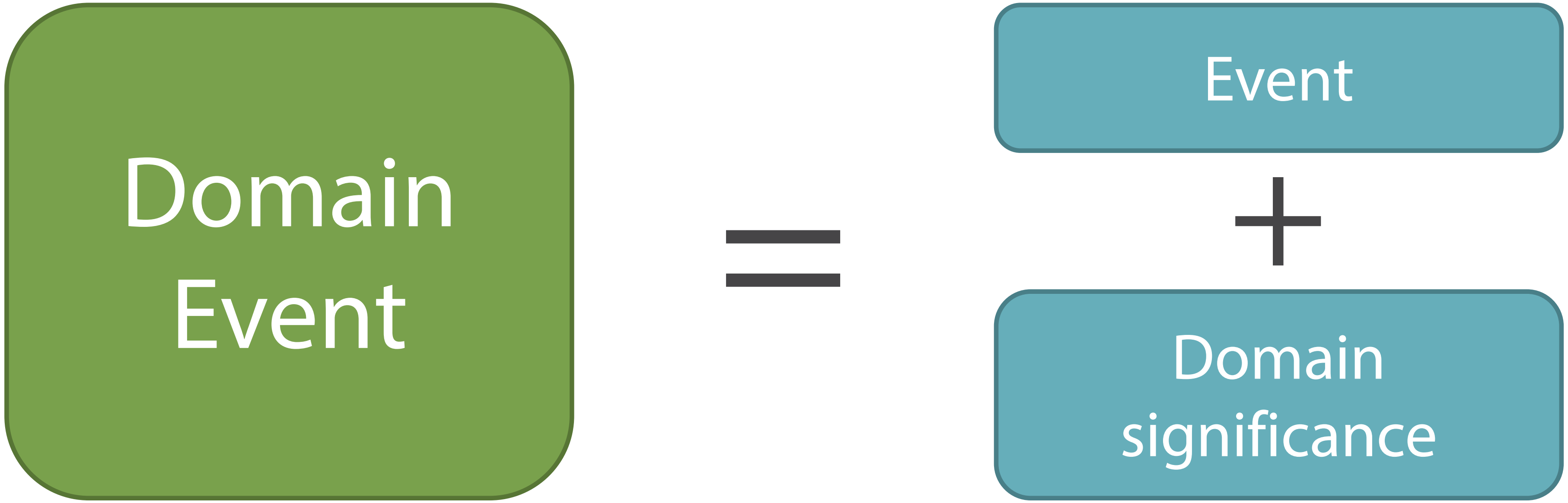
Implementation: The First Attempt



Implementation: The First Attempt



Domain Events



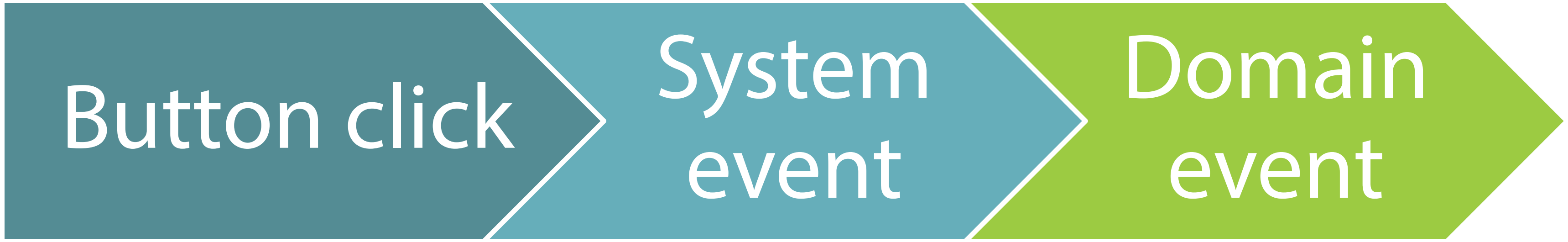
Domain Events

Domain Event

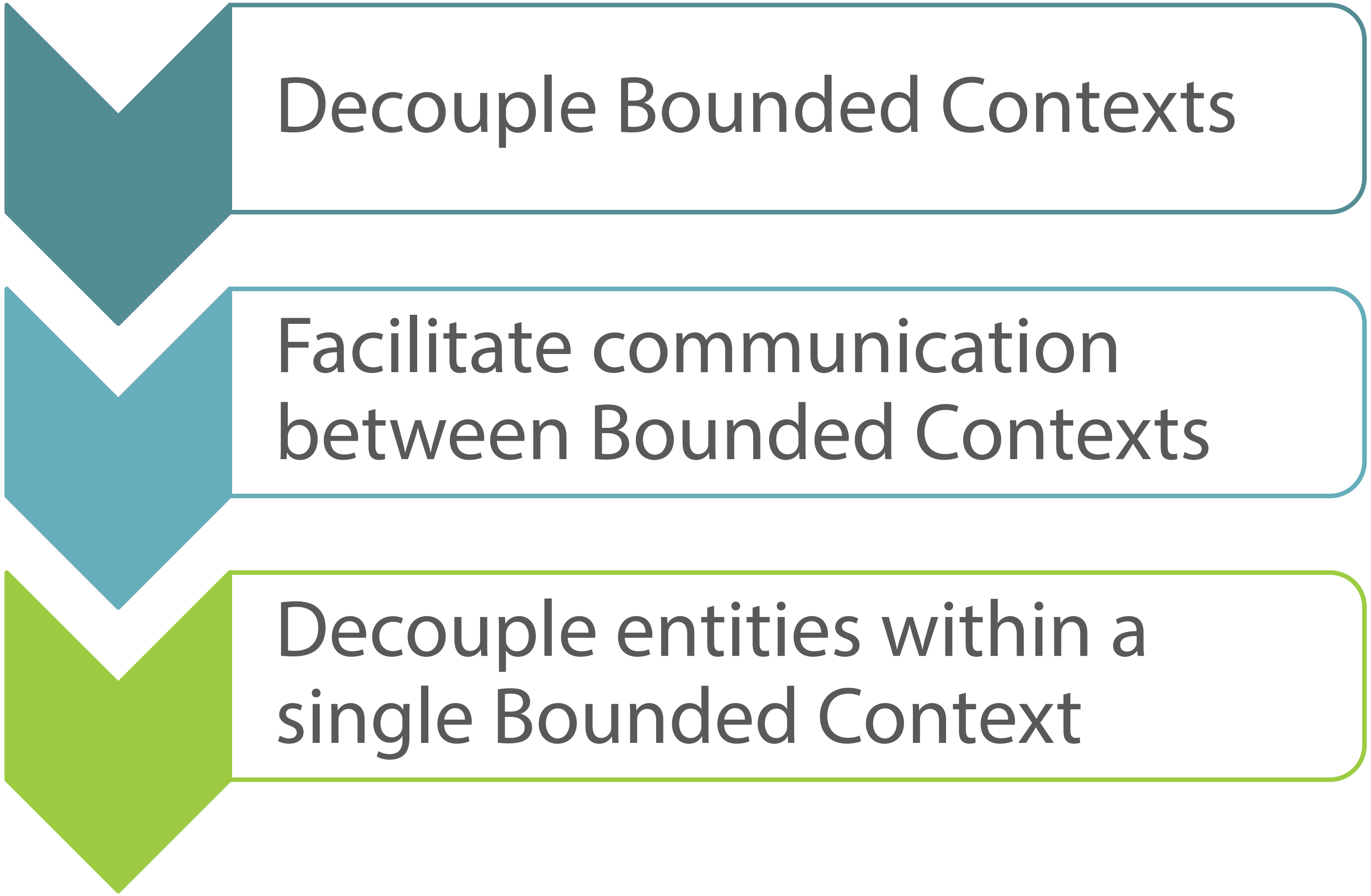
- □ Important for the domain

System Event

- □ Infrastructure



Domain Events



Decouple Bounded Contexts

Facilitate communication between Bounded Contexts

Decouple entities within a single Bounded Context

Introducing a Domain Event

Domain event definition:

```
public class BalanceChangedEvent {
    public decimal Delta { get; private set; }

    public BalanceChangedEvent(decimal delta) {
        Delta = delta;
    }
}
```

Domain event generation:

```
public virtual void TakeMoney(decimal amount) {
    if (CanTakeMoney(amount) != string.Empty)
        throw new InvalidOperationException();

    Money output = MoneyInside.Allocate(amount);
    MoneyInside -= output;

    decimal amountWithCommission = CaluculateAmountWithCommission(amount);
    MoneyCharged += amountWithCommission;

    var ev = new BalanceChangedEvent(amountWithCommission);
    // Raise the event
}
```

Domain Events Guidelines

Naming

- Past tense
- BalanceChangedEvent

Data

- Include as little data as possible

Domain Events Guidelines

What data structures should be used to represent data in domain events?

```
public class Person : Entity
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string MiddleName { get; set; }
}
```

```
public class PersonChangedEvent
{
    public Person Person { get; set; }

    public PersonChangedEvent(Person person)
    {
        Person = person;
    }
}
```

✗ Include more information than needed

✗ Additional point of coupling

Domain Events Guidelines

Naming	• Past tense • BalanceChangedEvent
Data	• Include as little data as possible
Domain classes	• Don’t use domain classes to represent data in events • Use primitive types instead

Domain Events Guidelines

Id of the entity or full information about it?

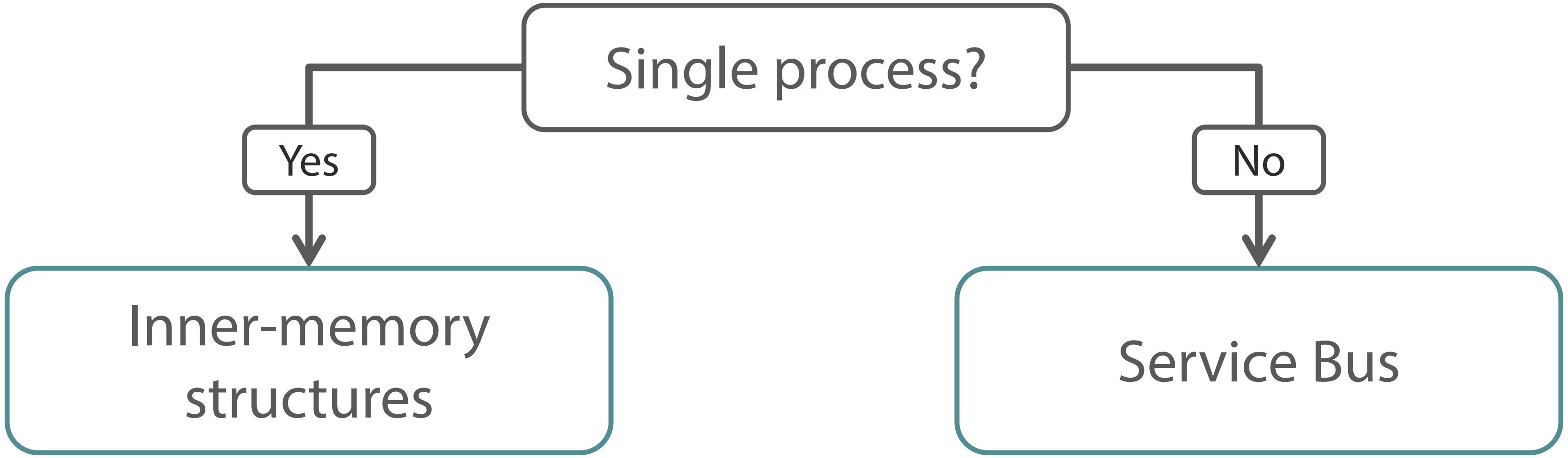
Ids

- When consuming BC knows about producing BC

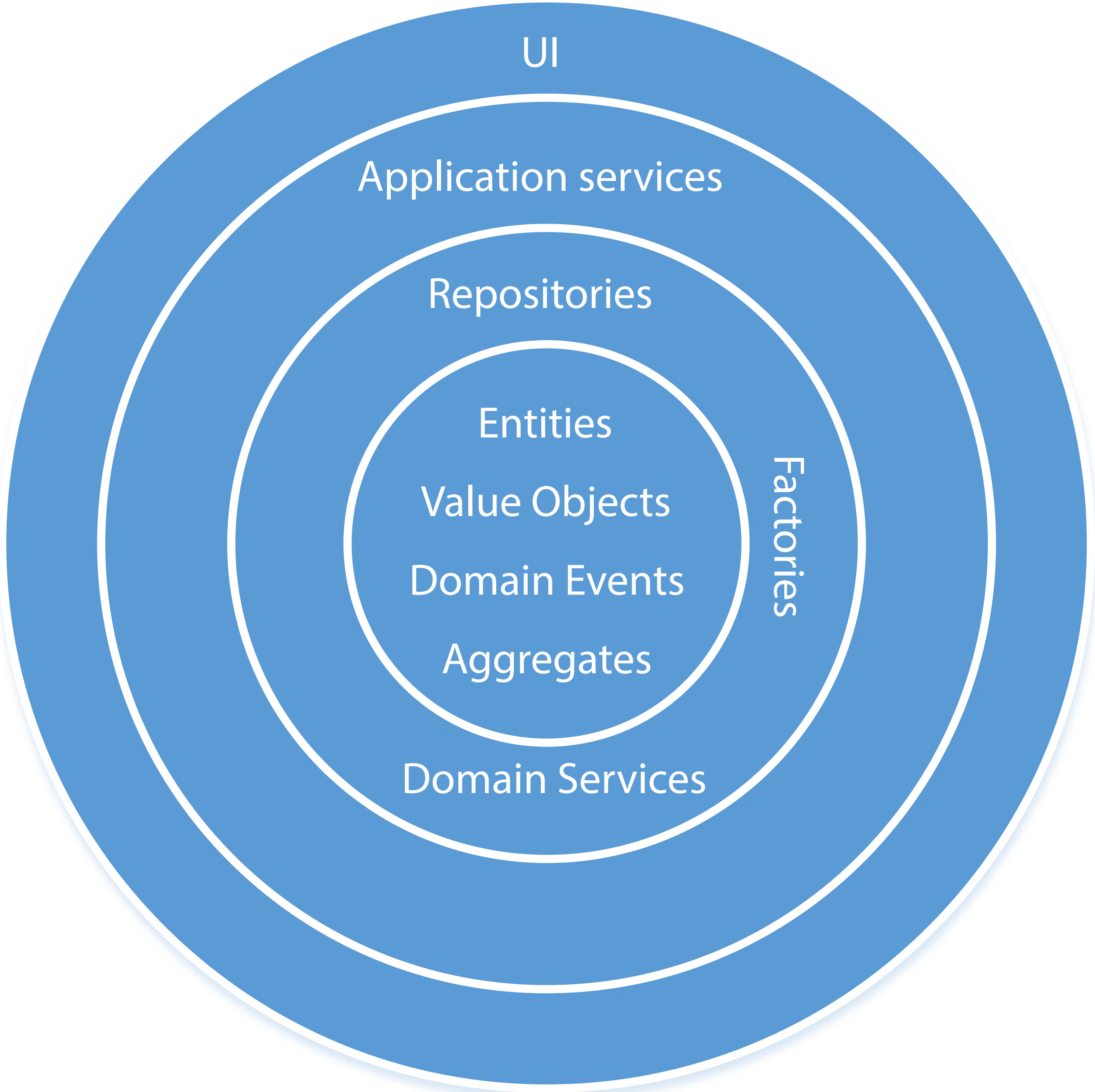
Full information

- When consuming BC doesn’t know about producing BC

Physical Delivery



Physical delivery is orthogonal to the notion of domain events



UI

Application services

Repositories

Entities

Value Objects

Domain Events

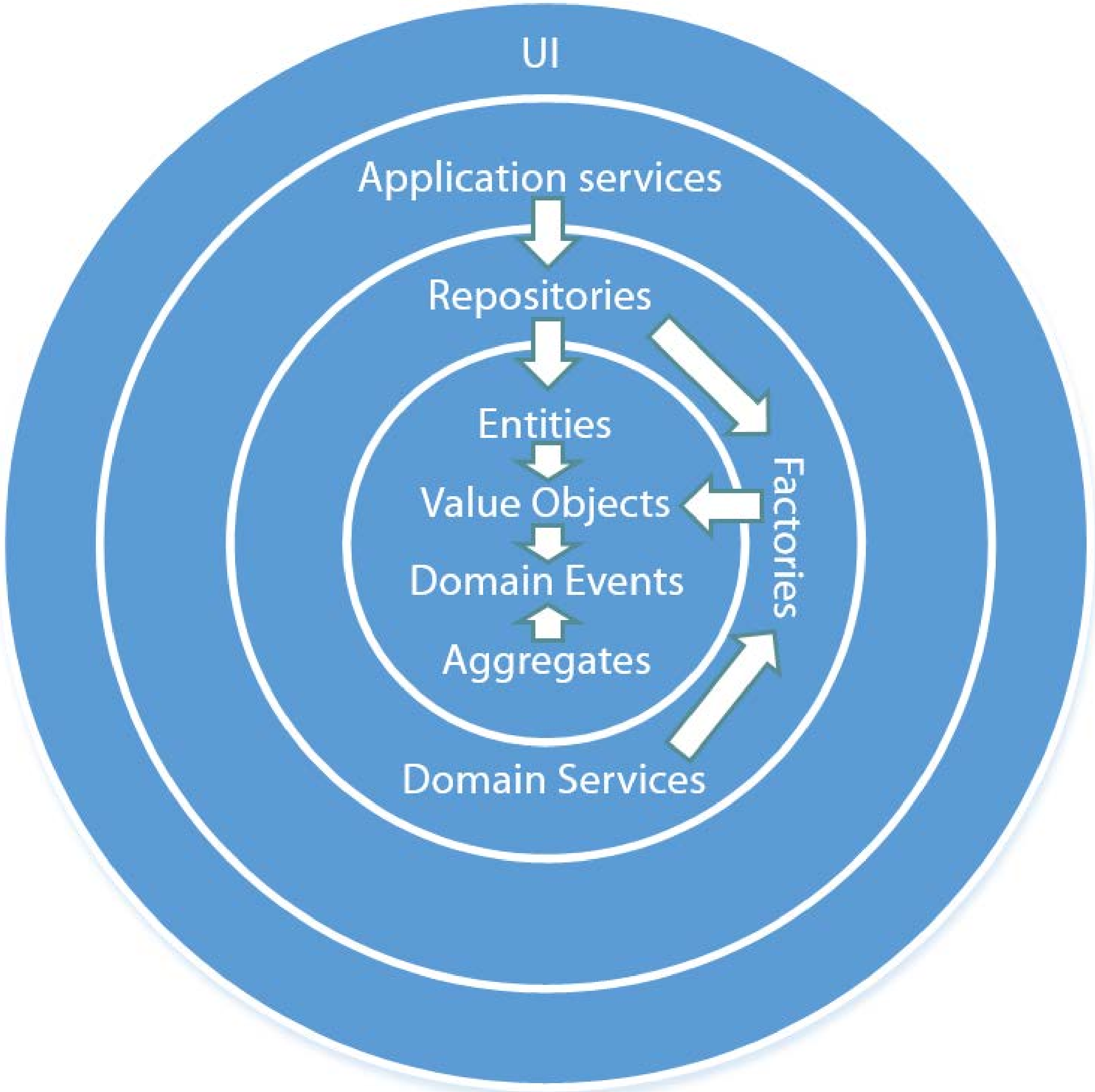
Aggregates

Factories

Domain Services

Recap: Classic Approach

Damages domain model isolation



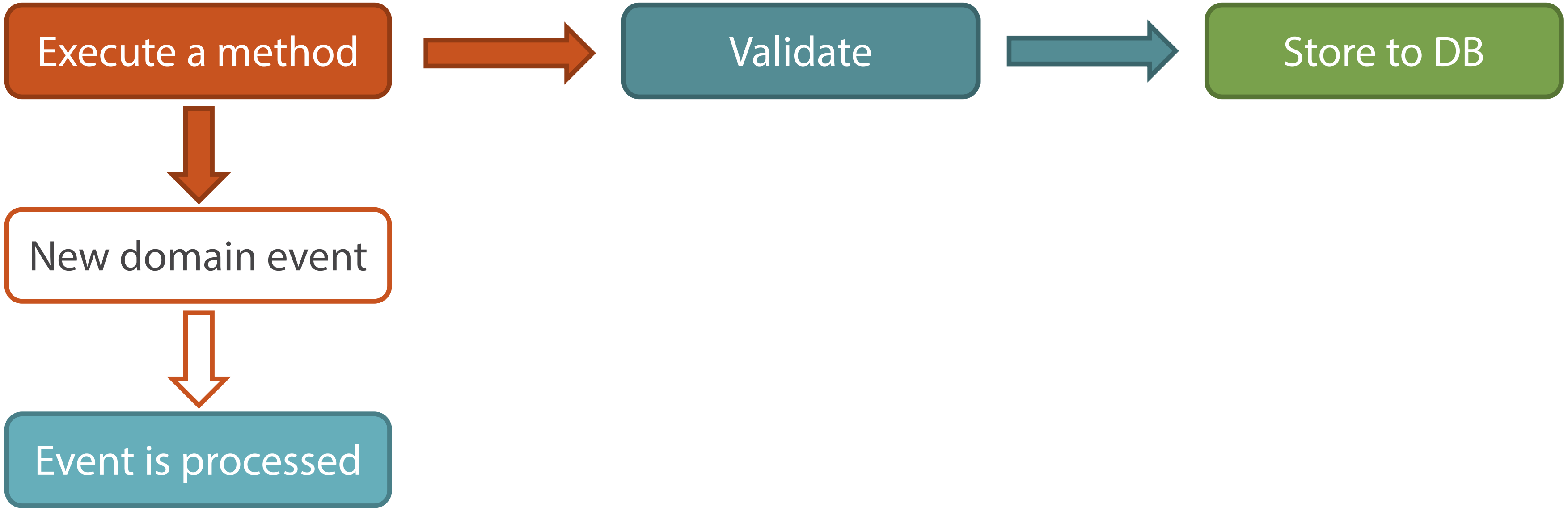
Recap: Classic Approach

```
[Fact]
public void Take_money_raises_an_event()
{
    Atm atm = new Atm();
    atm.LoadMoney(Dollar);
    BalanceChangedEvent balanceChangedEvent = null;
    DomainEvents.Register<BalanceChangedEvent>(ev => balanceChangedEvent = ev);

    atm.TakeMoney(1m);

    balanceChangedEvent.Should().NotBeNull();
    balanceChangedEvent.Delta.Should().Be(1.01m);
}
```

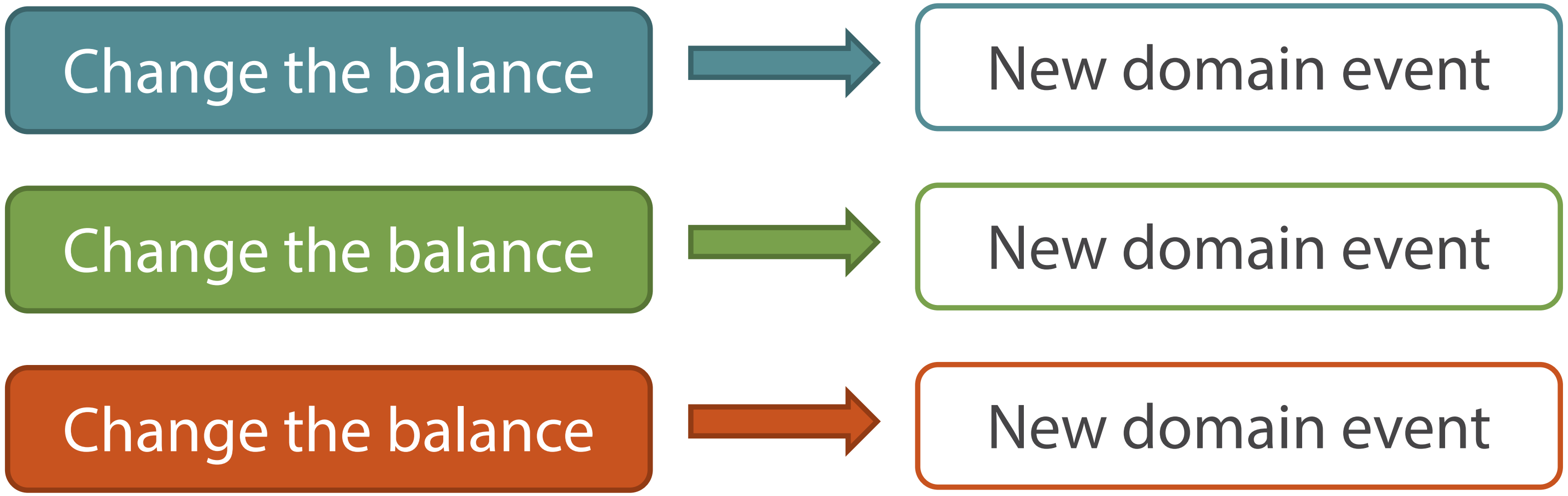
```
public static class DomainEvents
{
    private static Dictionary<Type, List<Delegate>> _dynamicHandlers;
    private static List<Type> _staticHandlers;
}
```

Recap: Classic Approach

Doesn't fit into the notion of Unit of Work



Recap: Classic Approach

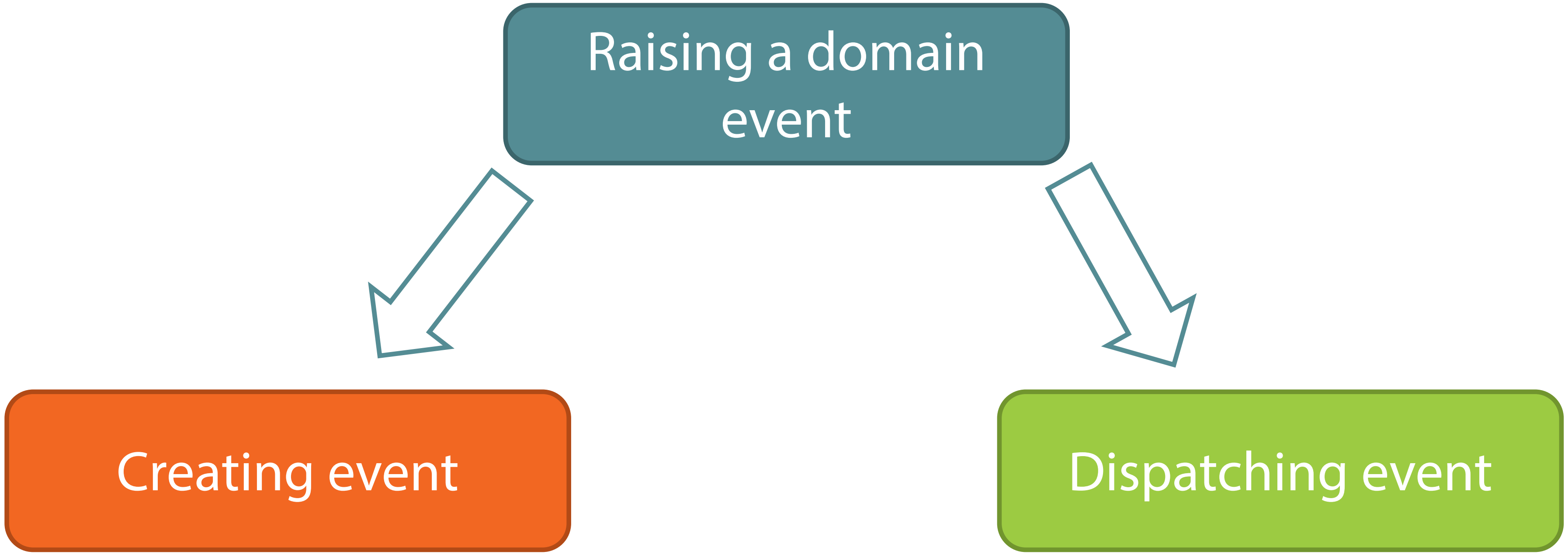


Change the balance → New domain event

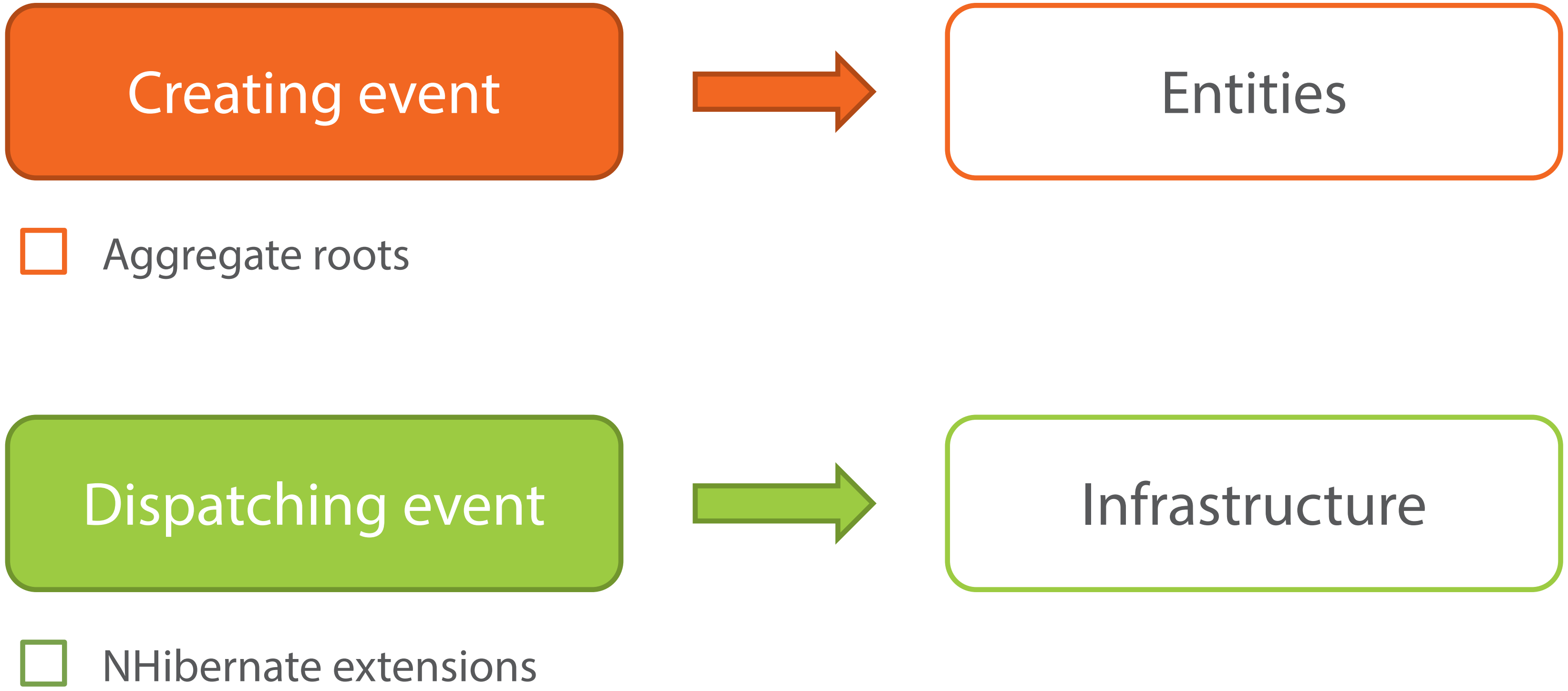
Change the balance → New domain event

Change the balance → New domain event

A Better Approach to Handling Domain Events



Recap: a Better Approach



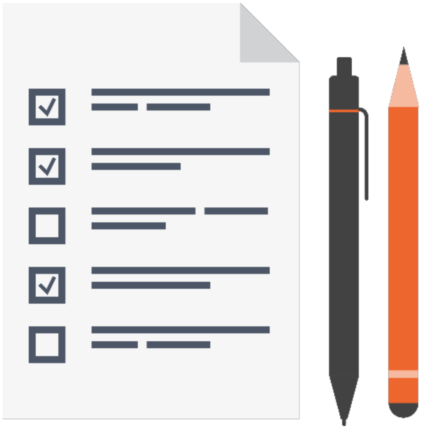
Creating event → Entities

□ Aggregate roots

Dispatching event → Infrastructure

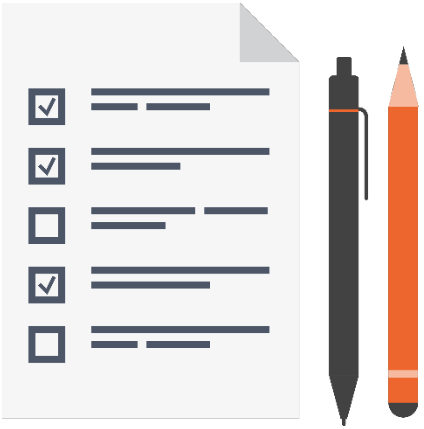
□ NHibernate extensions

Summary



- Domain Events
- Best practices for defining a Domain Event
  - Naming in the past tense
  - Include as little data as possible
  - Don’t include domain classes
  - Id vs full information
- Physical delivery of Domain Events is an orthogonal topic

Summary



- Two ways of handling domain events
  - Avoid use of the classic approach
- Working with lists of objects on the interface
  - Don’t use domain entities to display data on the screen
- Source code: http://bit.ly/1OxbGEA

In the Next Module

Other DDD concepts

Further enhancements