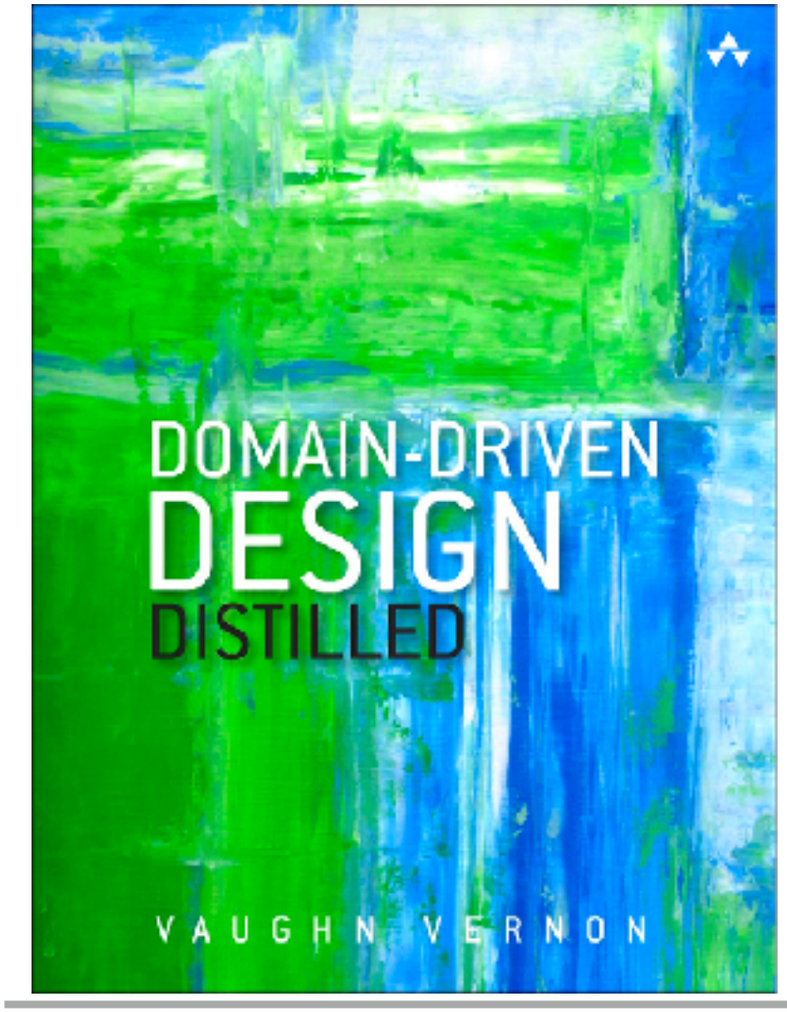
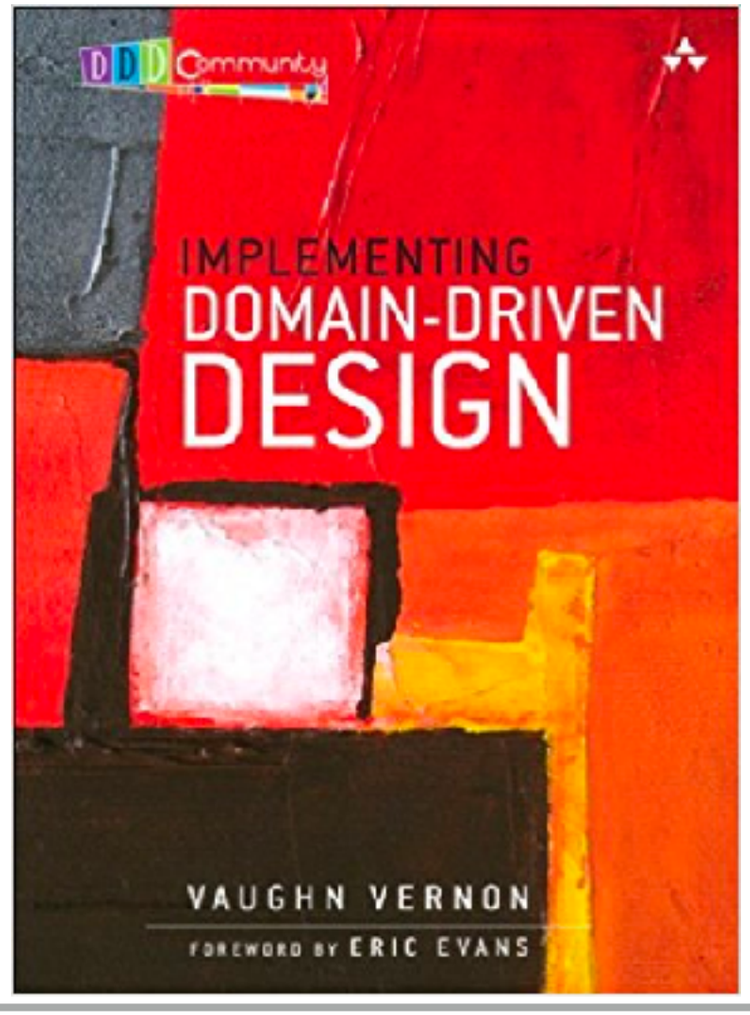
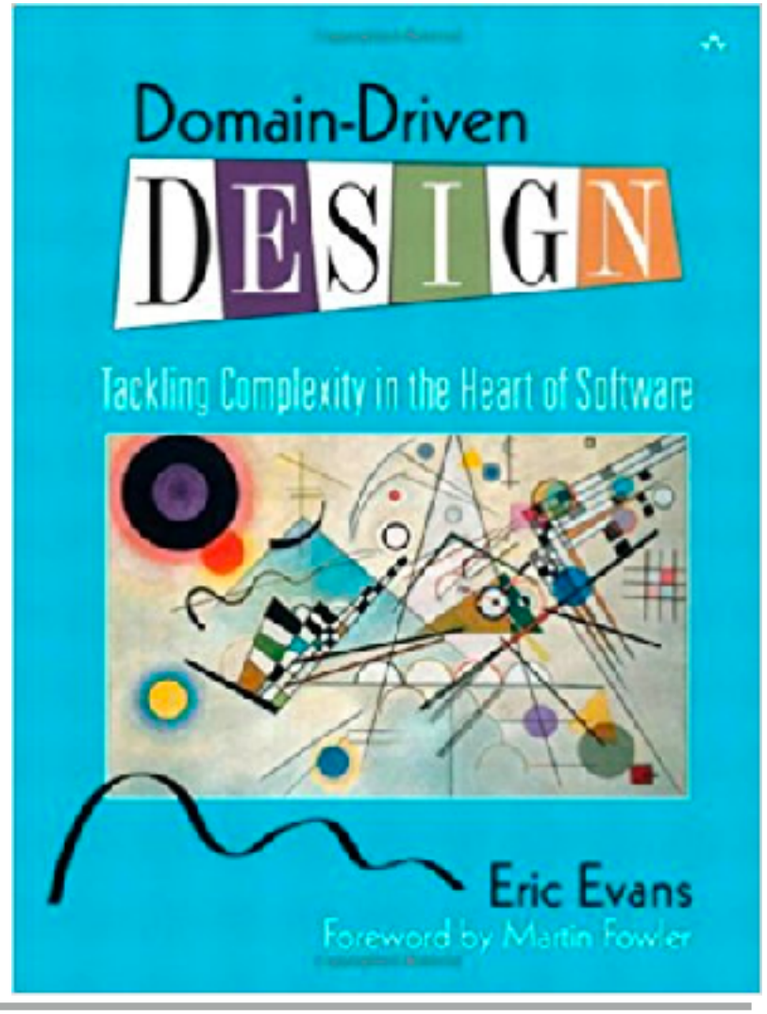






EFFECTIVE DOMAIN DRIVEN DESIGN

Oğuzhan Soykan

WHO AM I?

Oğuzhan Soykan
osoykan

Interests: DDD, CQRS, Event Sourcing

Edit bio

Software Developer @Trendyol
Istanbul
oguzhansoykan@gmail.com
http://www.oguzhansoykan.com

Organizations

osoykan

oguzsoykan

Overview | Repositories 25 | Stars 355 | Followers 75 | Following 94

Pinned repositories — Customize your pinned repositories

DynamicTranslator/DynamicTranslator
Instant translation application for windows in .NET
C# ★ 76 24

stoveproject/Stove
Domain Driven Design oriented application framework, meets CRUD needs
C# ★ 69 24

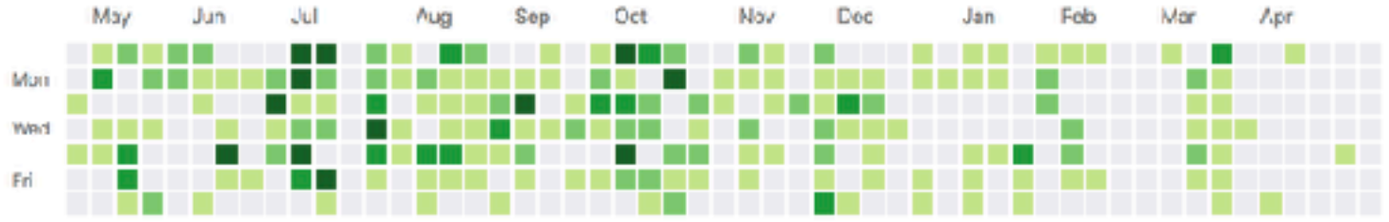
aspnetboilerplate/aspnetboilerplate
ASP.NET Boilerplate - Web Application Framework
C# ★ 4.5k 2k

FluentAssemblyScanner
Fluent assembly and type scanner for .Net
C# ★ 40 4

Autofac.Extras.IocManager
An abstraction for Autofac resolvings and registrations.
C# ★ 19 5

DDDify
Async await first, functional Domain Driven Design and CQRS library
C# ★ 5 2

890 contributions in the last year — Contribution settings



Learn how we count contributions.
Less More

LET'S GET STARTED

WHY?

THE NEED OF DDD

- **What is DDD?**
- **Learning existing domain as daily language**
- **Implementing business requirements effectively, we are business devs eventually(?)**
- **Improving the communication between teams and domain experts**
- **Everyone has no same way to learn new things in the business (shapes, visualized things, speaking, documentation)**

DESIGN PROBLEMS

- **Devs wrapped up too much technology**
- **The Database is given too much priority**
  - **One-To-Many, Zero-To-One, ForeignKey, NotNull, Where is my Join?**
- **Poor collaboration**
  - **Not focusing business complexity, struggling with tech side, not talking with Domain Experts**
- **Hidden business or leaked business to UI**
  - **UtilityHelpers, Providers, BullshitManagers!**
- **Task board shuffling rather than thoughtful design**

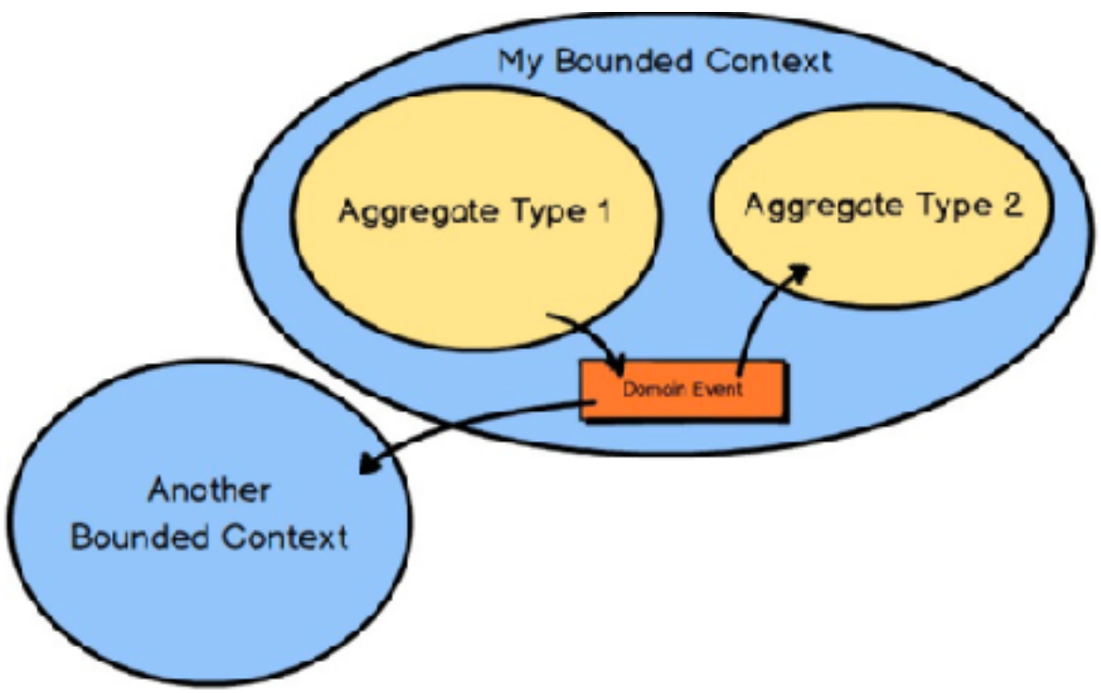
HOW TO DECIDE DDD?

Problems	Competitive Advantage	Complexity
Accounting(Internal)	S	L
Banner	L	S
Product	L	L

HOW TO DESIGN?

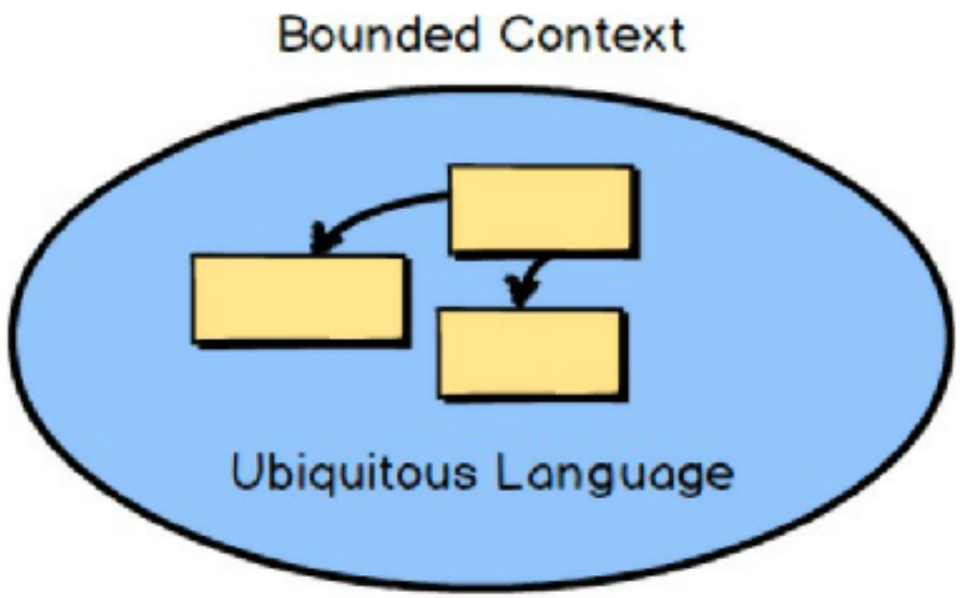
STRATEGIC DESIGN

- **Bounded Context**
- **Ubiquitous Language**

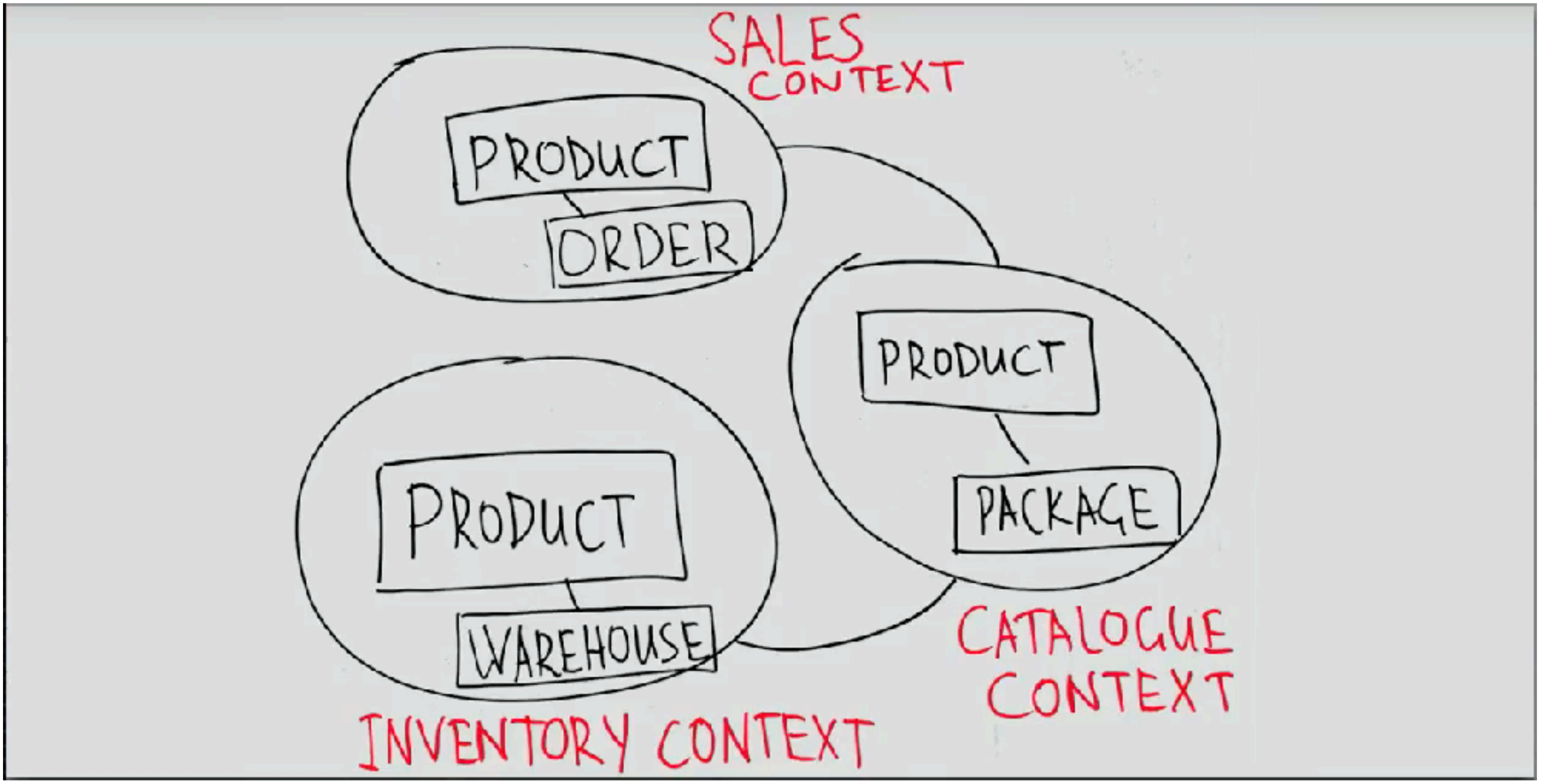


TACTICAL DESIGN

- **AggregateRoot**
- **Entity**
- **Value Objects**
- **Domain Events**



UBIQUITOUS LANGUAGE



WHAT ABOUT EFFECTIVENESS ?

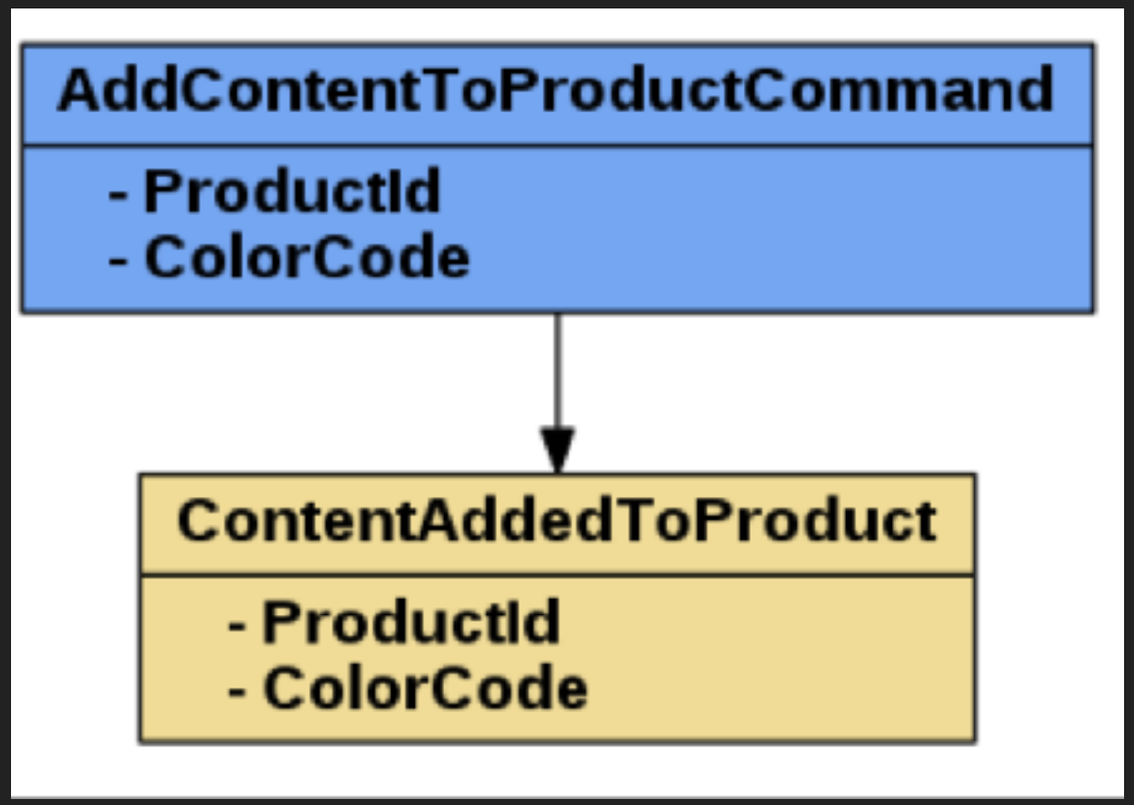
- **Anemic Domain Model**
- **Transaction script**
- **Mixing Read-Write**
- **Microservices?**
- **Modelling**
- **What about performance ?**
- **Coupled Services**
  - **Command –> Domain –> Rest(Another) –//–> Save**
- **The thought of changing the ORM, repository abstractions**

ANEMIC DOMAIN MODEL

USE AGGREGATE PATTERN EFFECTIVELY

ANEMIC DOMAIN MODEL

```
public class Product : AggregateRoot
{
    public ICollection<ProductContent> Contents;

    protected Product()
    {
        Register<Events.ContentAddedToProduct>(When);
    }
}
```



ANEMIC DOMAIN MODEL

```
public class ProductCommandHandler : IHandles<AddContentToProductCommand>
{
    public Task Handle(AddContentToProductCommand command)
    {
        if(_productRepository.Exists(command.ProductId))
        {
            if(command.ColorCode.Lenght > 50)
            {
                throw new BusinessException("ColorCode can not be greater than 50");
            }

            Product aggregate = _productRepository.Get(command.ProductId);
            if(aggregate == null)
            {
                throw new AggregateNotFoundException(command.ProductId);
            }

            ProductContent productContent = _productContentRepository
                                    .GetByCodeAndProductId(command.ColorCode, command.ProductId);
            if(productContent != null)
            {
                throw new BusinessException($"Content already added {command.ColorCode}");
            }

            aggregate.Contents.Add(new ProductContent(aggregate, command.ColorCode));

            _productRepository.Save(aggregate);
        }
    }
}
```

ANEMIC DOMAIN MODEL

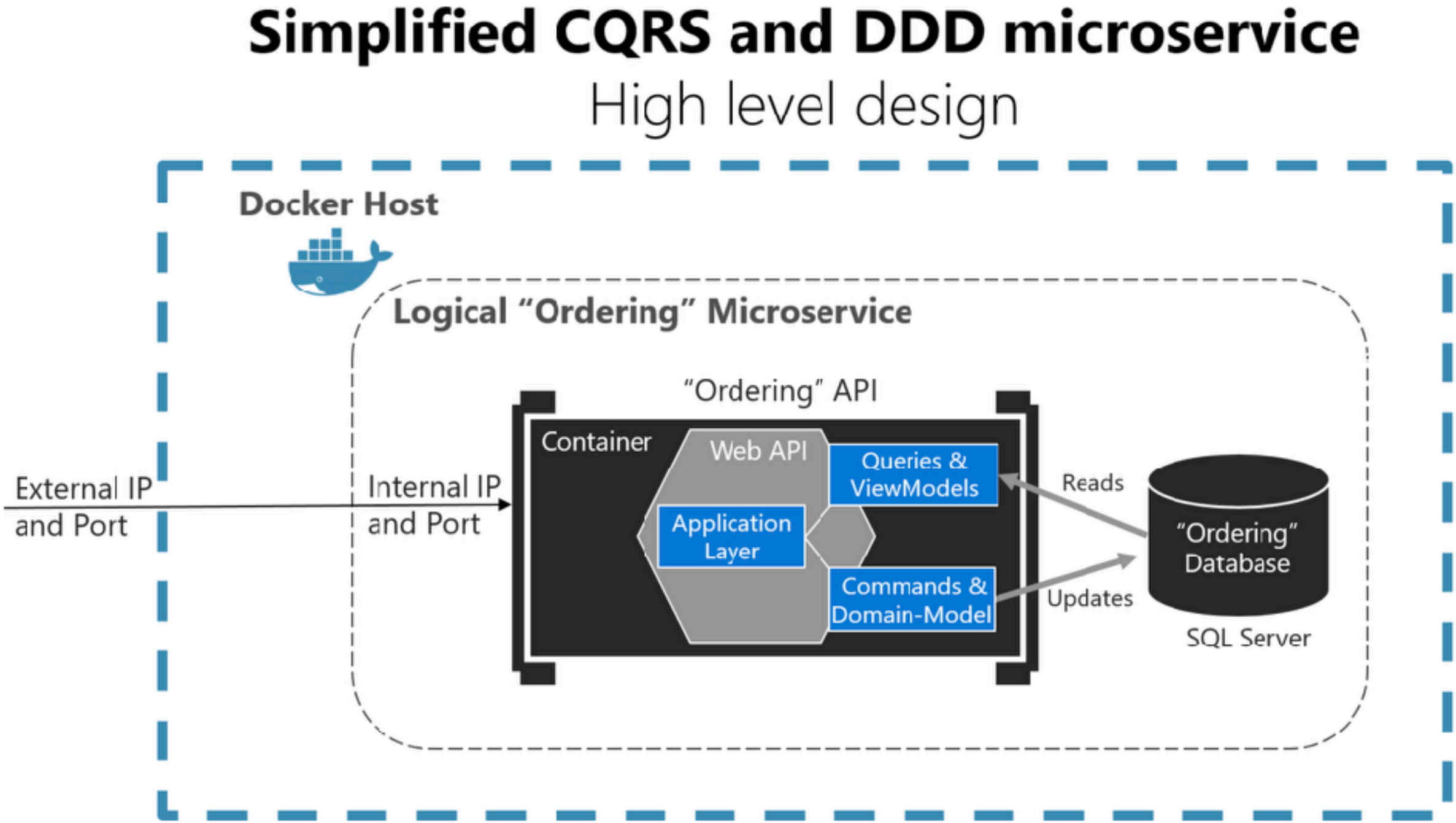
```
public class ProductCommandHandler : IHandles<AddContentToProductCommand>
{
    public Task Handle(AddContentToProductCommand command)
    {
        _productRepository
        .Get(command.ProductId)
        .Match(
            product => product.AddContent(command.ColorCode),
            () => throw new AggregateNotFoundException(command.ProductId);
        );
    }
}
```

ANEMIC DOMAIN MODEL

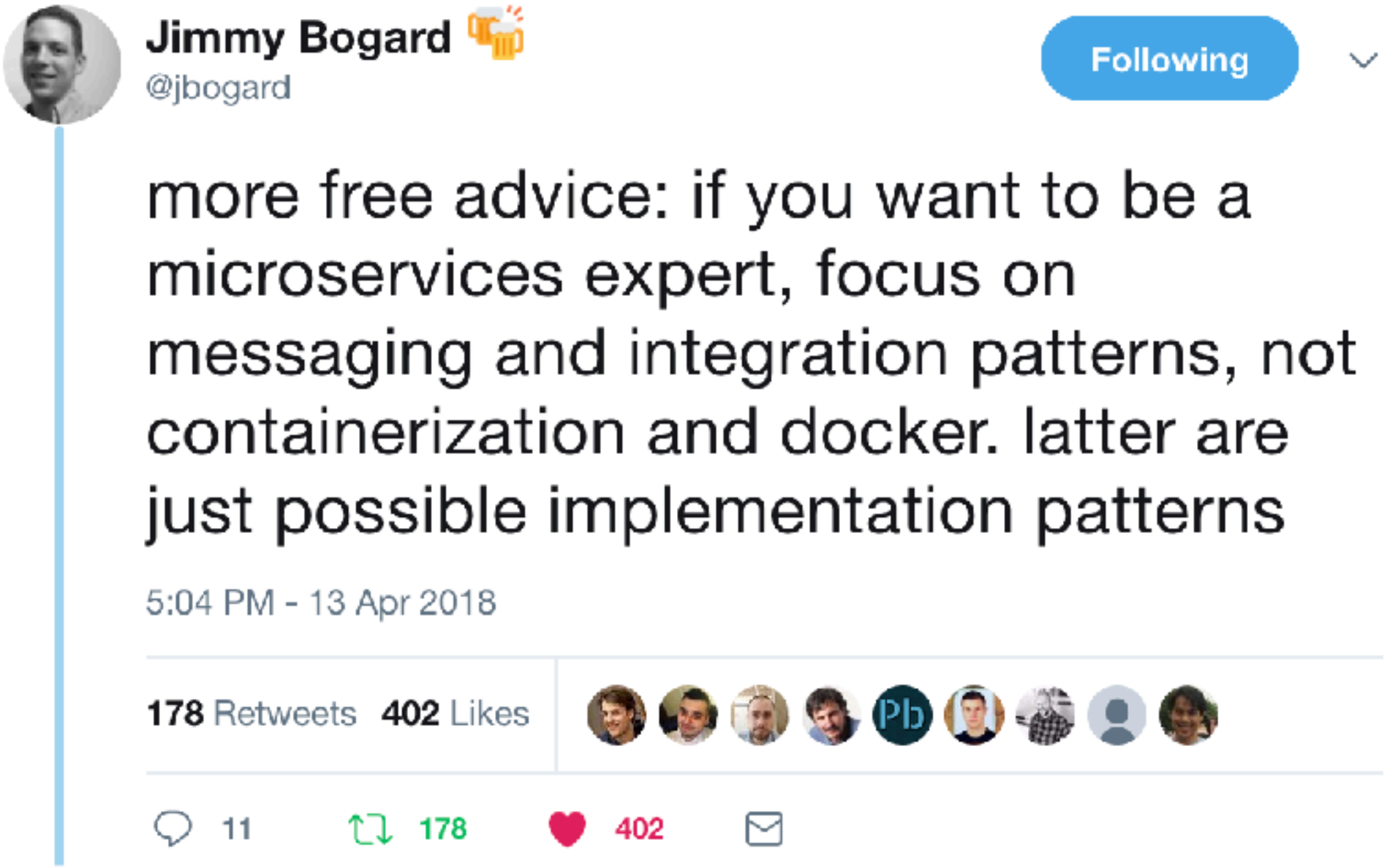
```
public class Product : AggregateRoot
{
    public ICollection<ProductContent> Contents;

    protected Product()
    {
        Register<Events.ContentAddedToProduct>(When);
    }

    public void AddContent(string colorCode)
    {
        if(colorCode.Lenght > 50)
        {
            throw new BusinessException("ColorCode can not be greater than 50");
        }

        if(Contents.Any(x => x.ColorCode == colorCode))
        {
            throw new BusinessException($"Content already added {colorCode}");
        }

        ApplyChange(
            new Events.ContentAddedToProduct(this.Id, colorCode);
        )
    }

    private void When(Events.ContentAddedToProduct @event)
    {
        Contents.Add(new ProductContent(this, @event.ColorCode));
    }
}
```

MICROSERVICES

BOUNDED CONTEXT PER MICRO SERVICE


Simplified CQRS and DDD microservice
High level design
Docker Host
Logical "Ordering" Microservice
"Ordering" API
Container
Web API
Queries & ViewModels
Application Layer
Commands & Domain-Model
External IP and Port
Internal IP and Port
Reads
Updates
"Ordering" Database
SQL Server

BOUNDED CONTEXT PER MICRO-SERVICE



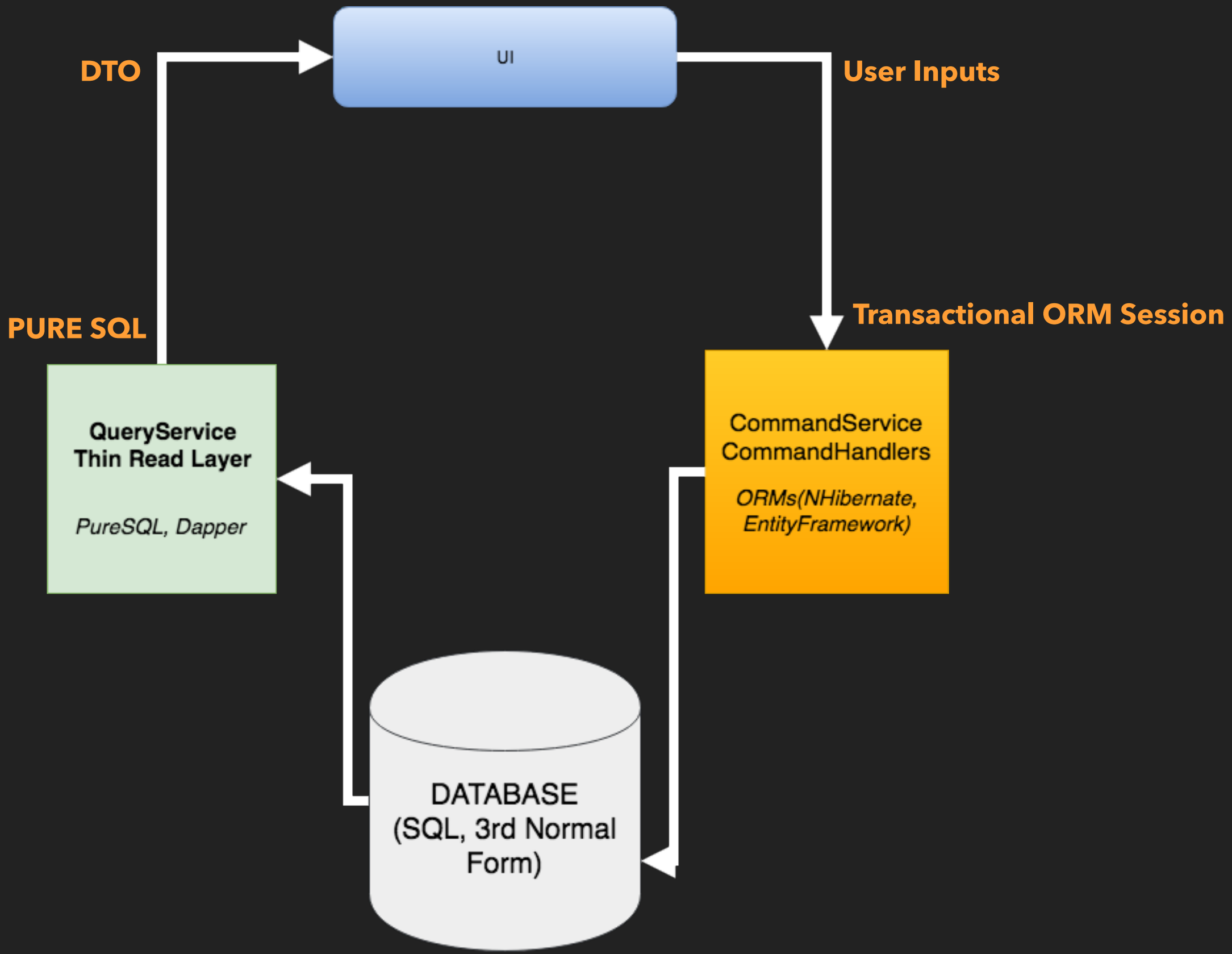
WHAT ABOUT PERFORMANCE ?

IMPEDANCE MISMATCH & CQRS

IMPEDANCE MISMATCH & CQRS

```
Set(x => x.Products, c =>
{
    c.Fetch(CollectionFetchMode.Join);
    c.BatchSize(100);
    c.Lazy(CollectionLazy.Lazy);
    c.Access(Accessor.Field);
    c.Sort<CustomComparer>();
    c.Type<CustomType>();
    c.Persister<CustomPersister>();
    c.OptimisticLock(true);
    c.Mutable(true);

  c.Key(k =>
      {
          k.Column("columnName");
          k.Column(x =>
          {
              x.Name("columnName");
          });

          k.ForeignKey("collection_fk");
          k.NotNullable(true);
          k.OnDelete(OnDeleteAction.NoAction);
          k.PropertyRef(x => x.Name);
          k.Unique(true);
          k.Update(true);
      });

      c.Cache(x =>
      {
          x.Include(CacheInclude.All);
          x.Usage(CacheUsage.ReadOnly);
          x.Region("regionName");
      });

      c.SqlDelete("SQL command");
      c.SqlDeleteAll("SQL command");
      c.SqlInsert("SQL command");
      c.SqlUpdate("SQL command");
      c.Subselect("SQL command");
      c.Loader("loaderRef");
}, r =>
{
      r.Element(e => { });
      r.Component(c => { });
      r.OneToMany(o => { });
      r.ManyToMany(m => { });
      r.ManyToAny<IAnyType>(m => { });
});

    c.Table("tableName");
      c.Schema("schemaName");
      c.Catalog("catalogName");

      c.Cascade(Cascade.All);
      c.Inverse(true);

      c.Where("SQL command");
      c.Filter("filterName", f => f.Condition("condition"));
      c.OrderBy(x => x.Name); // or SQL expression
```

IMPEDANCE MISMATCH & CQRS



(LOOSELY) COUPLED SERVICES

DOMAIN EVENTS

DOMAIN EVENTS

```
public abstract class AggregateRoot
{
    private ICollection<object> _events;
    private Dictionary<Type, Action<object>> _routes;

    void Register<TEvent>(Action<TEvent> route)
    {
        _routes.Add(typeof(TEvent), route);
    }

    void ApplyChange(object @event)
    {
        _routes[@event.GetType()].Invoke(@event);
    }

    public ICollection<object> GetChanges()
    {
        return _events;
    }
}
```

```
public class Product : AggregateRoot
{
    public ICollection<ProductContent> Contents;

    protected Product()
    {
        Register<Events.ContentAddedToProduct>(When);
    }
}
```

ENTITY FRAMEWORK & EVENT PUBLISHING

```
Func<Task> PrepareCompleteAction(TDbContext dbContext)
{
    if (dbContext.ChangeTracker.HasChanges())
    {
            List<EntityEntry> changes = dbContext.ChangeTracker.Entries().ToList();
            IEnumerable<AggregateRoot> trackedChanges = changes.Where(x => x.Entity is AggregateRoot)
                                                               .Select(x => (AggregateRoot)x.Entity);
           return async () =>
           {
                List<Event> events = trackedChanges.SelectMany(x => x.GetChanges()).OfType<Event>().ToList();
                foreach (Event @event in events)
                {
                    await _publisher.Publish(@event);
                }
           };
    }

    return () => Task.CompletedTask;
}
```

NHIBERNATE & EVENT PUBLISHING

```
public class NHibernateInterceptor : EmptyInterceptor
    {
        private readonly IMediator _mediator;

        public override void PostFlush(ICollection entities)
        {
            var changes = entities.OfType<object>().Where(e => e is AggregateRoot);
            foreach (var entity in changes)
            {
                TriggerDomainEvents(entity);
            }

            base.PostFlush(entities);
        }

        protected virtual void TriggerDomainEvents(object entity)
        {
            var aggregateChangeTracker = (AggregateRoot) entity;

            if (!aggregateChangeTracker.GetChanges().Any())
            {
                return;
            }

            var domainEvents = aggregateChangeTracker.GetChanges().ToList();
            aggregateChangeTracker.ClearChanges();

            foreach (var @event in domainEvents)
            {
                _mediator.Publish((INotification) @event);
            }
        }
    }
```

CHANGING THE O/RM

REALLY?

MODELLING

USE EVENT STORMING

https://webeventstorming.com

ADAPTECH
SOFTWARE THAT DEFIES CONVENTION

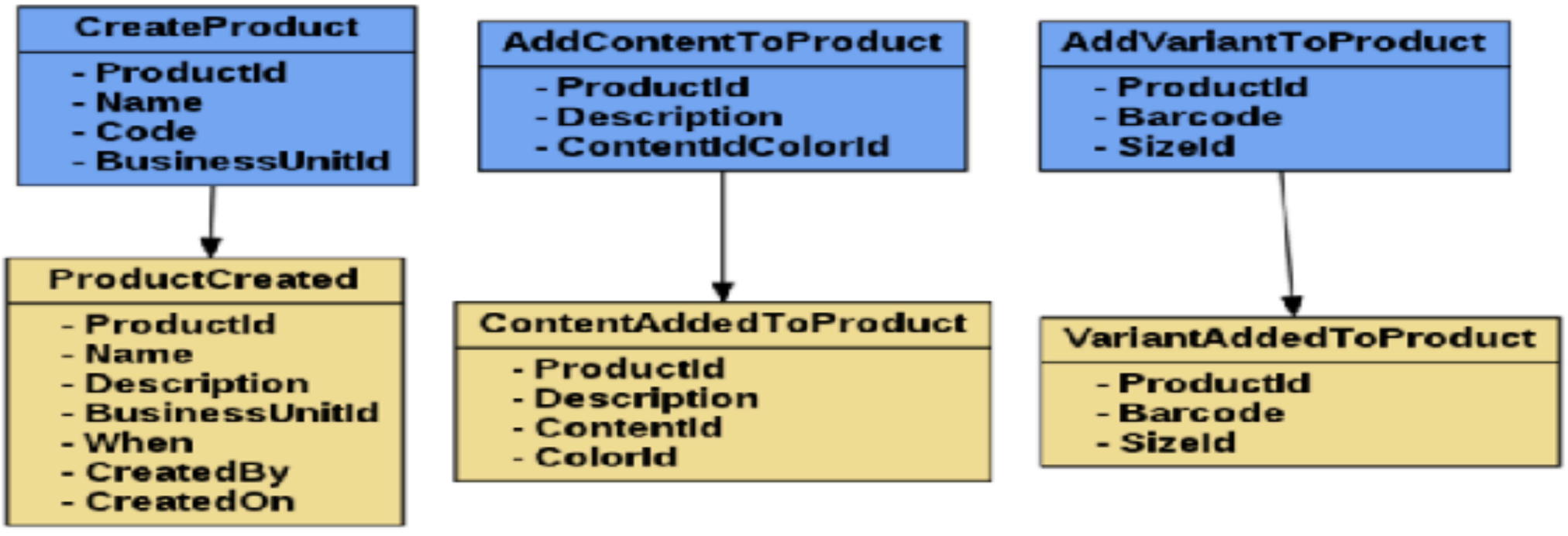
Example 1 Example 2 Example 3

CreateProduct-> // ProductId, Name, Code, BusinessUnitId
ProductCreated // ProductId, Name, Description, BusinessUnitId, When, CreatedBy, CreatedOn

AddContentToProduct-> // ProductId, Description, ContentId ColorId
ContentAddedToProduct //ProductId, Description, ContentId, ColorId

AddVariantToProduct-> //ProductId, Barcode,SizeId
VariantAddedToProduct //ProductId, Barcode, SizeId

Commands-> Events Documents*



THANKS

Oğuzhan Soykan