

DDD в микросервисах: сложность против сложности

Райффайзен



Обо мне



Константин Густов

архитектор,
Райффайзенбанк

10+ лет опыта в разработке
konst.gustov@gmail.com

Темы

I. Зачем нам DDD?

Темы

I. Зачем нам DDD?

II. Стратегическое проектирование

Темы

- I. Зачем нам DDD?
- II. Стратегическое проектирование
- III. Архитектура сервисов

Темы

- I. Зачем нам DDD?
- II. Стратегическое проектирование
- III. Архитектура сервисов
- IV. Представление в коде

Темы

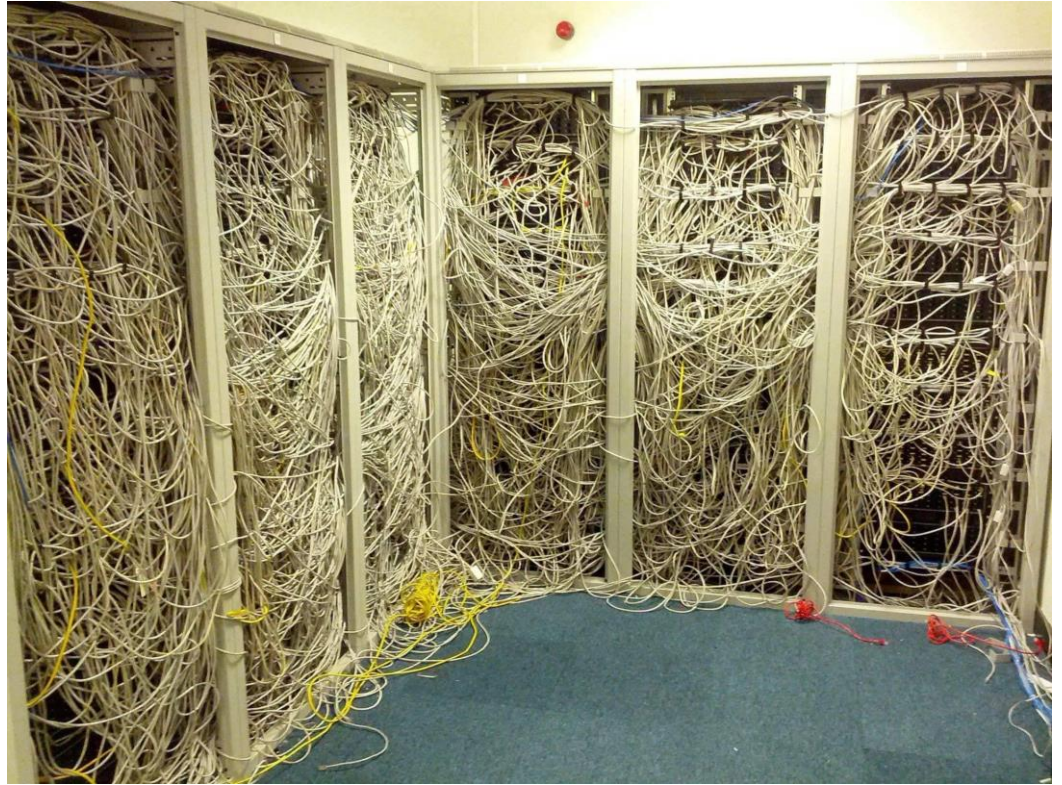
- I. Зачем нам DDD?
- II. Стратегическое проектирование
- III. Архитектура сервисов
- IV. Представление в коде
- V. Сложности внедрения

Темы

- I. Зачем нам DDD?
- II. Стратегическое проектирование
- III. Архитектура сервисов
- IV. Представление в коде
- V. Сложности внедрения
- VI. Что в итоге?

Начало: корпоративное приложение

- 500+ проектов
- 700+ тыс. строк кода



**Начало: процесс
разработки**

Бизнес



**Начало: процесс
разработки**

Бизнес



**Начало: процесс
разработки**

Бизнес



Разработка



Начало: процесс разработки

Бизнес



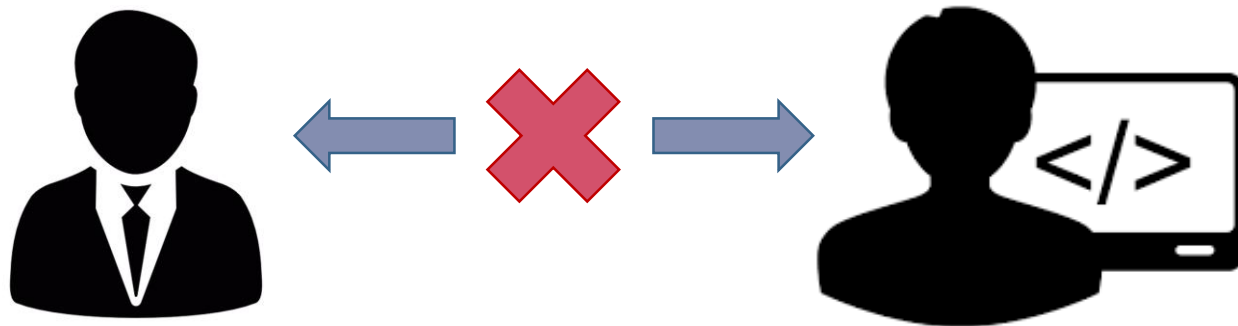
Разработка

Тестирование



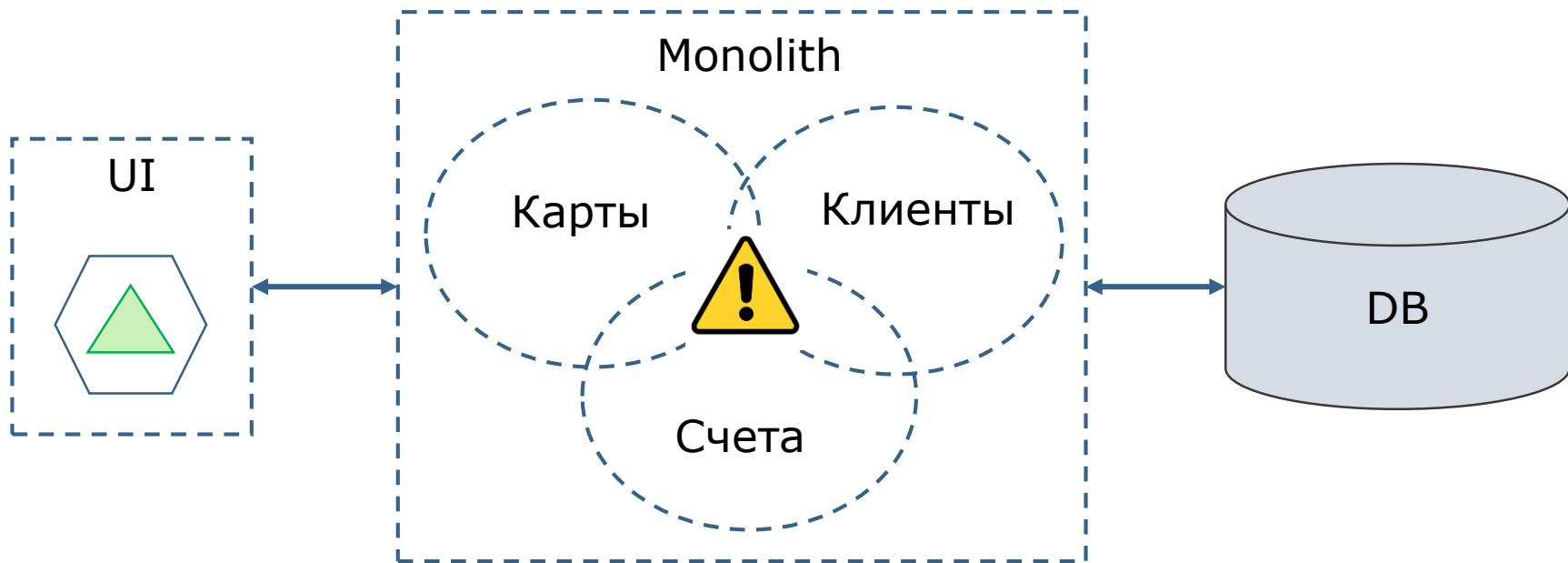
Недостатки

Бизнес и разработка говорили на разных языках



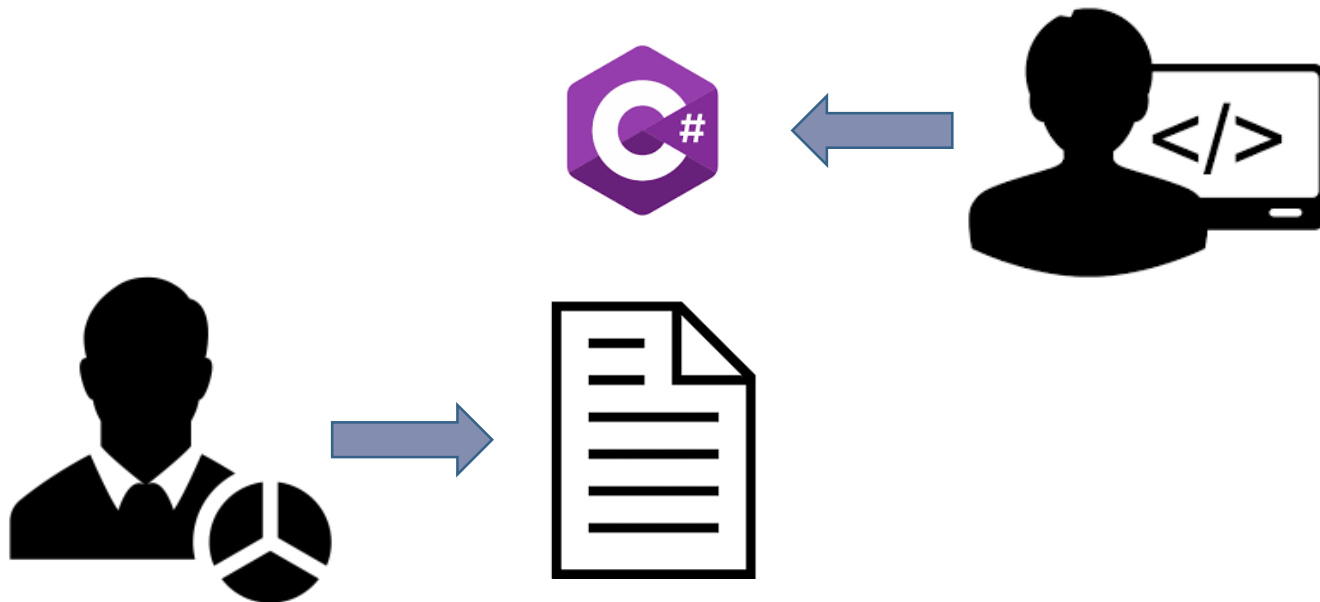
Недостатки

Бизнес-домены в приложении были смешаны



Недостатки

Участники команды разработки плохо понимали друга друга



Недостатки

Код плохо отражал бизнес-логику решения

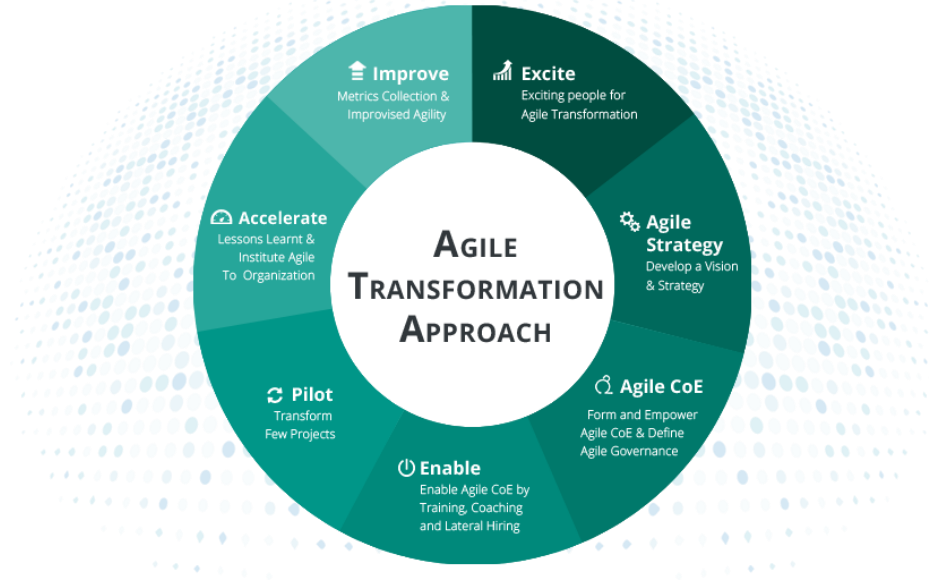
```
var customerOperationDT = new
CustomerOperationsDataSet.CUSTOMERS_OPERATIONSDataTable();

var customerOperationRow = customerOperationDT
    .CreateRow(cnum, comissionId.ToString(), CallingUser.Name);

var chargeAction = ComissionChangeAction.CHARGE;
var nonChargeAction = ComissionChangeAction.NONCHARGE;

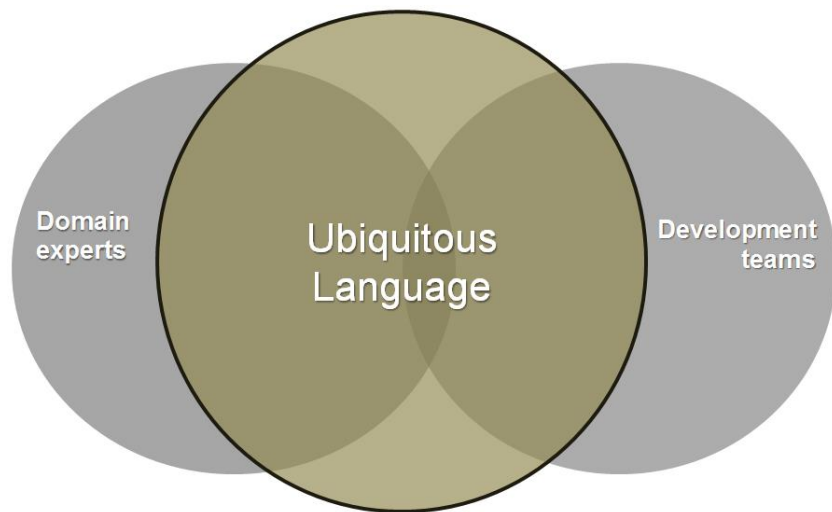
var changeActionRow = customerComissionData
    .CreateRow(cnum, comissionId.ToString(), CallingUser.Name, 1);
changeActionRow.OPTION_TITLE =
((ComissionOption)ComissionOption.ChangeAction).ToString();
```

Agile-трансформация



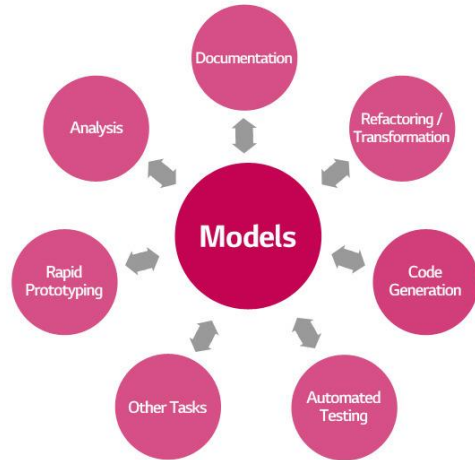
Ожидания

Появление единого языка



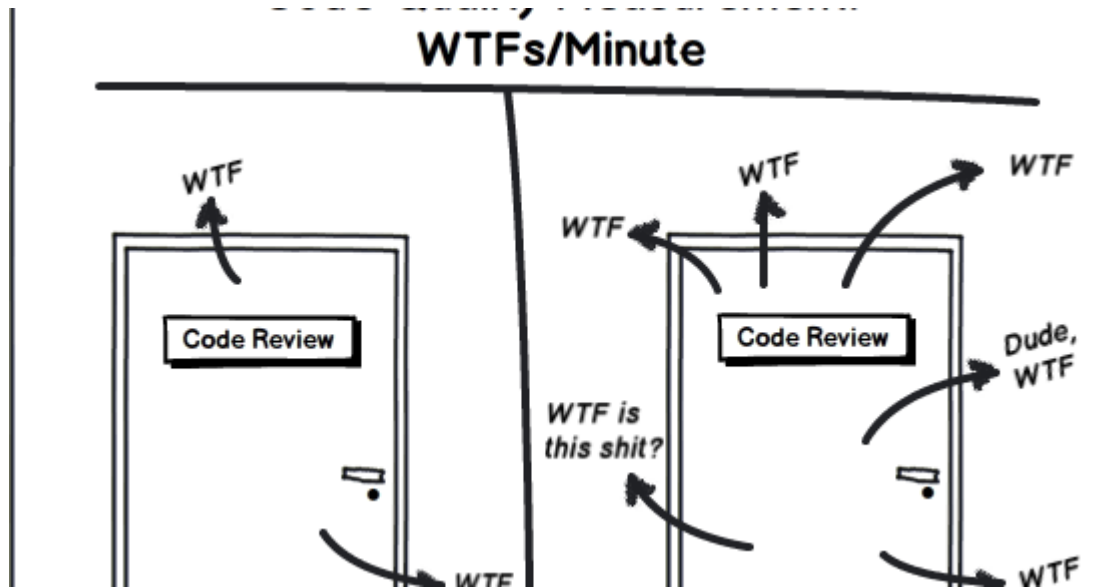
Ожидания

Выделение ограниченных контекстов

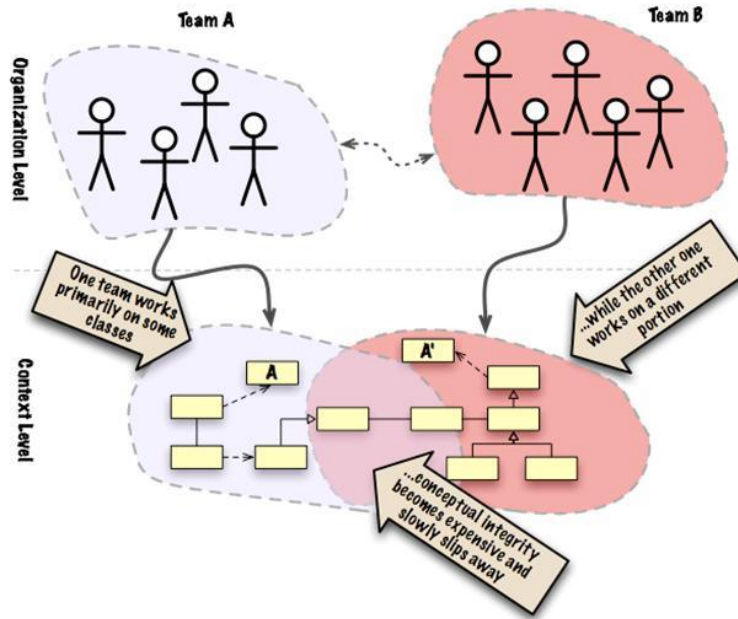


Ожидания

Повышение качества исходного кода



Стратегическое проектирование



Составляющие DDD

$$DDD = MDD + UL$$

Составляющие DDD

$$DDD = MDD + UL$$



model-driven design

Составляющие DDD

$$DDD = MDD + UL$$

model-driven design

Ubiquitous language

Составляющие DDD

$$\text{DDD} = \text{MDD} + \text{UL}$$



model-driven design

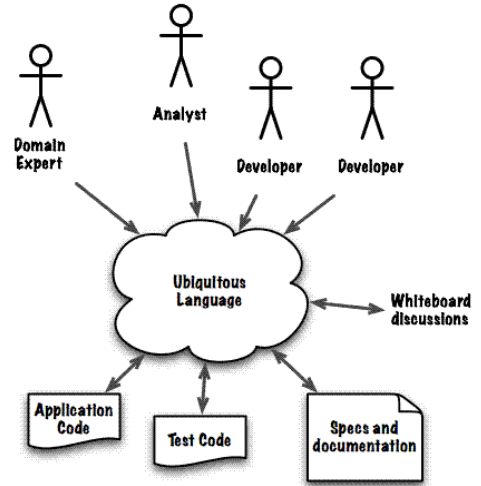


Ubiquitous language

Алексей Мерсон: Domain-driven design: рецепт для прагматика
<https://habr.com/ru/company/jugru/blog/440772/>

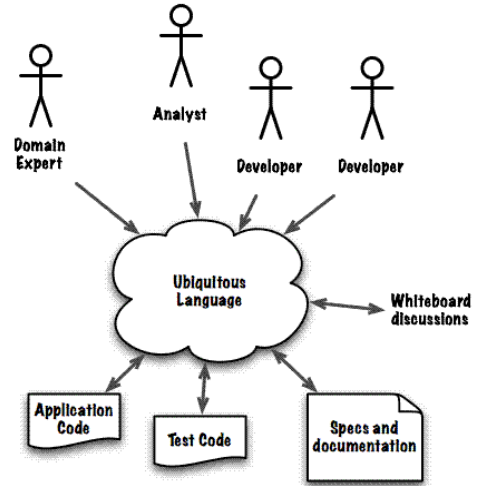
Работа над UL

- Использование устоявшихся терминов



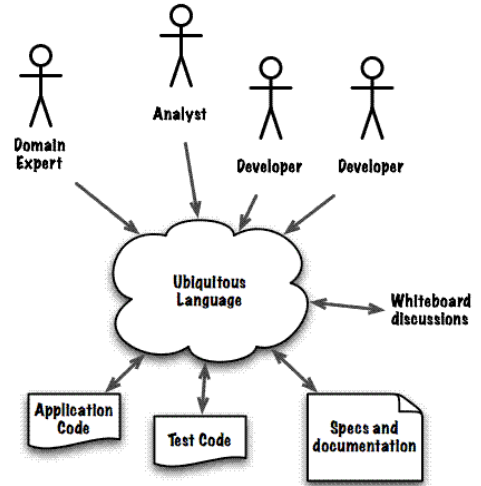
Работа над UL

- Использование устоявшихся терминов
- Выбор одного английского перевода



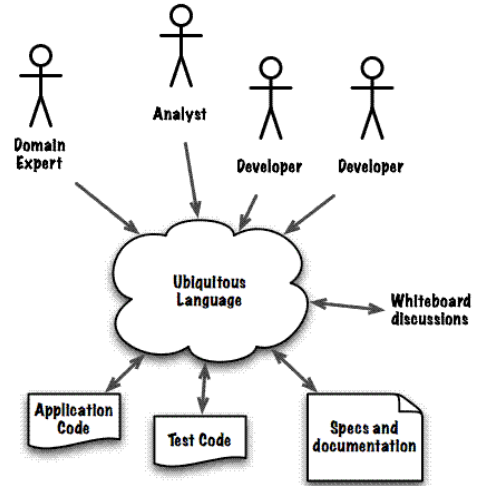
Работа над UL

- Использование устоявшихся терминов
- Выбор одного английского перевода
- Избегание многозначных терминов



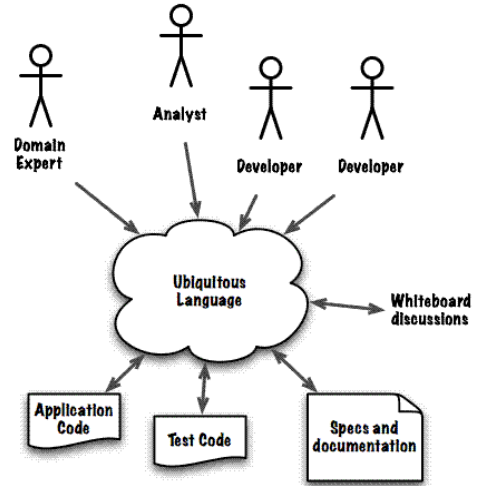
Работа над UL

- Использование устоявшихся терминов
- Выбор одного английского перевода
- Избегание многозначных терминов
- Отказ от имен паттернов проектирования в модели

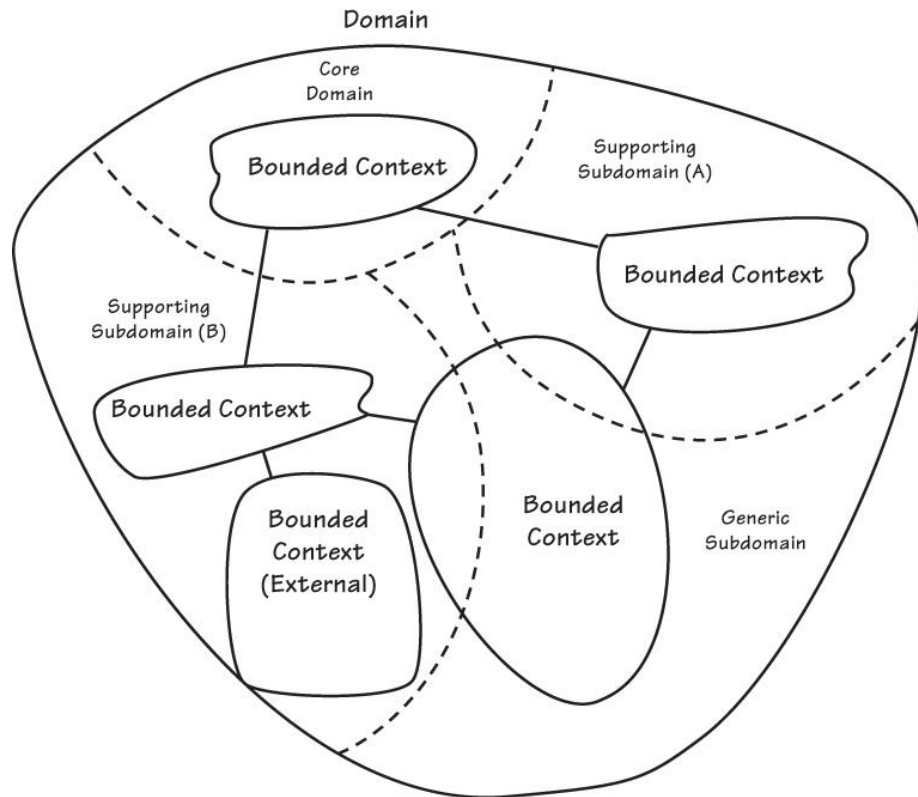


Работа над UL

- Использование устоявшихся терминов
- Выбор одного английского перевода
- Избегание многозначных терминов
- Отказ от имен паттернов проектирования в модели
- Контроль на ревью

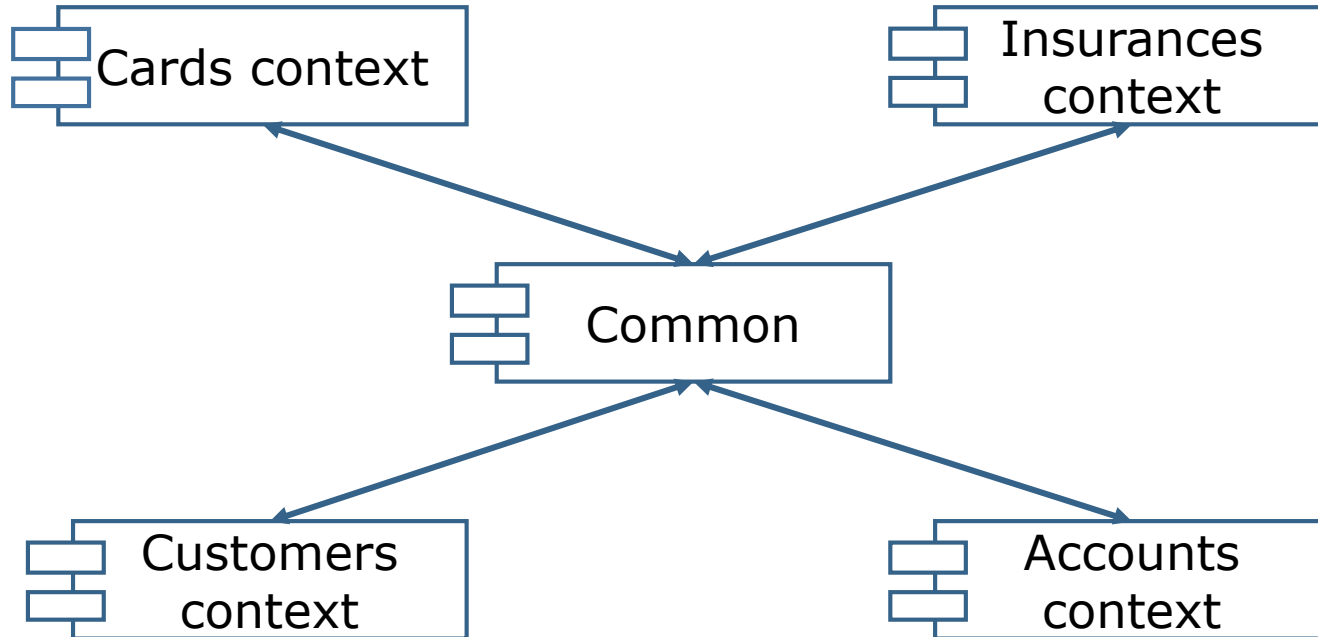


Разделение контекстов



First blood

Взаимодействие контекстов через общую сборку



First blood

```
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <class xmlns="urn:nhibernate-mapping-2.2" name="Cards.Domain.Entities.Card,
Cards.Domain" table="CARDS">
    <id name="Id" type="System.Int32">
      <column name="Id" />
      <generator class="identity" />
    </id>
    <property name="Number" type="System.String" />

    <many-to-one name="Account" class="Accounts.Domain.Entities.Account,
Accounts.Domain" column="AccountId" />

  </class>
</hibernate-mapping>
```



Ссылка на счет

First blood

```
public class Card : ICard
{
    protected Card() {}

    public virtual int Id { get; protected set; }
    public virtual string Number { get; protected set; }

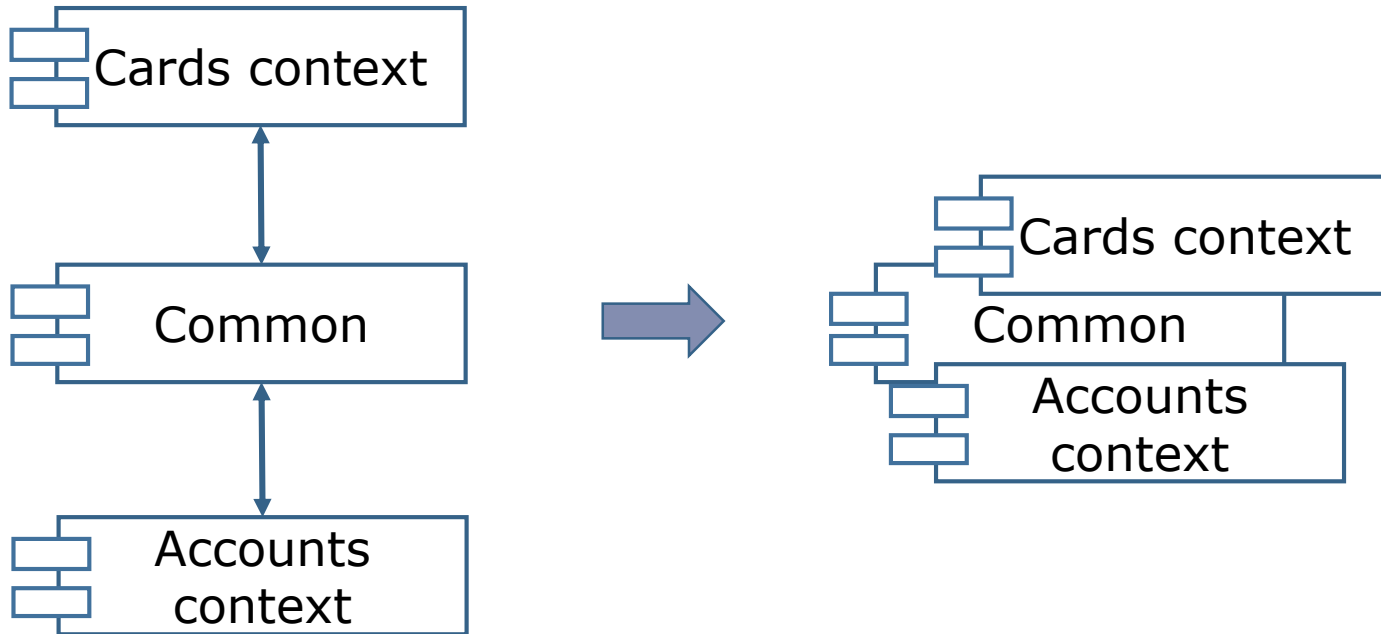
    public virtual IAccount Account { get; protected set; }
    ...
}
```



Ссылка на счет

First blood

Сильная связь между контекстами



Разделение контекстов

```
public class Card : ICard
{
    protected Card() {}

    public virtual IAccount Account { get; protected set; }
    ...
}
```




```
public class Card : IEntity<int>
{
    protected Card() {}

    public virtual AccountId Account { get; protected set; }
    ...
}
```

Общий интерфейс

Идентификатор


Разделение контекстов

```
using (var uow = _entityRepositoryFactory.Create())
{
    card.Account.Lock();  Интеграция через БД

    card.AddBlock(blockDetails);

    uow.Commit();
}
```



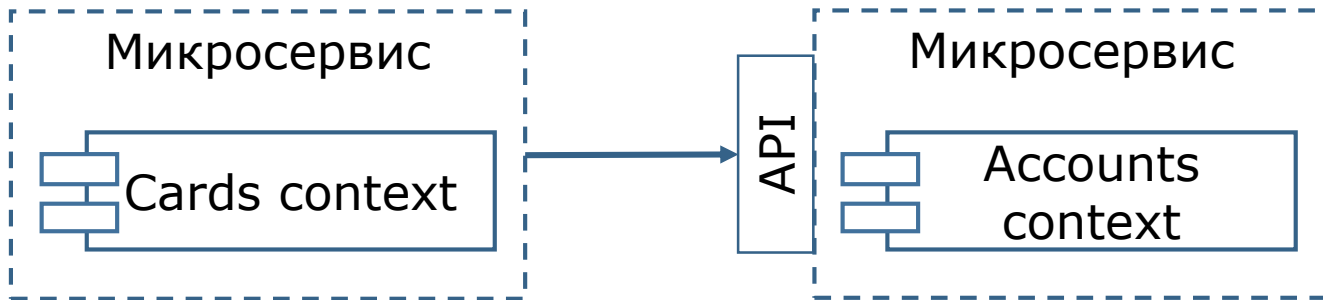
```
using (var uow = _entityRepositoryFactory.Create())
{
    _accountsService.LockAccount(card);  Интеграция через API

    card.AddBlock(blockDetails);

    uow.Commit();
}
```

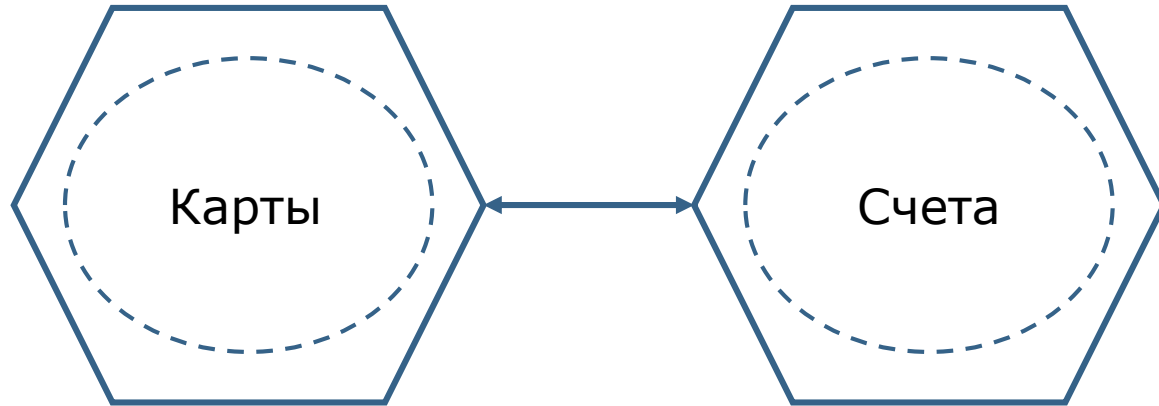
Разделение контекстов

Контексты общаются через API и разделены между сервисами



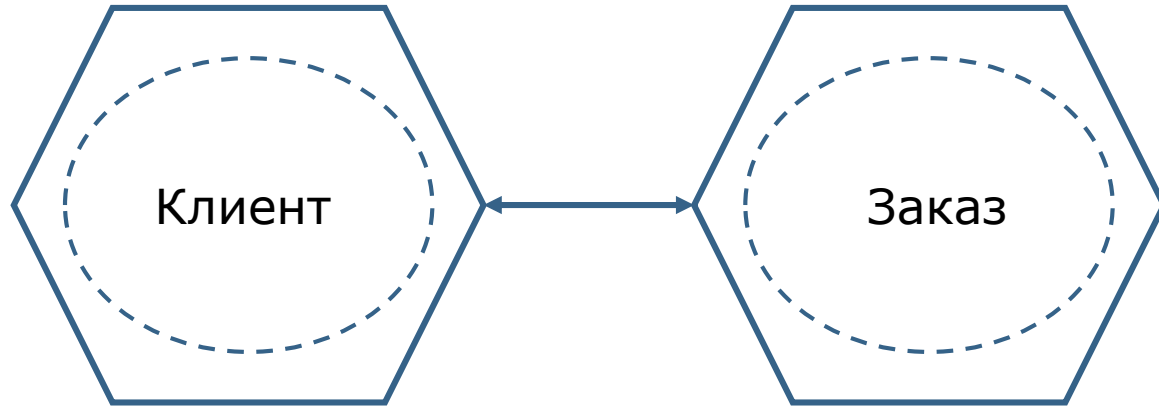
Размер микросервиса

Ограниченный контекст на микросервис



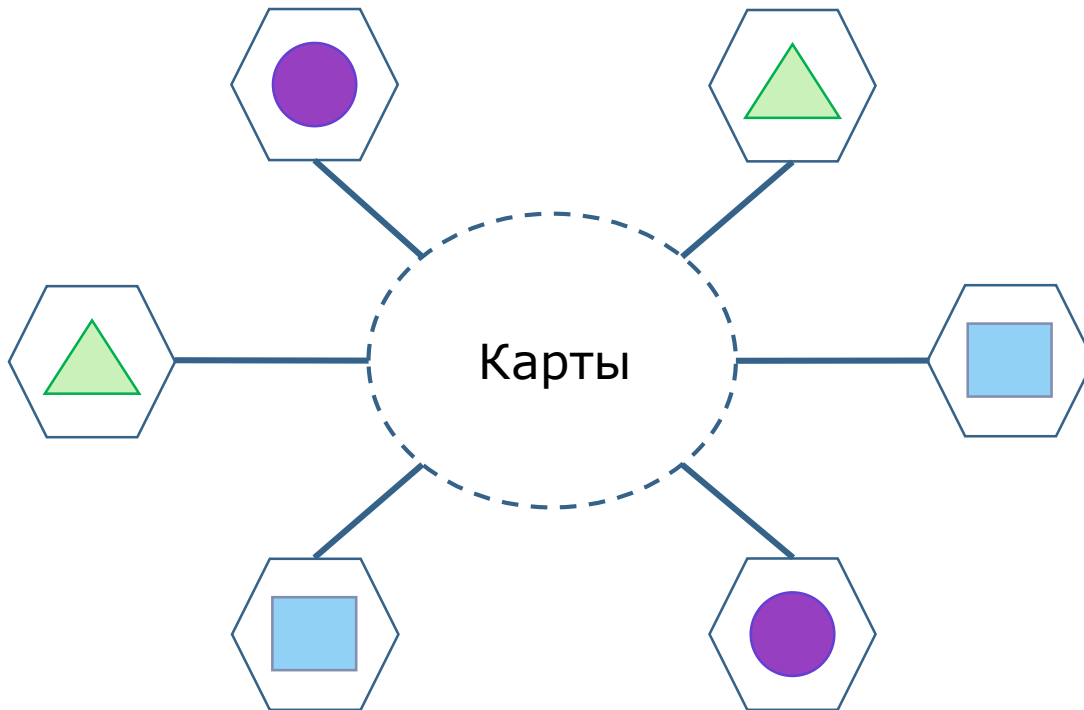
Размер микросервиса

Агрегат на микросервис



Размер микросервиса

Агрегат на несколько физических микросервисов



Размер микросервиса: пример

```
public class CardForBlocking : IEntity<int>
{
    public virtual void AddBlock([NotNull] BlockDetails blockDetails)
    {
        Validate(blockDetails);

        var newBlock = new CardBlock(blockDetails, this);
        _blocks.Add(newBlock);
        ...
        UpdateInfo = new UpdateInfo(blockDetails.User);
    }

    public virtual void RemoveBlock([NotNull] BlockDetails blockDetails)
    {
        ...
    }
}
```

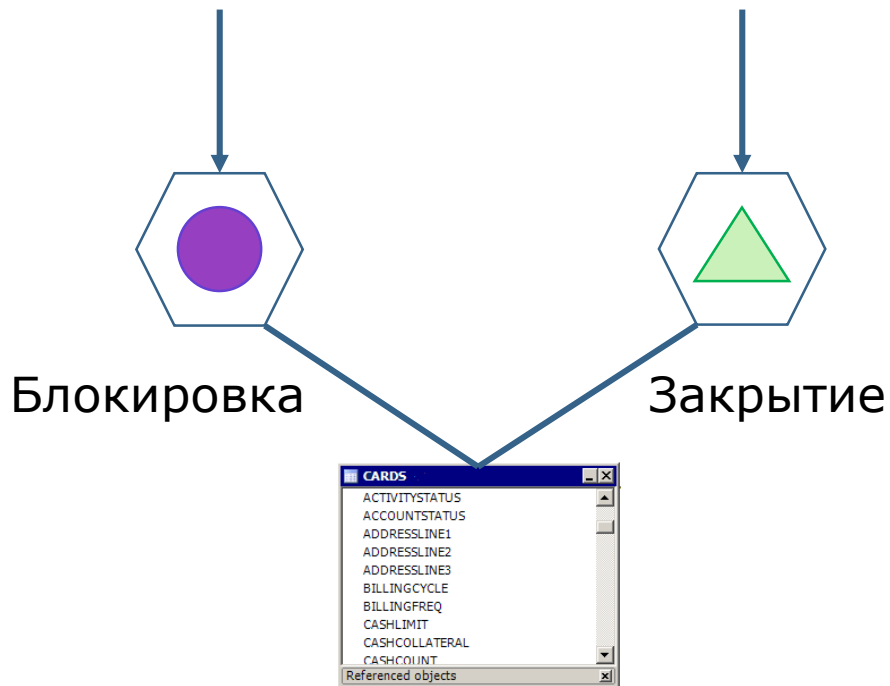
Размер микросервиса: пример

```
public class CardForClosing : IEntity<int>
{
    public virtual void CloseCard([NotNull] ClosingDetails closingDetails)
    {
        Validate(closingDetails);

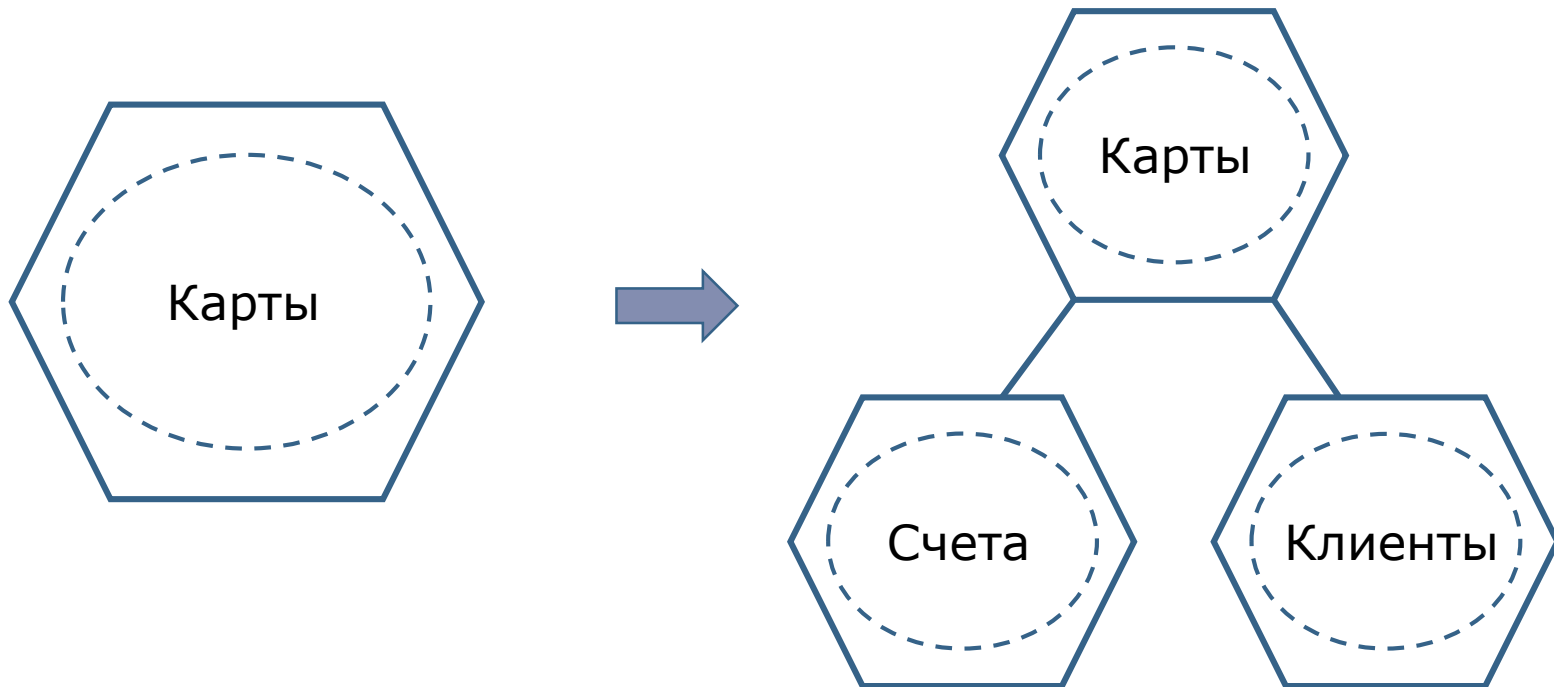
        Status = CardStatus.CreateClosed();
        SetFreezed(closingDetails);
        SetCloseReason(closingDetails);

        UpdateInfo = new UpdateInfo(closingDetails.User);
    }
}
```

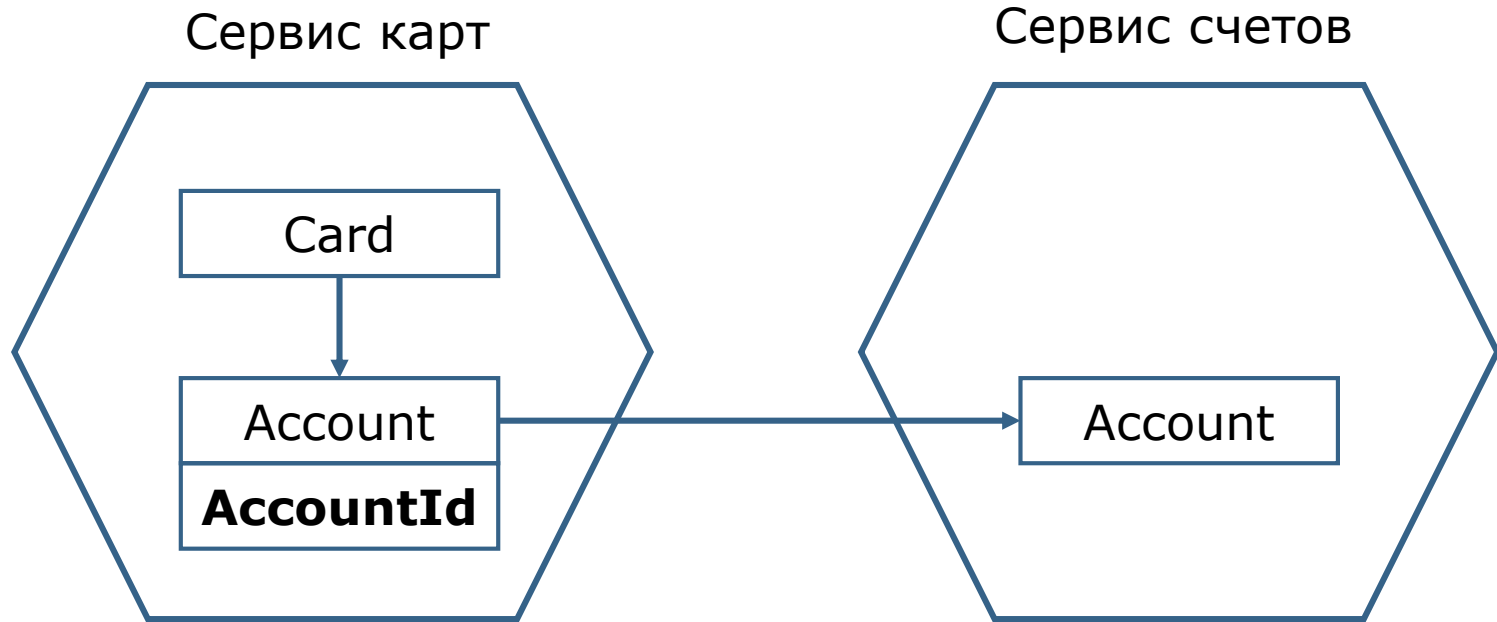
Размер микросервиса: пример



Связь микросервисов



Связь микросервисов

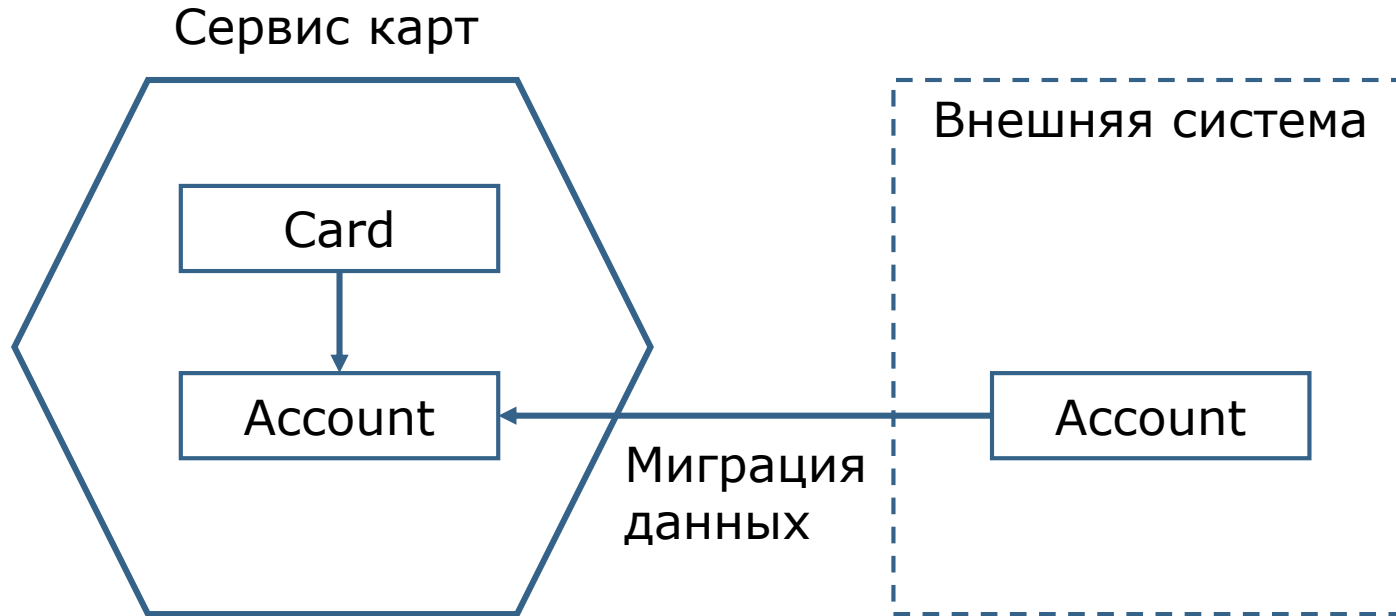


Мнение Ayende



Any 3rd party system
that I have to integrate
with was written by a
drunken monkey typing
with his feet

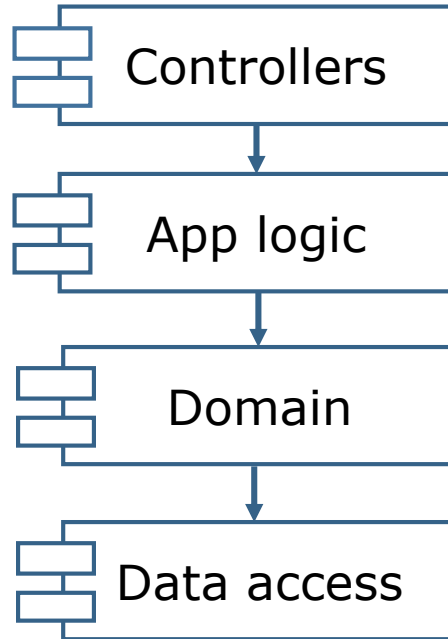
Связь с внешними сервисами



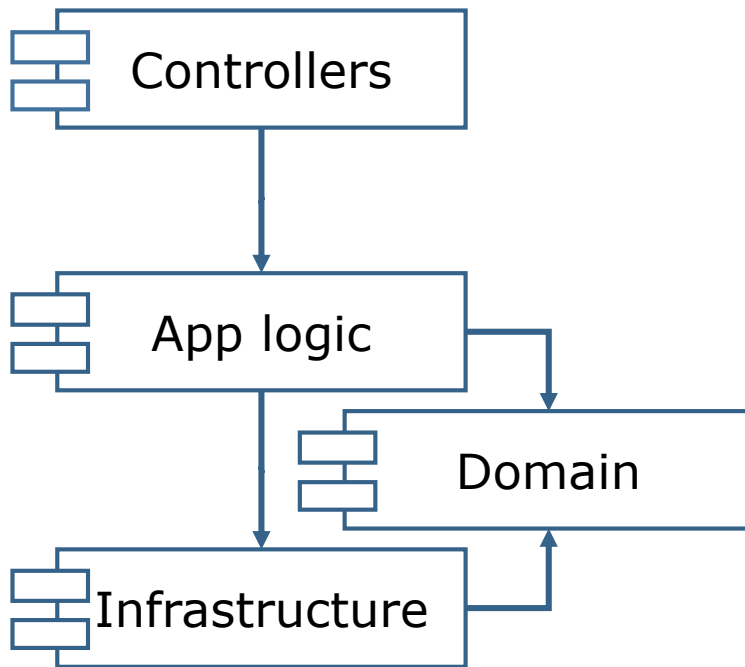
Архитектура сервисов



Классическая архитектура

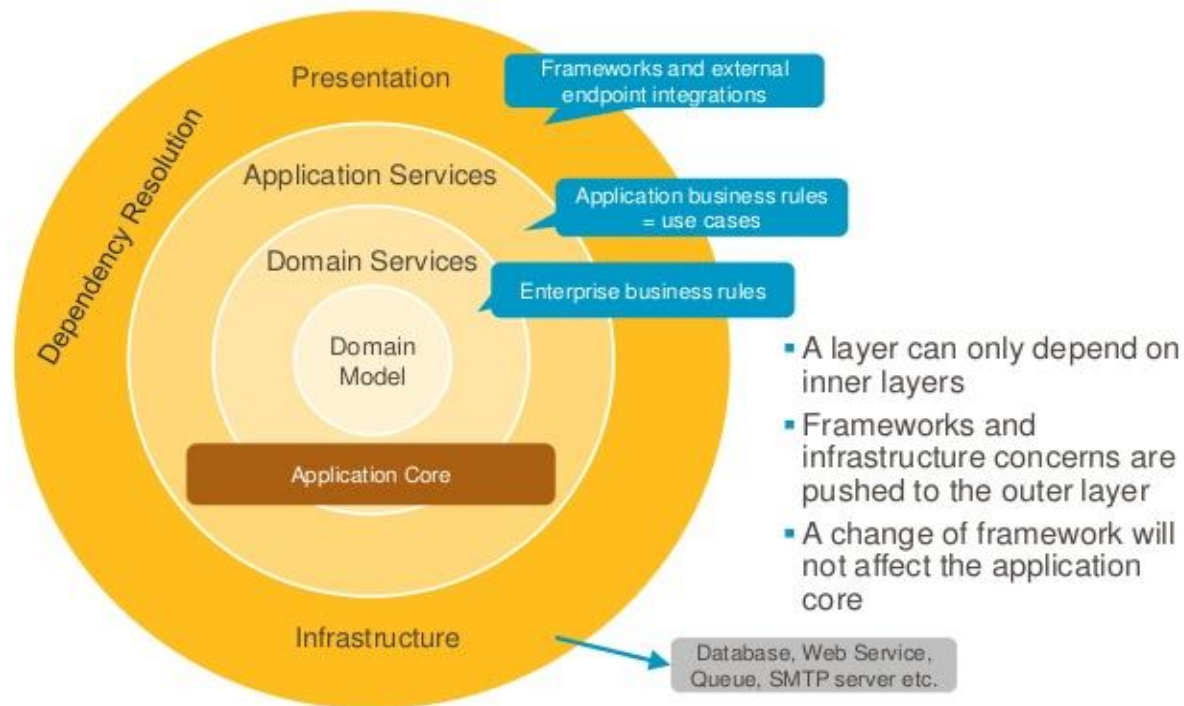


Архитектура, ориентированная на домен



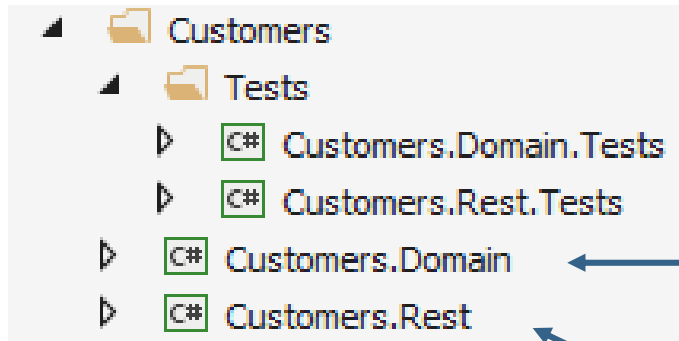
Луковая архитектура

Onion Architecture



Распределение слоев по сборкам

Модульные и интеграционные тесты

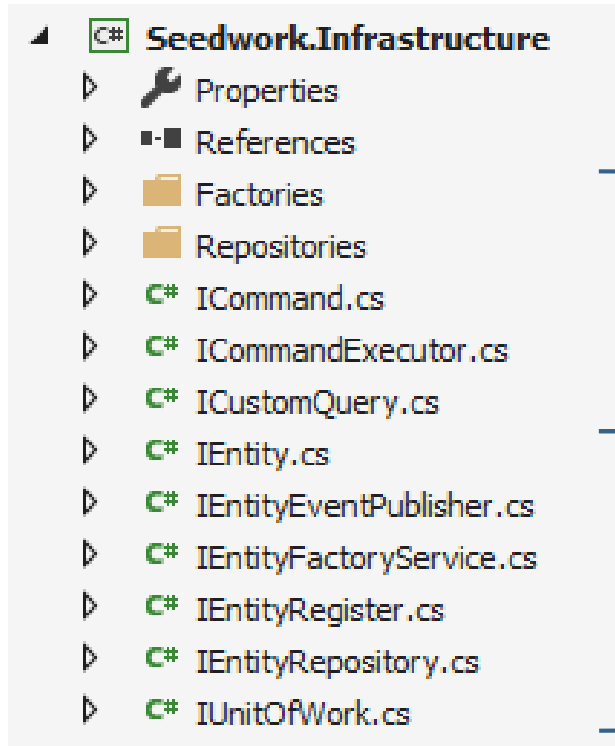


Доменная логика

- Web API
- Аппликационная логика
- Использование инфраструктуры

Seedwork

Базовые классы расположены в сборке с инфраструктурой



← Инфраструктура

← Базовые интерфейсы
доменной модели

Проблемы с зависимостями

```
try
{
    ...
}
catch (StaleObjectStateException)
{
    _stateProcessing = ProcessState.Skip;
}
```

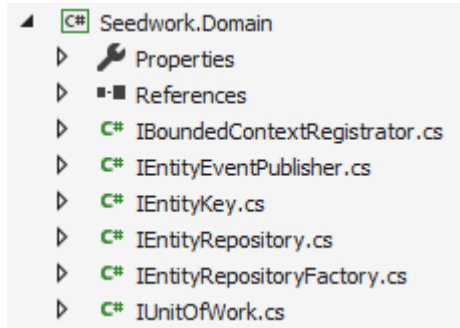
Оптимистическая
блокировка NHibernate

```
public void NullSafeSet(IDbCommand cmd, object value, int index)
{
    var goods = value as IList<GoodsService>;
    if (goods != null)
    {
        goods.ForEach(fee => NHibernateUtil.Decimal.NullSafeSet(cmd,
            good.Amount, index + (int) good.Type));
    }
}
```

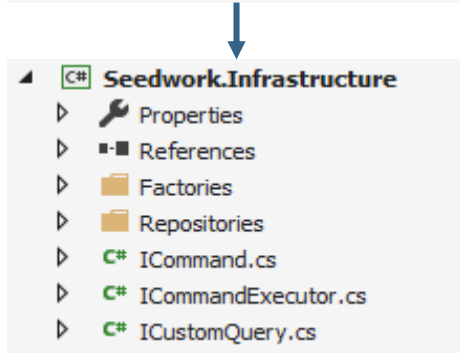
Инфраструктурный тип с
бизнес-логикой

Seedwork v2.0

Базовые классы и инфраструктура в отдельных сборках

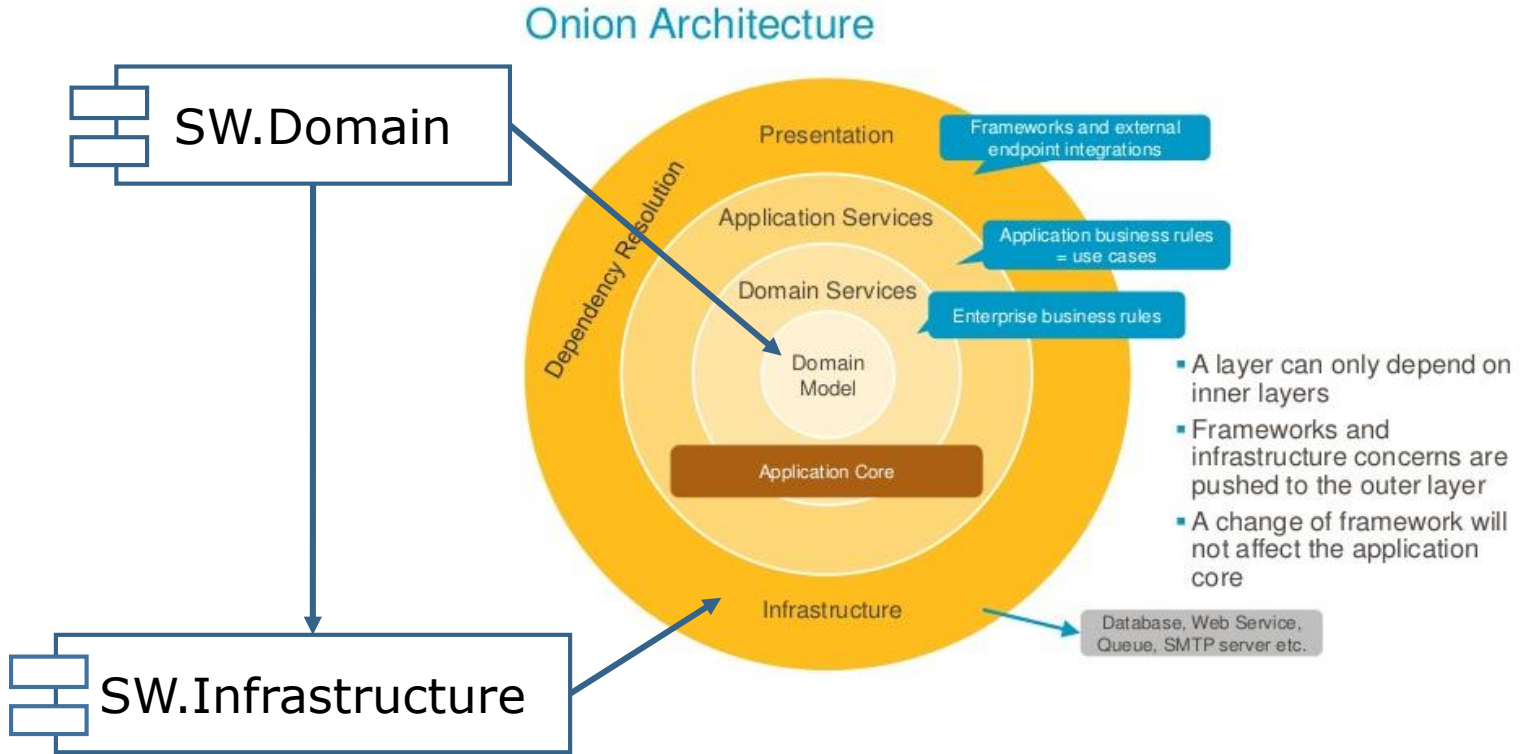


← Базовые интерфейсы
доменной модели



← Инфраструктура

Separated interface

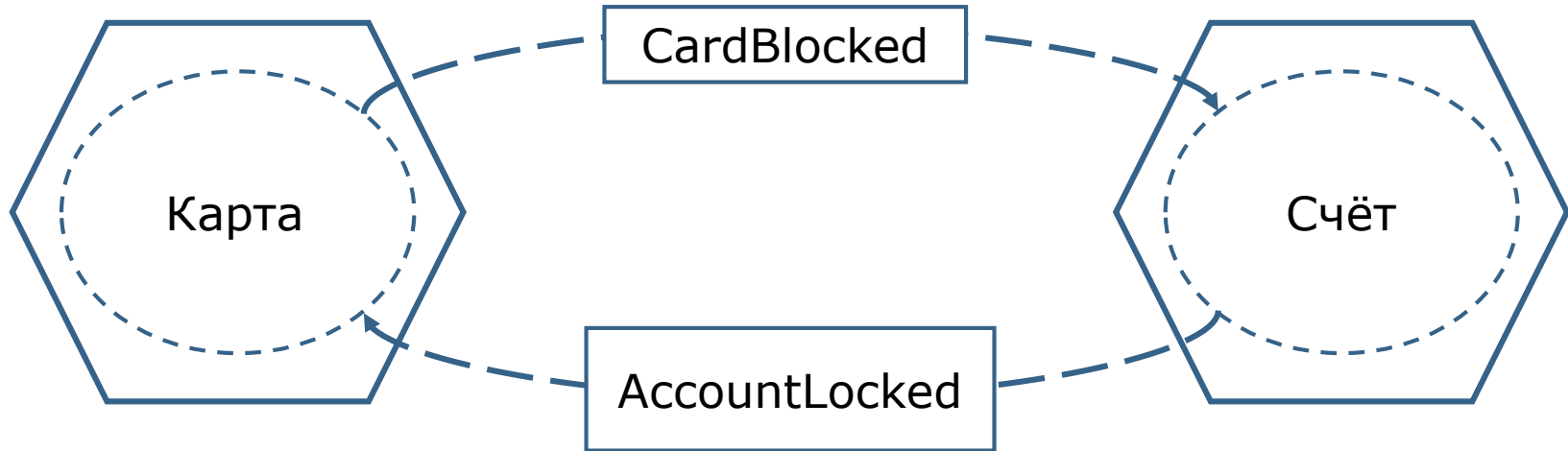


Представление в коде

```
17 string sInput;  
18 int iLength, iN;  
19 double dblTemp;  
20 bool again = true;  
21  
22 while (again) {  
23     iN = -1;  
24     again = false;  
25     getline(cin, sInput);  
26     system("cls");  
27     stringstream(sInput) >> dblTemp;  
28     iLength = sInput.length();  
29     if (iLength < 4) {  
30         again = true;  
31         continue;  
32     } else if (sInput[iLength - 3] != '.') {  
33         again = true;  
34         continue;  
35     } while (++iN < iLength) {  
36         if (isdigit(sInput[iN])) {  
37             continue;  
38         } else if (iN == (iLength - 3)) {  
39             continue;  
40         }  
41     }  
42 }
```

Доменные события

Основа событийно-ориентированного проектирования



Производство доменных событий

```
public class Card : IEntity<int>, IEventProvider
{
    private IEventCollector _eventCollector;
    ...
    public virtual void AddBlock([NotNull] BlockDetails blockDetails)
    {
        ...
        _eventCollector.CollectEvent(new CardBlocked(Id, blockDetails));
    }

    public virtual void SetCollector(IEventCollector sender)
    {
        _eventCollector = sender;
    }
}
```

Индикация в метод

Реализация с NHibernate

```
public class OurInterceptor : EmptyInterceptor
{
    private readonly IEventCollectorFactory _factory;

    public override bool OnLoad(object entity, ...)
    {
        if (entity is IEntityProvider eventSupportable)
            eventSupportable.SetCollector(_factory.Create());

        return false;
    }
    public override bool OnSave(object entity, ...)
    {
        ...
    }
}
```

Фабрика

```
public sealed class EventCollectorFactory : IEventCollectorFactory
{
    private readonly IEventSender _eventSender;
    private readonly AsyncLocal<IEventCollector> _collector;

    public EventCollectorFactory(IEventSender eventSender)
    {
        _eventSender = eventSender;
    }

    public IEventCollector Create()
    {
        return _collector.Value = _collector?.Value ??
            new EventCollector(_eventSender);
    }
}
```


Сборщик событий

```
internal sealed class SimpleEventCollector : IEventCollector
{
    private readonly ConcurrentQueue<IEvent> _events;
    private readonly IEventSender _sender;

    public void CollectEvent<TEvent>(TEvent message) where TEvent : class, IEvent
    {
        _events.Enqueue(evt);
    }

    public void Send()
    {
        while (!_events.IsEmpty)
            if (_events.TryDequeue(out var evt))
            {
                _sender.SendMessage(evt as dynamic);
            }
    }
}
```

Использование в коде приложения

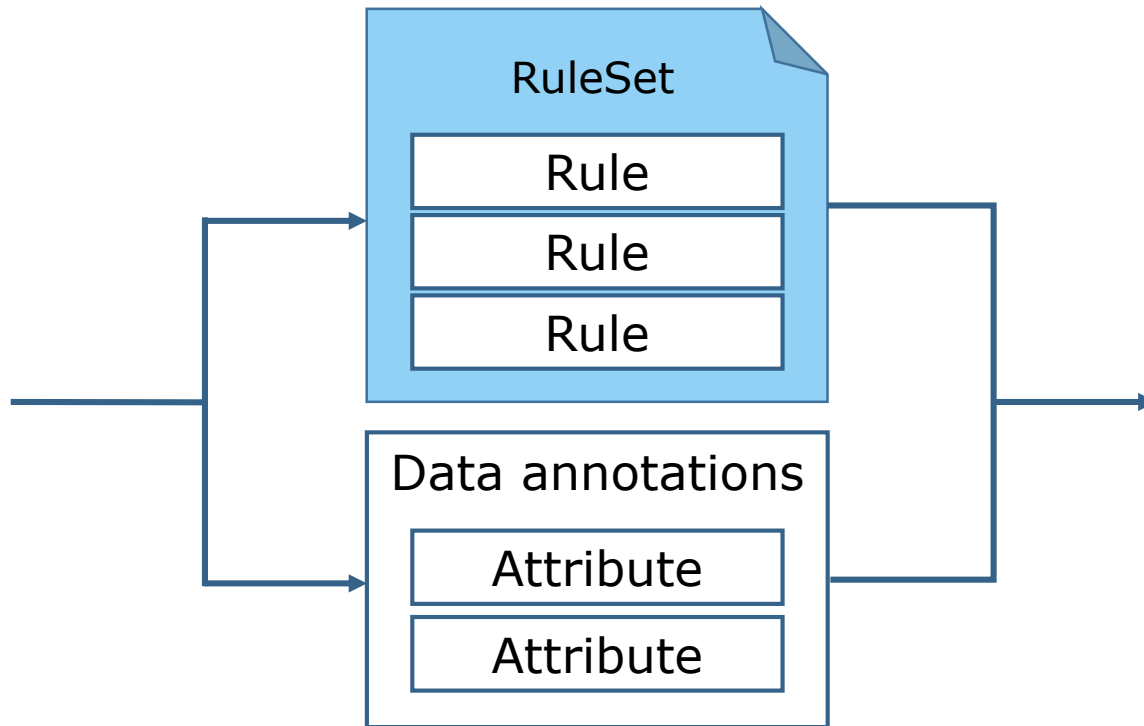
```
using (var scope = _eventCollectorFactory.BeginCollection())
{
    using (var uow = _entityRepositoryFactory.Create())
    {
        card.AddBlock(blockDetails);

        scope.Send();

        card.DoSomeOtherStuff(blockDetails);
        uow.Commit();
    }
}
```

Валидация данных

Валидация распространяется на все слои сервиса



Правило валидации

```
public class ClientIsAliveRule : BaseFluentRule<Card>
{
    private readonly IClientService _clientService;
    public ClientIsAliveRule(IClientService clientService)
    {
        _clientService = clientService;
        ForProperty(x => x.ClientId)
            .Must(IsClientAlive)
            .WithErrorMessage("Клиент недоступен");
    }
    private bool IsClientAlive(int clientId)
    {
        return _clientService.GetClient().IsAlive();
    }
}
```

Группировка правил

```
public class CardBlockingRuleSet : BaseValidationRuleSet<Card>
{
    public CardBlockingRuleSet(ClientIsAliveRule aliveRule,
NiceRule nRule, AwesomeRule aRule, StopRule stopRule, CrazyRule
cRule)
    {
        When(aliveRule,
            () =>
            {
                SetRule(nRule).DependsOn(aRule);
                SetRule(stopRule).StopOnFailure();
                SetRule(cRule);
            });
    }
}
```

Использование валидации

```
private readonly IValidator _validator;
public ValidationResult AddCardBlock(Card card, BlockDetails
blockDetails))
{
    var context = new CardBlockingContext(blockDetails);
    _validator.Validate(_cardBlockRuleSet, card, context);

    if (result.IsValid)
    {
        card.AddBlock(blockDetails);
    }

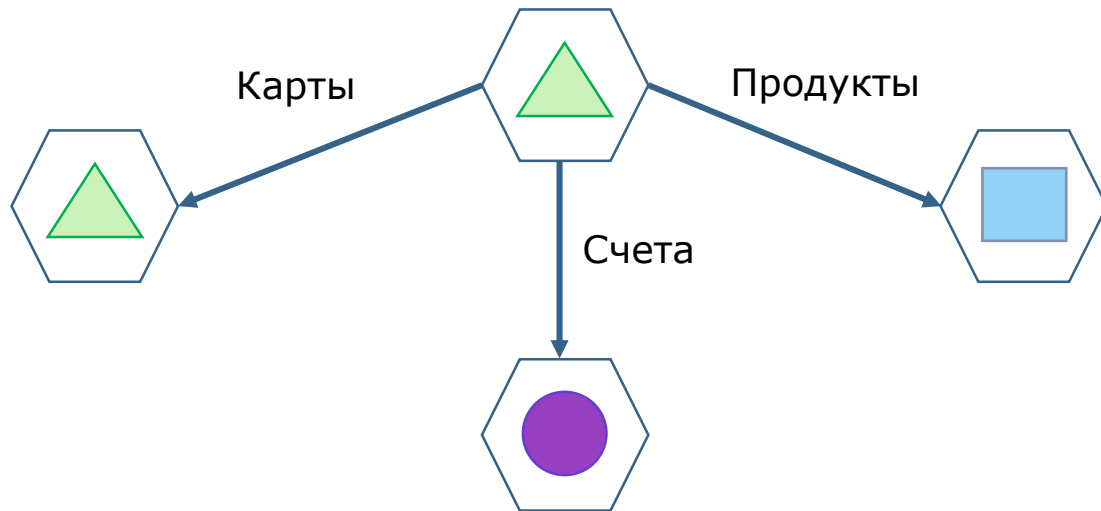
    return result;
}
```

Сложности внедрения



Проблема

В микросервисной системе чтение данных осуществить сложнее

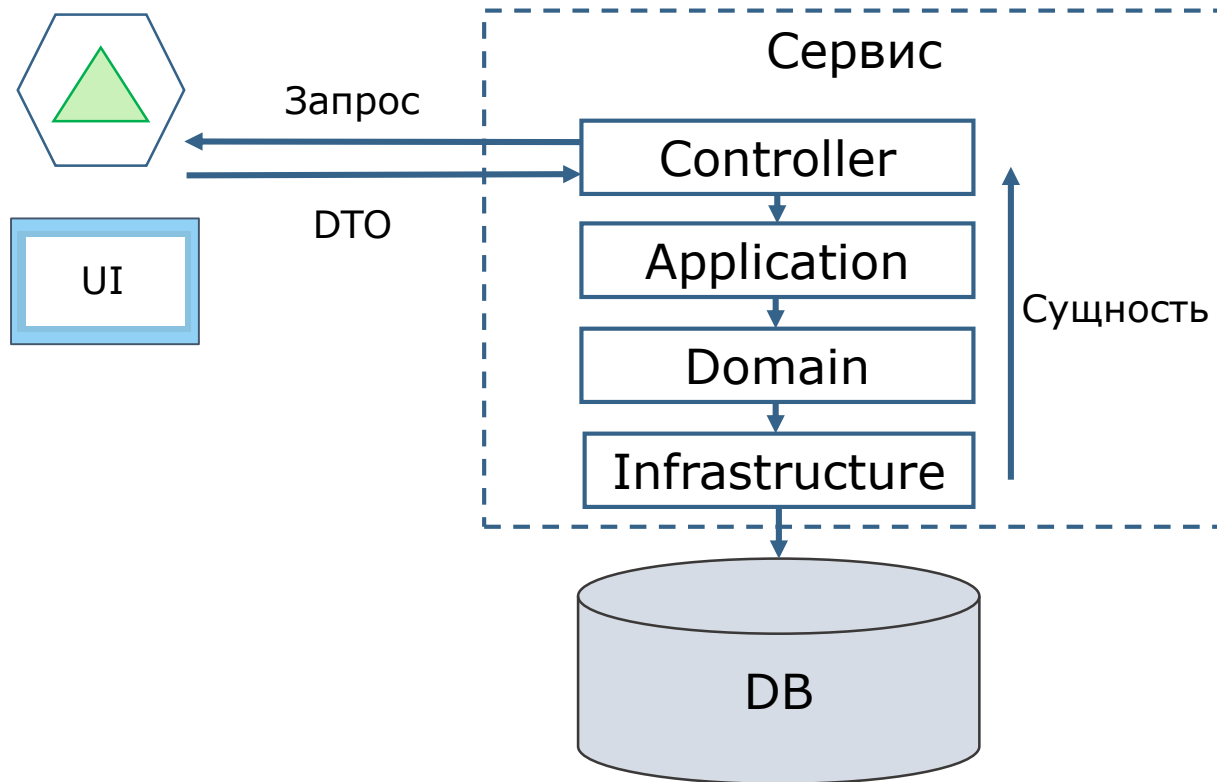


Запрос данных для UI

```
SELECT
    c.HOLDER,
    c.NUMBER,
    a.NUMBER
FROM CARDS AS c JOIN ACCOUNTS AS a ON
(CASE c.ACCOUNTTYPE
    WHEN 1 THEN 99
    WHEN 2 THEN 100
    ELSE c.ACCOUNTTYPE
    END) = a.TYPE
WHERE a.NUMBER = a.GENERALNUMBER AND c.Id = :id;
```

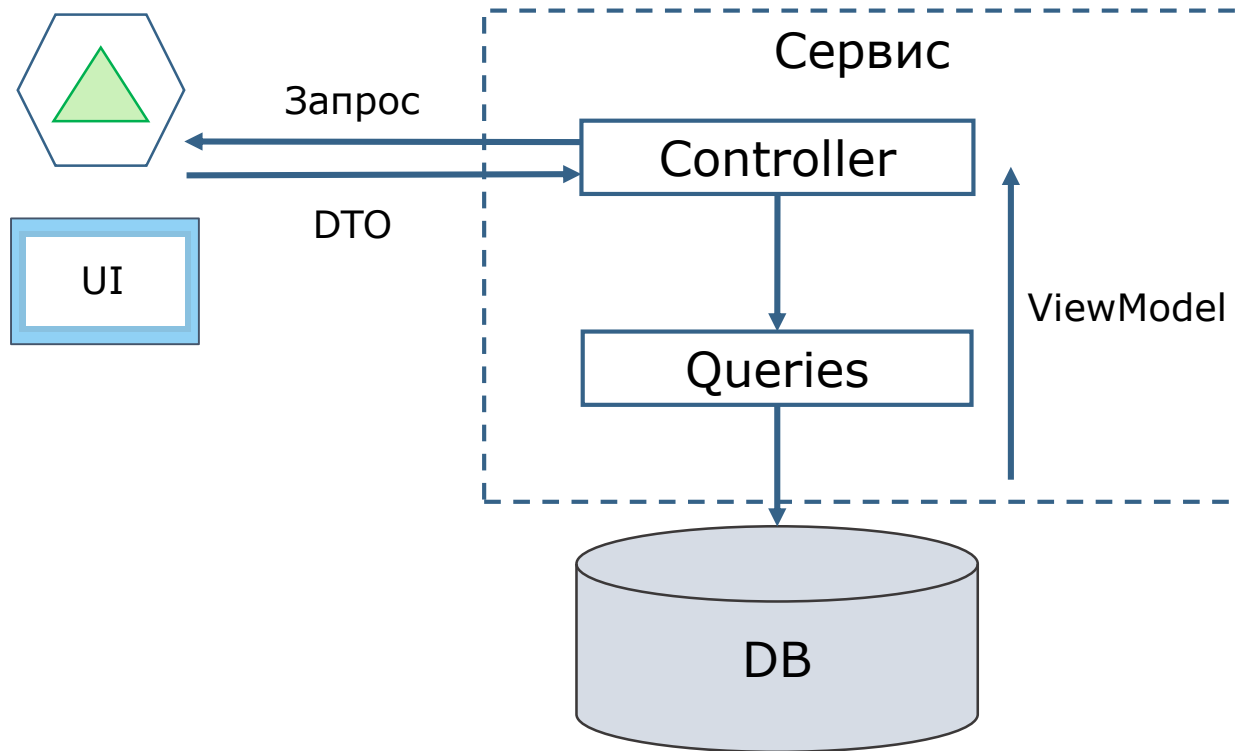
Реализация запросов

Для запросов использовались сущности модели



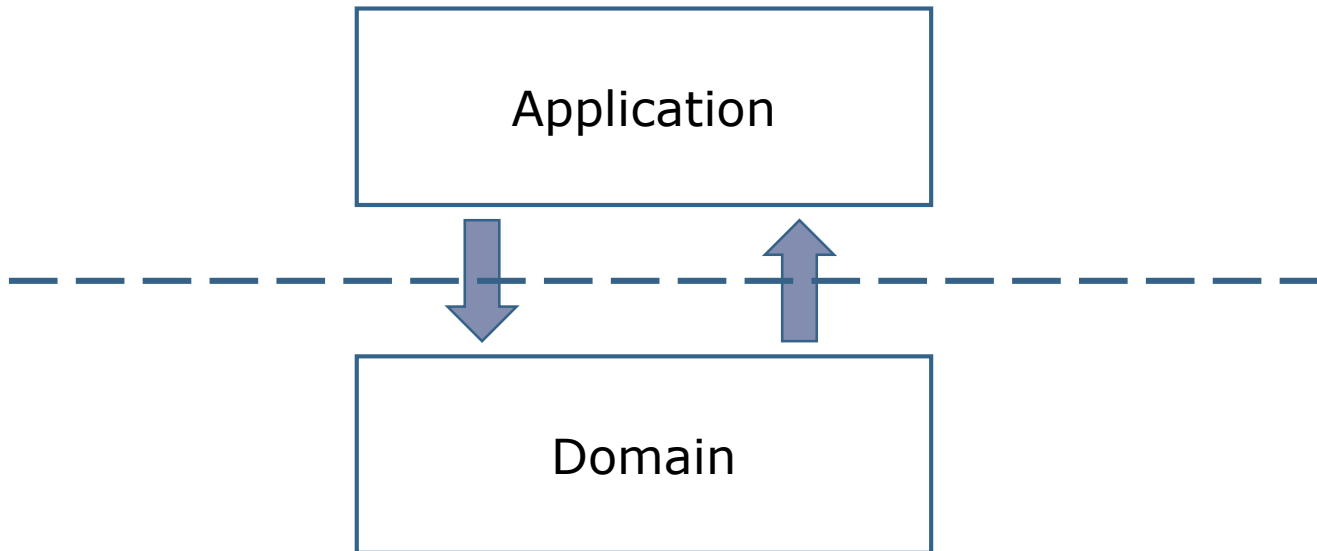
Реализация запросов

В итоге использовали отдельную подсистему чтения



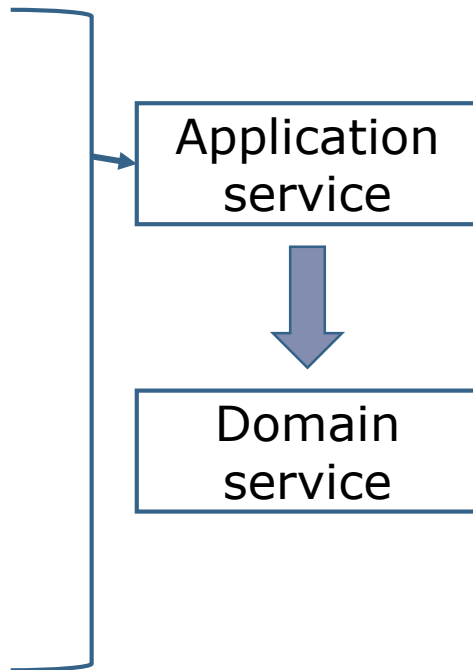
Проблема

Распределение логики между аппликационным и доменным слоем



Распределение логики

- Если нет принятия решений



Аппликационный сервис

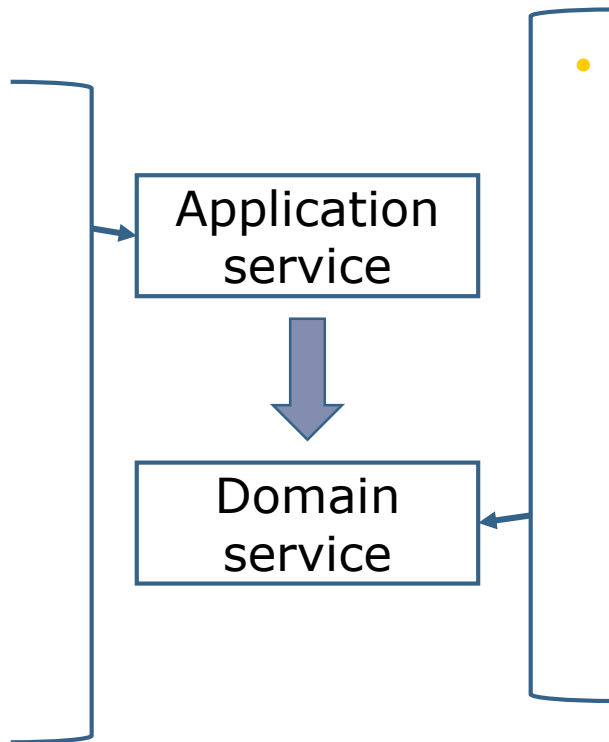
```
public BlockResult BlockCard(int cardId, BlockDto blockDto)
{
    if (cardId <= 0)
        return new BlockResult("Card id must be greater than zero");
    if (blockDto == null);
        return new BlockResult("Block details must not be null");

    var repository = _entityRepositoryFactory.Create<Card>();
    var card = repository.Get(cardId);
    if (card == null)
        return new BlockResult("Card does not exist");

    _accountsService.LockAccount(card);
    card.BlockCard(blockDto.ToDetails());
    ...
}
```

Распределение логики

- Если нет принятия решений



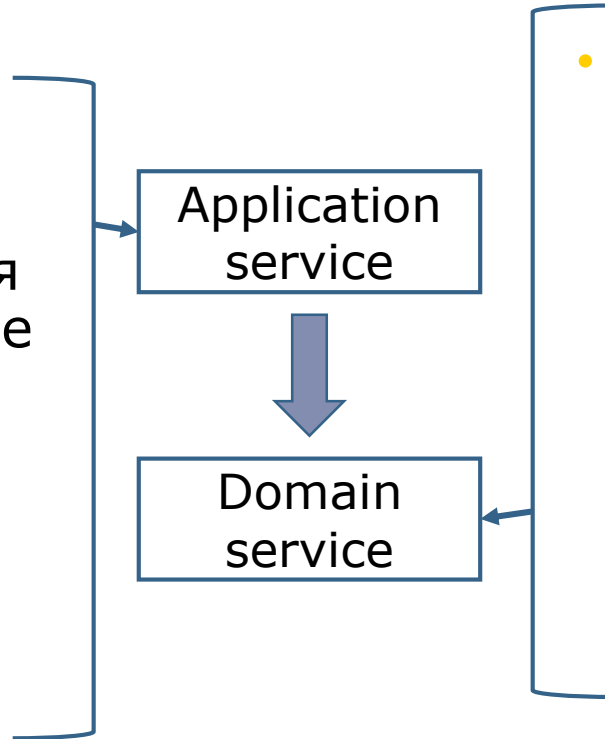
- Если есть решения, влияющие на доменную логику

Доменный сервис

```
public class BlockingService : IBlockingService
{
    ...
    public BlockResult BlockCard(Card card, BlockDetails blockDetails)
    {
        var result = _accountsService.LockAccount(card);
        if(result.HasErrors())
        {
            return new BlockResult(result.GetErrorMessage());
        }
        card.BlockCard(blockDetails);
    }
}
```


Распределение логики

- Если нет принятия решений
- Если не получается скрыть технические аспекты в интерфейсе



- Если есть решения, влияющие на доменную логику

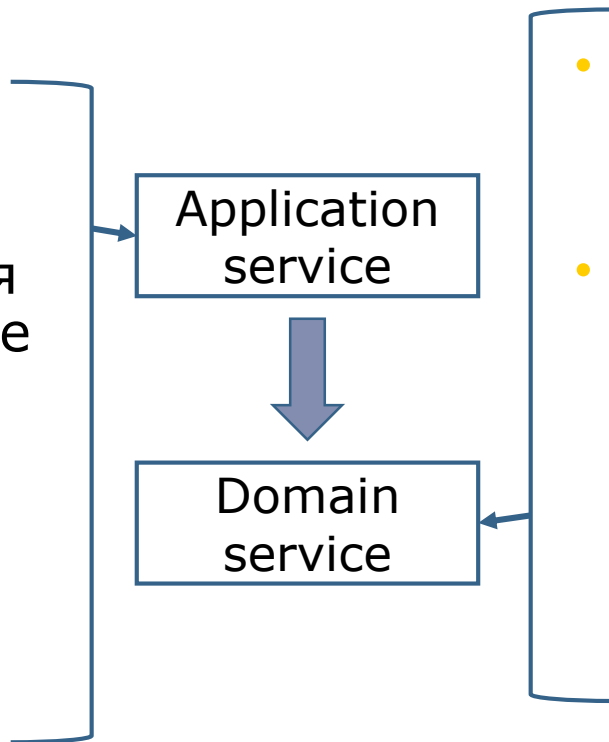
Аппликационный сервис

```
using(var session = _externalCardSessionFactory.CreateSession(card))
{
    var context = _cardContextFactory.Create(card);
    card.DoSomeStaff(context);

    if (card.HasSomeConditions())
    {
        var batchResult = session.Commit(card);
        if(batchResult.HasErrors())
        {
            card.DoErrorStaff(batchResult);
            return;
        }
        card.DoFinalStaff(batchResult);
    }
}
```

Распределение логики

- Если нет принятия решений
- Если не получается скрыть технические аспекты в интерфейсе



- Если есть решения, влияющие на доменную логику
- Если удастся использовать Separated Interface

Доменный сервис

```
public class BlockingService : IBlockingService
{
    ...
    public BlockResult BlockCard(Card card, BlockDetails blockDetails)
    {
        var result = _accountsService.LockAccount(card);
        if(result.HasErrors())
        {
            return new BlockResult(result);
        }
        card.BlockCard(blockDetails);
    }
}
```

Распределение логики

- Если нет принятия решений
- Если не получается скрыть технические аспекты в интерфейсе
- Технические оптимизации

Application
service



Domain
service

- Если есть решения, влияющие на доменную логику
- Если удастся использовать Separated Interface

Аппликационный класс

```
public AccountContext(IExternalService externalService,
IEnumerable<Account> accounts)
{
    _accountsInfo = new Lazy<IEnumerable<AccountInfo>>(() =>
        GetAccountsInfo(externalService, accounts));
}

public AccountInfo GetAccountInfo(int accountId)
{
    return _accountsInfo.Value.FirstOrDefault(x => x.Id == accountId);
}
```

Распределение логики

- Если нет принятия решений
- Если не получается скрыть технические аспекты в интерфейсе
- Технические оптимизации
- Любые транзакции

Application
service



Domain
service

- Если есть решения, влияющие на доменную логику
- Если удастся использовать Separated Interface

Аппликационный класс

```
using (var collector = _eventCollectorFactory.BeginCollection())
{
    using (var uow = _entityRepositoryFactory.Create())
    {
        var parser = _factory.Create(file);
        var parseResult = parser.Parse();
        uow.Commit();
        eventCollector.Send();
    }
}
```


Распределение логики

- Если нет принятия решений
- Если не получается скрыть технические аспекты в интерфейсе
- Технические оптимизации
- Любые транзакции

Application
service

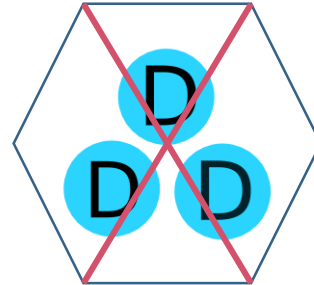
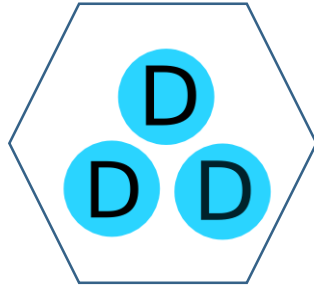
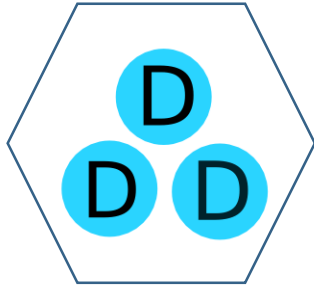


Domain
service

- Если есть решения, влияющие на доменную логику
- Если удастся использовать Separated Interface
- Валидация на основе доменных сущностей и внешних данных

Проблема

Не все микросервисы требуют DDD

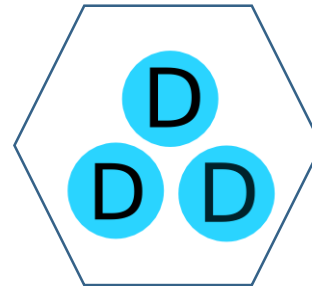


Как мы выбираем подход

- У сервиса есть сложная бизнес-логика, валидации

&&

- Сервис управляет сущностями бизнес-домена

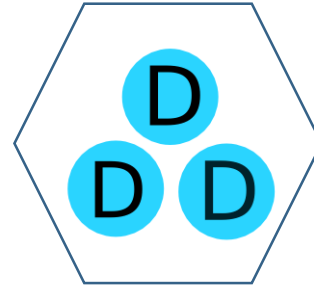


Как мы выбираем подход

- У сервиса есть только CRUD при работе с хранилищем

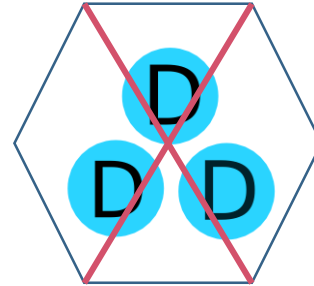
&&

- Сервис управляет сущностями бизнес-домена



Как мы выбираем подход

- У сервиса нет работы с хранилищем

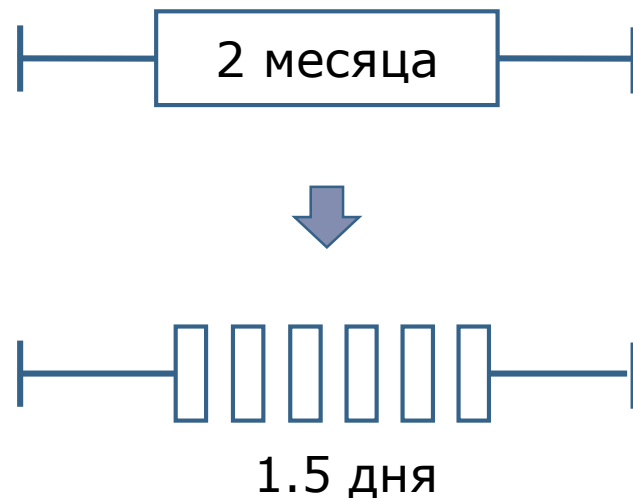
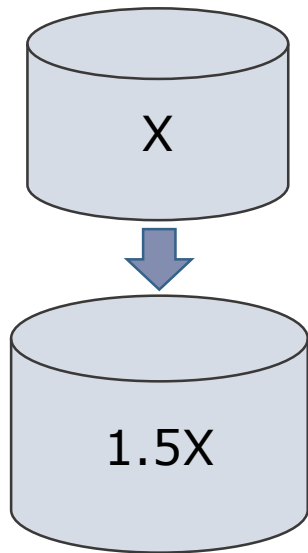


Что в итоге?



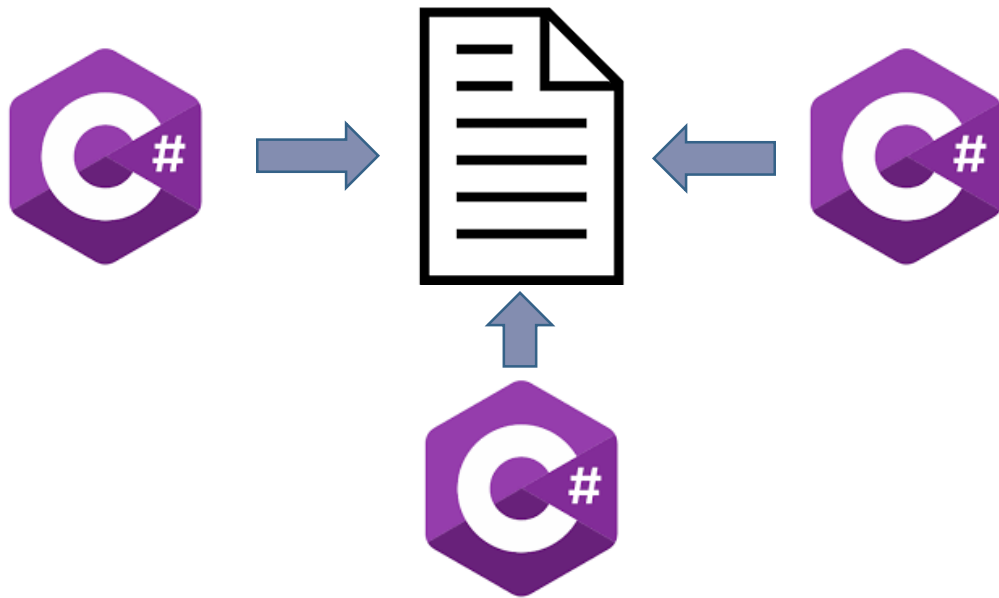
Результаты

Стабильная выдача задач



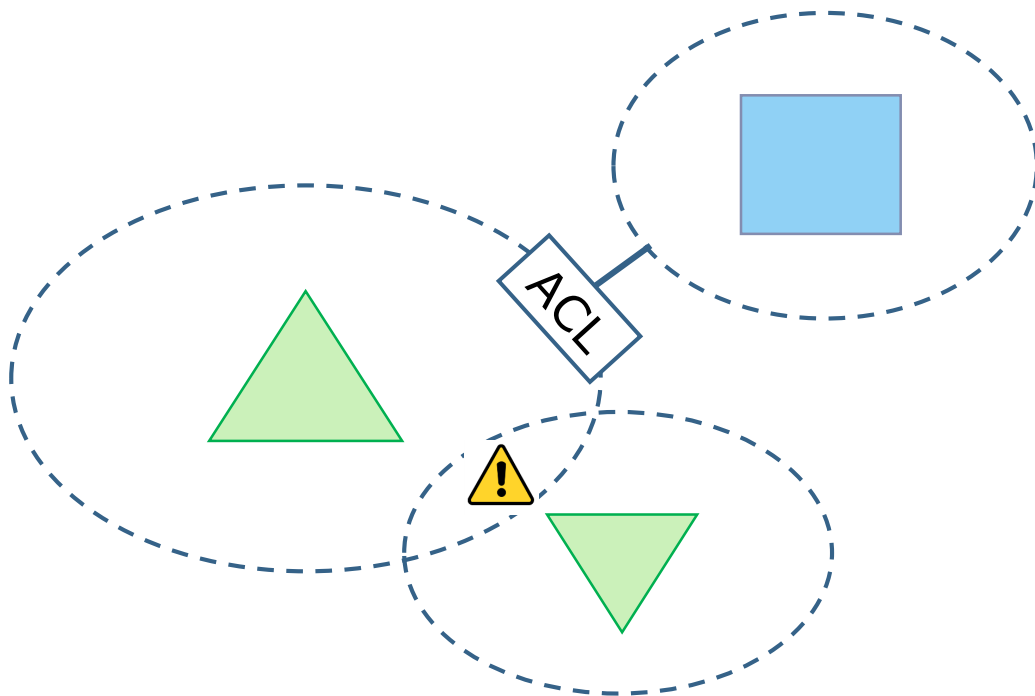
Результаты

Модель аналитиков совпадает с моделью в коде



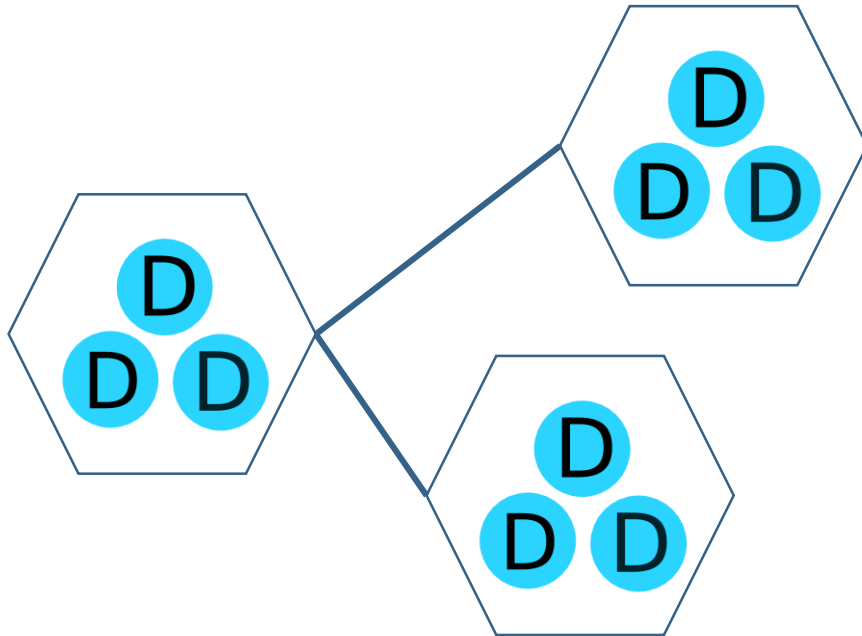
Результаты

Обозначили и разделили ограниченные контексты



Результаты

Микросервисы имеют сходную структуру



Результаты

Общение стало проще



Резюме

- Domain driven design дает четкое понимание бизнес-процессов для всех участников

Резюме

- Domain driven design дает четкое понимание бизнес-процессов для всех участников
- Основа успешной разработки на основе модели - стратегическое проектирование

Резюме

- Domain driven design дает четкое понимание бизнес-процессов для всех участников
- Основа успешной разработки на основе модели - стратегическое проектирование
- DDD в сочетании с микросервисами дают четкие границы контекстов

Резюме

- Domain driven design дает четкое понимание бизнес-процессов для всех участников
- Основа успешной разработки на основе модели - стратегическое проектирование
- DDD в сочетании с микросервисами дают четкие границы контекстов
- DDD – дорогой метод, микросервисы делают его еще дороже

Резюме

- Domain driven design дает четкое понимание бизнес-процессов для всех участников
- Основа успешной разработки на основе модели - стратегическое проектирование
- DDD в сочетании с микросервисами дают четкие границы контекстов
- DDD – дорогой метод, микросервисы делают его еще дороже
- Тактические шаблоны проектирования позволяют сделать код понятным и добиться эволюционного дизайна

Спасибо!

konst.gustov@gmail.com