Восьмая независимая
научно-практическая конференция
«Разработка ПО 2012»

1 - 2 ноября, Москва

2012 CEE-SEC(R)
Software Engineering Conference in Russia

CQRS: Архитектура, которая делает бизнес-приложения лучше

Ксения Мухортова

Intel

Проблемы разработчика бизнес-приложений

- Предметная область - чужой мир
- Частое изменение требований
- Не- или слабоструктурированные данные

Технические проблемы

- Изменения требований затрагивают всё приложение
- Эволюция технологий
- Поддержка нескольких типов UI
- Несоответствие моделей хранения – логики – отображения
- AJAX: Необходимость передавать и view и данные

Опять проблемы?

- Меняешь доменную модель – переделай маппинги
- Масштабируемость и синхронизация
- Бизнес-намерения утеряны при сохранении данных

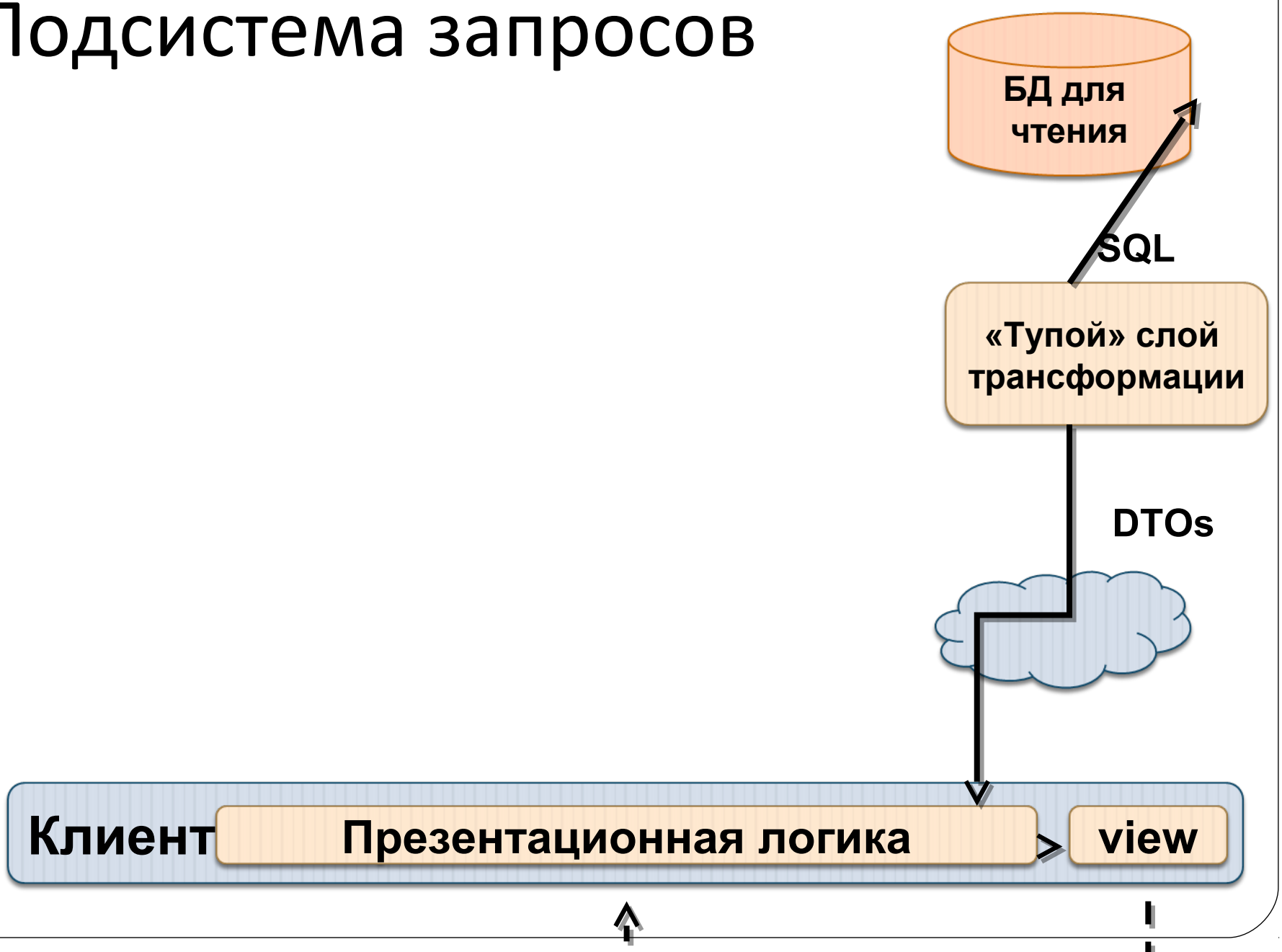
ЖУТКО СЛОЖНО!!!

CQS -> CQRS

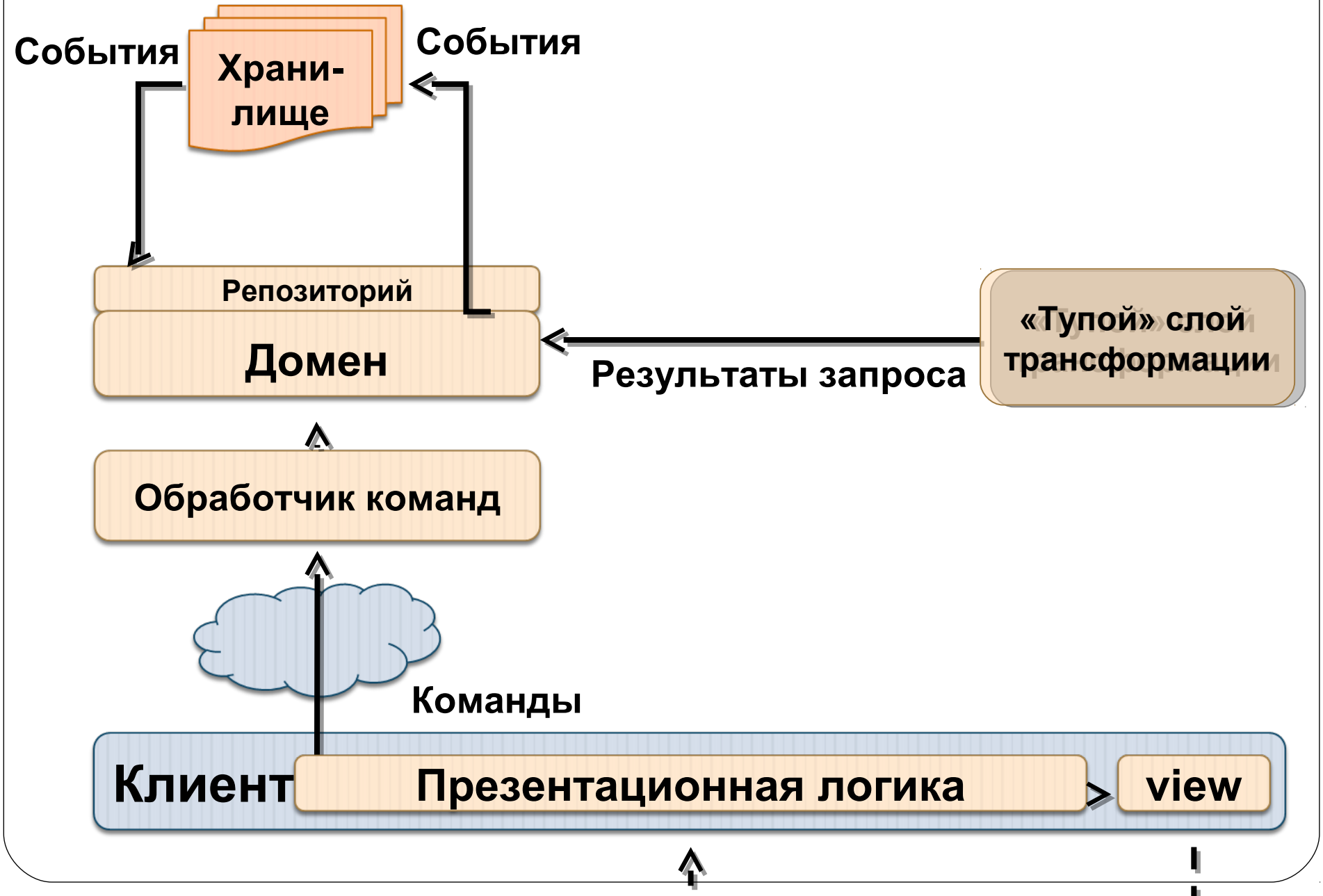
- Command: команда выполняет действие, не возвращает результат

- Query: запрос возвращает результат, не выполняет действий

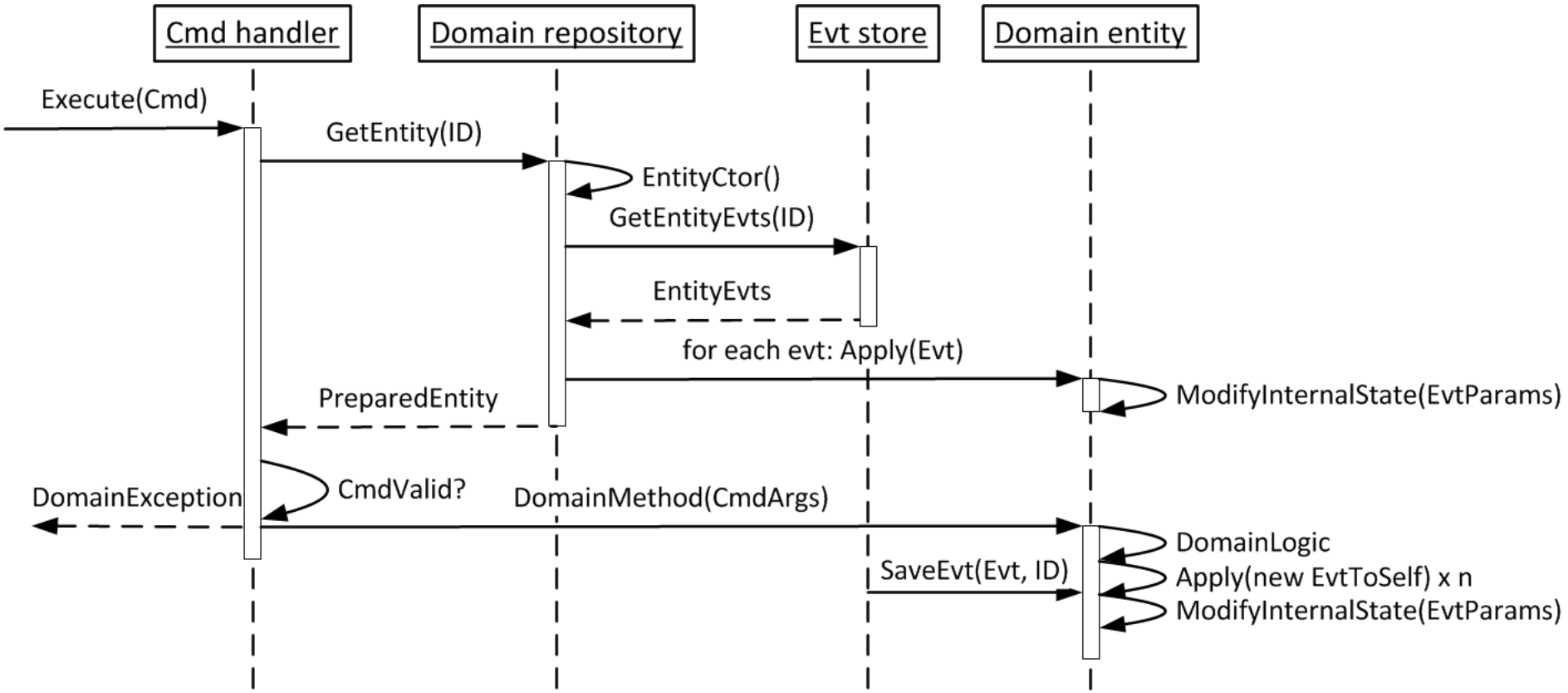
Подсистема запросов



Подсистема команд



Event sourcing



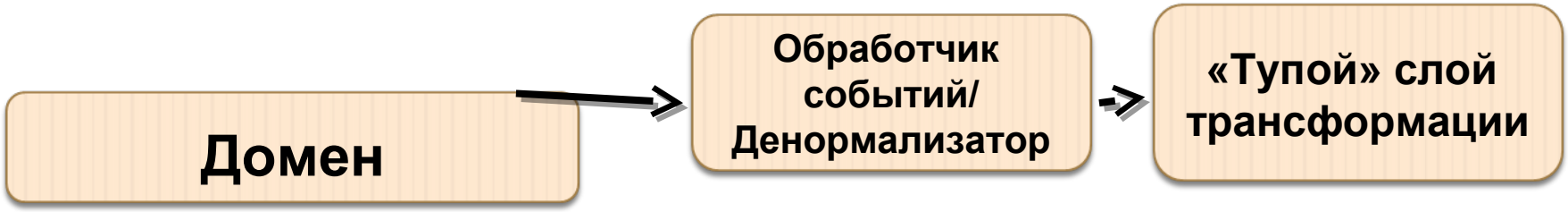
Команда

- Указание к действию
- Возможны исключения
- Может затрагивать один или несколько агрегатов

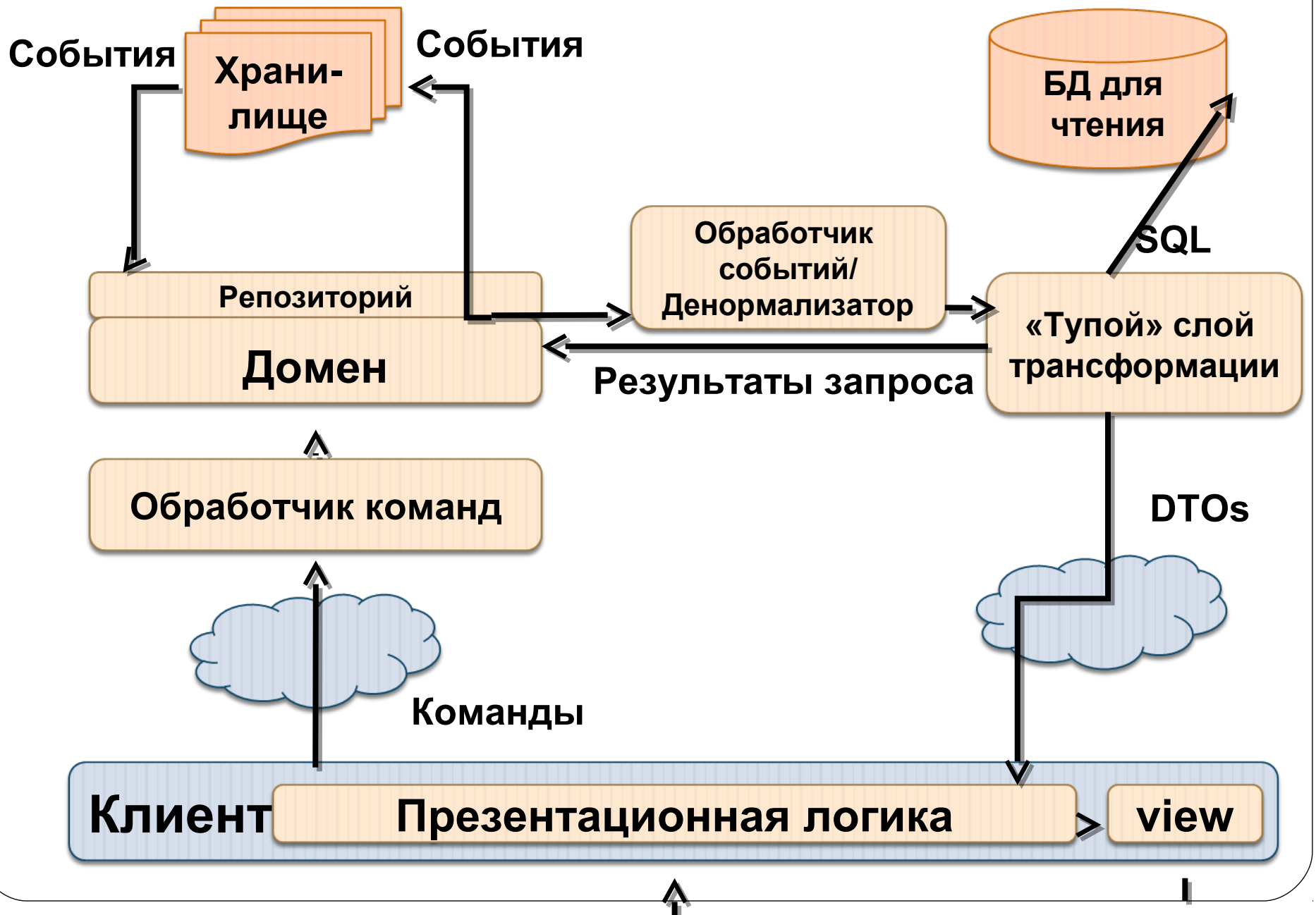
Событие

- Свершившийся факт
- Исключения невозможны
- Относится к одному агрегату

Подсистема синхронизации



Всё вместе


События
Храни-
лище
События
БД для
чтения
SQL
Обработчик
событий/
Денормализатор
Репозиторий
Домен
«Тупой» слой
трансформации
Результаты запроса
Обработчик команд
DTOs
Команды
Клиент
Презентационная логика
view

Преимущества event sourcing

- Хранятся только события – объективные факты
- Независимость структуры доменных объектов от модели хранения
- Возможность ретроспективной функциональности
- Тестирование поведения, а не состояния

Тестируем поведение (BDD)

Каждый сценарий описывается тремя категориями:

- Given:

  инстанциируем объект, устанавливаем состояние проигрывая цепочку событий

- When:

вызываем команду

- Then:

проверяем сгенерированные события

BDD заставляет мыслить в терминах функциональности

Универсальное решение?

Важно запомнить про CQRS

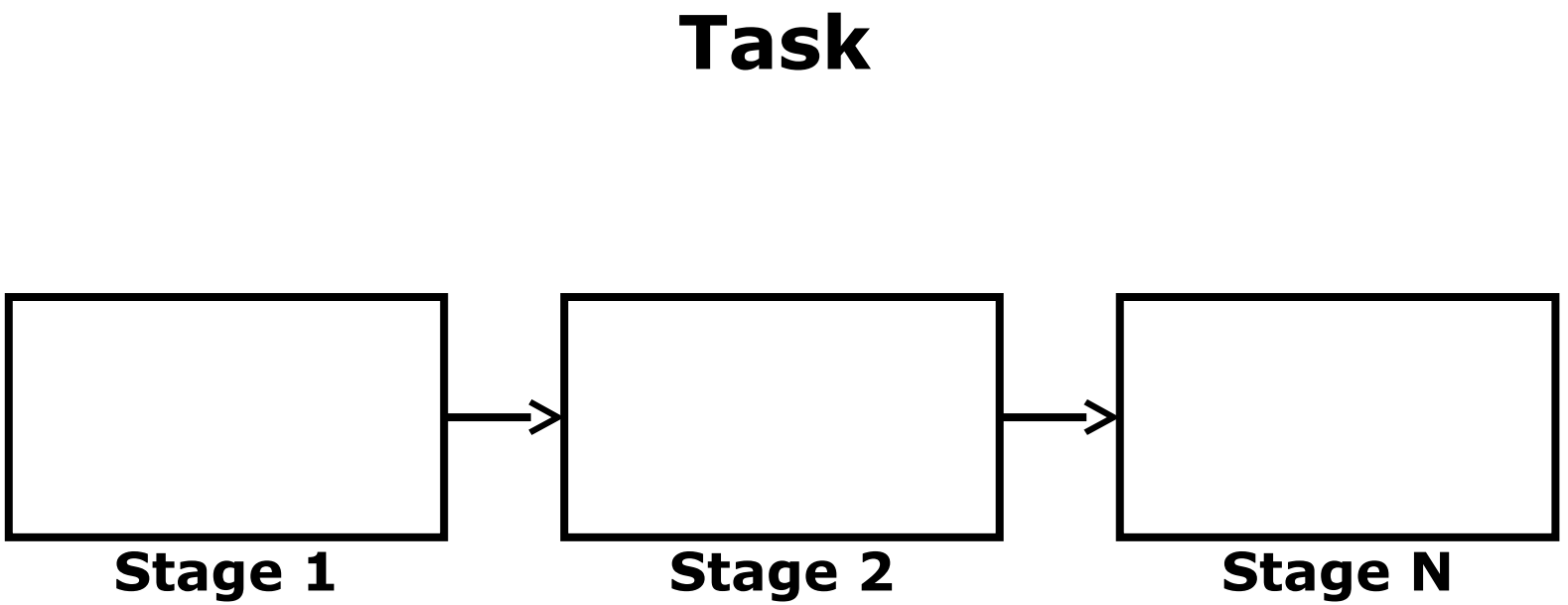
- “Правильный” DDD – залог успеха
  - Aggregates (eventual vs transactional consistency)
  - Bounded contexts
  - Моделирование поведения, а не реального мира
- Отличия между командами и событиями
- ORM в write части – ЗЛО
- Unit-тестируйте поведение, а не состояние

Источники информации

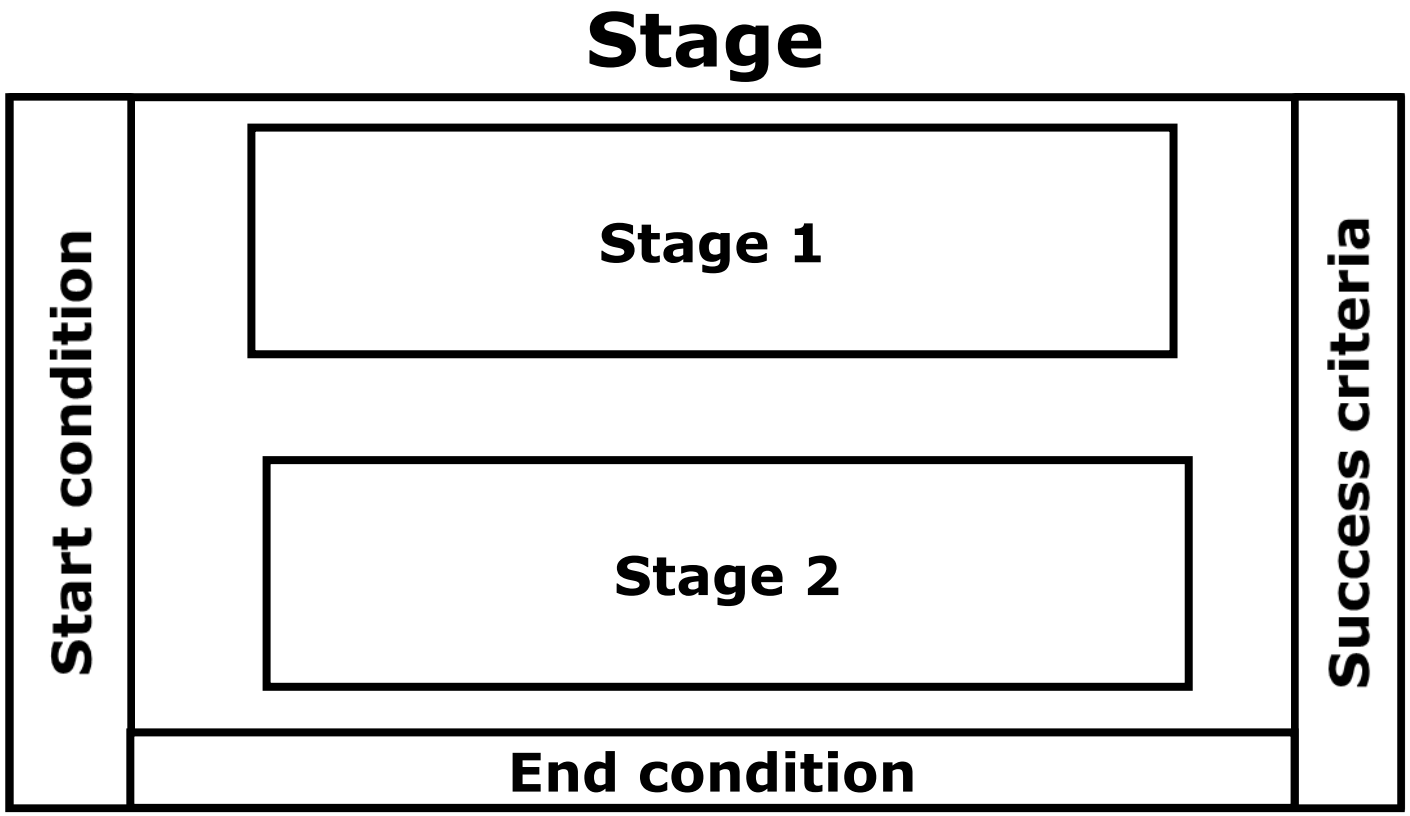
- Greg Young, Udi Dahan, Eric Evans, Rinat Abdullin
- Microsoft CQRS journey: http://msdn.microsoft.com/en-us/library/jj554200.aspx

Пример

Предметная область



Предметная область



Отправка команды добавления задачи на WCF сервис

```
private void AddNewTask()
{
    var command = new AddNewTaskCommand
                      {
                          TaskName = view.TaskName,
                          Stages = new Stage[] {new PackagingStage {}}
                      };
    service.ExecuteCommand(command);
}
```

Интерфейс WCF сервиса

```
[ServiceContract]
public interface ICommandService
{
    [OperationContract]
    [XmlInclude(typeof (AddAutoProcessCommand))]
    [XmlInclude(typeof (AddManualProcessCommand))]
    [XmlInclude(typeof (AddNewTaskCommand))]
    [XmlInclude(typeof (AddNewTestBuildTaskCommand))]
    [XmlInclude(typeof (AutomationStopProcessCommand))]
    [XmlInclude(typeof (CompleteAutoProcessCommand))]
    [XmlInclude(typeof (CompleteManualProcessCommand))]
    [XmlInclude(typeof(CompleteMarkedStageCommand))]
    [XmlInclude(typeof (FinishCurrentStageCommand))]
    [XmlInclude(typeof (MarkForFinishStageCommand))]
    [XmlInclude(typeof (PrepareAutoProcessCommand))]
    [XmlInclude(typeof (ProcessAutomationCompleteCommand))]
    [XmlInclude(typeof (ProcessAutomationWaitCommand))]
    [XmlInclude(typeof (RerunAutoProcessCommand))]
    [XmlInclude(typeof (RerunManualProcessCommand))]
                    ...
    [XmlInclude(typeof (PackagingStage))]
    [XmlInclude(typeof (ReplicationStage))]
    void ExecuteCommand(Command command);
}
```

Команда добавления задачи

```
[DataContract]
[KnownType(typeof(PackagingStage))]
public class AddNewTaskCommand : Command
{
    [DataMember]
    public readonly IEnumerable<Stage> Stages;

    [DataMember]
    public string TaskName { get; private set; }

    public AddNewTaskCommand()
        : base(Guid.NewGuid())
    {
    }

    public AddNewTaskCommand(string name, IEnumerable<Stage> stages)
        : this()
    {
        Stages = stages;
        TaskName = name;
    }
}
```

Обработчик команды добавления задачи

```
public class AddNewTaskCommandHandler : ICommandHandler<AddNewTaskCommand>
{
    private readonly IDomainRepository<IDomainEvent> repository;

    public AddNewTaskCommandHandler(IDomainRepository<IDomainEvent> repository)
    {
        this.repository = repository;
    }









}
```

Доменный объект “задача”

```
public class Task : BaseAggregateRoot<IDomainEvent>, IOrginator
{
    public string Name { get; set; }

    public TaskStatus Status { get; set; }

    public ExecutionResult Result { get; set; }

    public EntityList<Stage, IDomainEvent> Stages { get; protected set; }
```

Обработчик события добавления задачи

```
public class NewTaskAddedEventHandler : IEventHandler<NewTaskAddedEvent>
{
    private readonly IAggregateReadRepository<NotStartedTasksReport> notStartedTasksRepository;
    private readonly ITaskReadRepository<TaskDetailsReport> taskDetailsRepository;
    private readonly IAggregateReadRepository<TasksReport> taskRepository;

    public NewTaskAddedEventHandler(IAggregateReadRepository<TasksReport> taskRepository,
                                    IAggregateReadRepository<NotStartedTasksReport> notStartedTask
                                    ITaskReadRepository<TaskDetailsReport> taskDetailsRepository)
    {
        this.taskRepository = taskRepository;
        this.notStartedTasksRepository = notStartedTasksRepository;
        this.taskDetailsRepository = taskDetailsRepository;
    }

    #region IEventHandler<NewTaskAddedEvent> Members

    public void Execute(NewTaskAddedEvent theEvent)...

    #endregion
}
```

```
public void Execute(NewTaskAddedEvent theEvent)
{
    var tasksReport = new TasksReport
                      {Id = theEvent.TaskId, TaskName = theEvent.TaskName, TaskStatus = TaskStatus.NotStarted};
    taskRepository.Save(tasksReport);
```

Id	Name	Status
1	First task	NotStarted
3	Third task	InProgress

```
, TaskName = theEvent.TaskName};
```

```
                ent.TaskId,
                Event.TaskName,
                TaskStatus = TaskStatus.NotStarted,
                StageName = stage.Name,
                StageStatus = stage.Status
            };
    taskDetailsRepository.Save(taskDetailsReport);
}
}
```

Id	Task name	Task Status	Stage name	Stage status
1	First task	NotStarted	First stage	NotStarted
1	First task	NotStarted	Second stage	NotStarted

Получение списка задач

```
private void GetTasks(IAggregateReadRepository<TasksReport> repository)
{
    view.Tasks = repository.GetAll();
}
```

```
public partial class TasksReport : IAggregateDto
{
}
```

Тестирование

```
[TestClass]
public class When_starting_task : CommandTestFixture<StartTaskCommand, StartTaskCommandHandler, Task>
{
    private readonly Guid taskId = Guid.NewGuid();

    protected override IEnumerable<IDomainEvent> Given()
    {
        var stages = new List<Stage>
                        {
                            new PackagingStage(new PositiveAllSuccessCriteria(), new EndAllEndCondition()
                        };
        yield return PrepareDomainEvent.Set(
            new NewTaskAddedEvent(taskId, "NewTask", new StartNowCondition(), stages)).ToVersion(1);
    }
```

```
[TestClass]
public class When_task_was_started : EventTestFixture<TaskStartedEvent, TaskStartedEventHandler>
{
    private static readonly Guid TaskId = Guid.NewGuid();
    private TasksReport updateTaskObject;

    protected override void SetupDependencies()
    {
        OnDependency<IAggregateReadRepository<TasksReport>>()
            .Setup(x => x.GetById(It.IsAny<Guid>()))
            .Returns(new TasksReport { Id = TaskId, TaskName = "Test Task" });

        OnDependency<IAggregateReadRepository<TasksReport>>()
            .Setup(x => x.Update(It.IsAny<TasksReport>()))
            .Callback<TasksReport>(a => updateTaskObject = a);
    }
```