

Статический анализ кода

(на примере DDD-фреймворка)

Алексеев Алексей
alekseev.aleksei@gmail.com
aalekseev@custis.ru

Николай Гребнев
ngrebnev@gmail.com
ngrebnev@custis.ru

CUSTIS®

Структура доклада

- Введение
- Статические проверки
- DDD-фреймворк
- Поддержка LINQ
- Модель состояний
- Верификация модели состояний

Структура доклада

- Введение
- Статические проверки
- DDD-фреймворк
- Поддержка LINQ
- Модель состояний
- Верификация модели состояний

Человеческий фактор



Общепринятые методологии

- Ручное тестирование
- Автоматическое тестирование, TDD
- Code Review

Средства разработки



Структура доклада

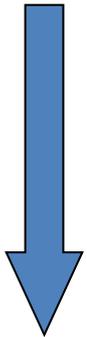
- Введение
- **Статические проверки**
- ДДД-фреймворк
- Поддержка LINQ
- Модель состояний
- Верификация модели состояний

Стоимость исправления ошибок

Момент выявления ошибки:

- До написания кода
- Статические проверки
- Unit-тесты
- Code Review
- Интеграционные тесты
- Ручные тесты
- Ошибка при эксплуатации

Цена

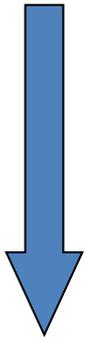


Стоимость исправления ошибок

Время выявления ошибки:

- До написания кода
- Статические проверки
- Unit-тесты
- Code Review
- Интеграционные тесты
- Ручные тесты
- Ошибка при эксплуатации

Цена



Аспекты статических проверок

- Диагностика
- Скорость
- Полнота

Полнота статических проверок

C++: `if (a = 2)`

`if (ptr == null)`

Корректность

VS

`if (ptr)`

Лаконичность

Структура доклада

- Введение
- Статические проверки
- **DDD-фреймворк**
- Поддержка LINQ
- Модель состояний
- Верификация модели состояний

Терминология

- ORM – object relational mapper:
 - Отображение:
 - Класс → таблица
 - Объект → запись
 - Свойство → колонка
 - Запросы
 - Процесс компиляции

Демонстрация

ВАЛИДАЦИЯ МОДЕЛИ ВО ВРЕМЯ КОМПИЛЯЦИИ

```
Bill
CustIS.UniNet;
CustIS.UniNet.Markup;
CustIS.UniNet.Metadata;
```

Space Samples

```
</summary> Сущность - накладная. </summary>
[EntityType(Abtract = true)]
[TableStorage(Table Name = "t_bill", EntityTypeColumnName = "type")]
public abstract class Bill : Entity

/// <summary> Идентификатор. </summary>
[Attr]
[Column("id_bill")]
public abstract long Id { get; }

/// <summary> Состояние накладной. </summary>
[Attr(RequireKind = AttrRequireKind.Mandatory, InitialValue = BillState.New)]
public abstract BillState State { get; set; }

/// <summary> Начало обработки. </summary>
[StateRestriction(BillState.New)]
[StateTransition(BillState.New, BillState.InWork)]
public virtual void StartWork()
{
    State = BillState.InWork;
}
```

Структура доклада

- Введение
- Статические проверки
- DDD-фреймворк
- **Поддержка LINQ**
- Модель состояний
- Верификация модели состояний

Полнота поддержки LINQ напрямую влияет на снижение влияния человеческого фактора в разработке ПО

ПОДДЕРЖКА LINQ

Запросы к доменной модели

- На языке ORM:

```
SimpleQuery<Post> q =  
    new SimpleQuery<Post>(  
        @"from Post p  
        where p.Blog.Author = ?",  
        author);  
return q.Execute();
```

- LINQ:

```
from Post p in Session.Posts  
where p.Blog.Author == author  
select p;
```

Преимущества LINQ

- Статическая типизация
- IntelliSense
- Полная интеграция в язык программирования

Максимальная типизация в Linq

Требуется:

```
Employee leader = Session.Employee.First();  
var q = from Department d in Session.Department  
        where d.Leader == leader  
        select d;
```

Но в Entity Framework:

Не удалось создать константу с типом "Тип замыкания". В этом контексте поддерживаются только типы-примитивы ("например Int32, String и Guid").

Свойства, используемые в запросах

```
public class Employee
{
    ...
    public bool IsManager
    {
        get { return Subordinates.Count() > 0; }
    }
    ...
}

...
var q = from Employee e in Session.Employee
        where e.IsManager
        select e;
q.ToList();
```

В Entity Framework:

Указанный член типа "IsManager" не поддерживается в выражениях LINQ to Entities. Поддерживаются только инициализаторы, члены сущности и свойства навигации сущности.

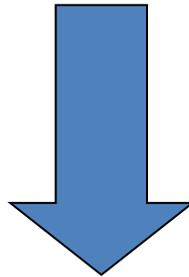
Решение

```
public class Employee
{
    ...
    [Attr]
    [Implemented]
    public abstract bool IsManager {get; }

    // Это реализация для атрибута IsManager.
    static Expression<Func<Employee, bool>> IsManagerImpl
    {
        get { return e => Subordinates.Count() > 0; }
    }
    ...
}
...
var q = from Employee e in Session.Employee
        where e.IsManager
        select e;
q.ToList();
```

Свойства, используемые в запросах

```
from Employee e in Session  
select e.IsManager
```



```
from Employee e in Session  
select Subordinates.Count() > 0
```

Корректность

```
[Attr]
[Implemented]
public abstract MyEntity Attr1 {get; }
```

```
[Attr]
[Implemented]
public abstract MyEntity Attr2 {get; }
```

```
static Expression<Func<MyEntity, MyEntity>> Attr1Impl
{
    get {return e => e.Attr2; }
}
```

```
static Expression<Func<MyEntity, MyEntity>> Attr2Impl
{
    get {return e => e.Attr1; }
}
```

Пример анализа реализации

```
static Expression<Func<MyEntity, MyEntity>>  
    Attr1Impl  
{  
    get {return e => e.Attr2;} } →
```

Обнаружена циклическая
зависимость

```
static Expression<Func<MyEntity, MyEntity>>  
    Attr2Impl  
{  
    get {return e => e.Attr1;} }  
}
```

Структура доклада

- Введение
- Статические проверки
- Валидация модели во время компиляции
- Поддержка LINQ
- **Модель состояний**
- Верификация модели состояний

Состояния

```
/// <summary> Состояние автомобиля. </summary>
[Flags]
[State]
public enum AutoState
{
    /// <summary> Машина стоит и не заведена.
    </summary>
    Stopped = 1,

    /// <summary> Машина заведена и не едет.
    </summary>
    Winded = 2,

    /// <summary> Машина едет. </summary>
    Driving = 4,
}
```

Императивные проверки

```
[Method]
public virtual void WindUp()
{
    if (State != AutoState.Stopped)
        throw new InvalidEntityStateException(...);
    ...
}
```

```
[Method]
public virtual bool TryRun()
{
    if (State != AutoState.Winded)
        throw new InvalidEntityStateException(...);
    ...
}
```

Декларативные ограничения

```
[Method]
```

```
[StateRestriction (AutoState.Stopped) ]
```

```
public virtual void WindUp ()
```

```
{...}
```

```
[Method]
```

```
[StateRestriction (AutoState.Winded) ]
```

```
public virtual bool TryRun ()
```

```
{...}
```

Декларативные ограничения

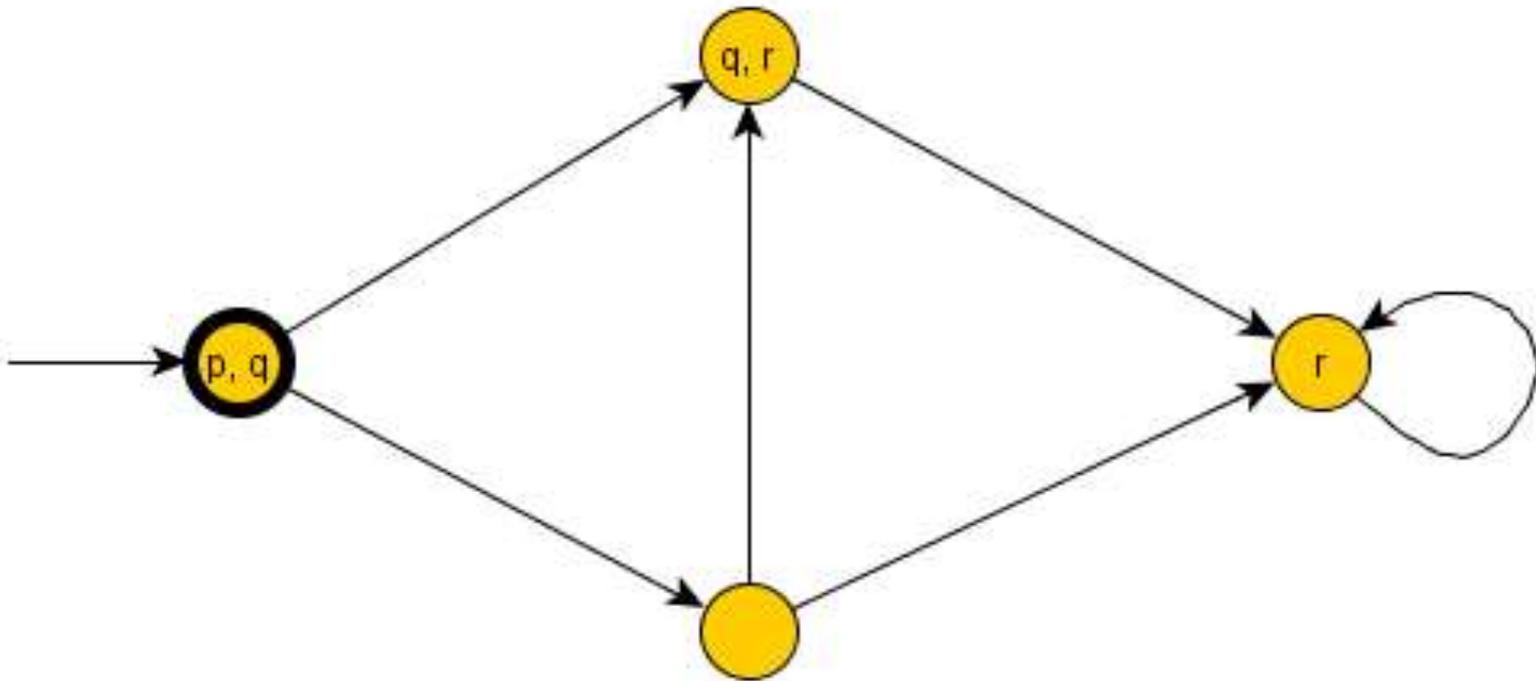
```
[Method]
[StateRestriction (AutoState.Stopped) ]
[StateTransition (AutoState.Stopped,
    AutoState.Stopped | AutoState.Winded) ]
public virtual void WindUp()
{...}
```

```
[Method]
[StateRestriction (AutoState.Winded) ]
[StateTransition (AutoState.Winded,
    AutoState.Driving | AutoState.Stopped) ]
public virtual bool TryRun()
{...}
```

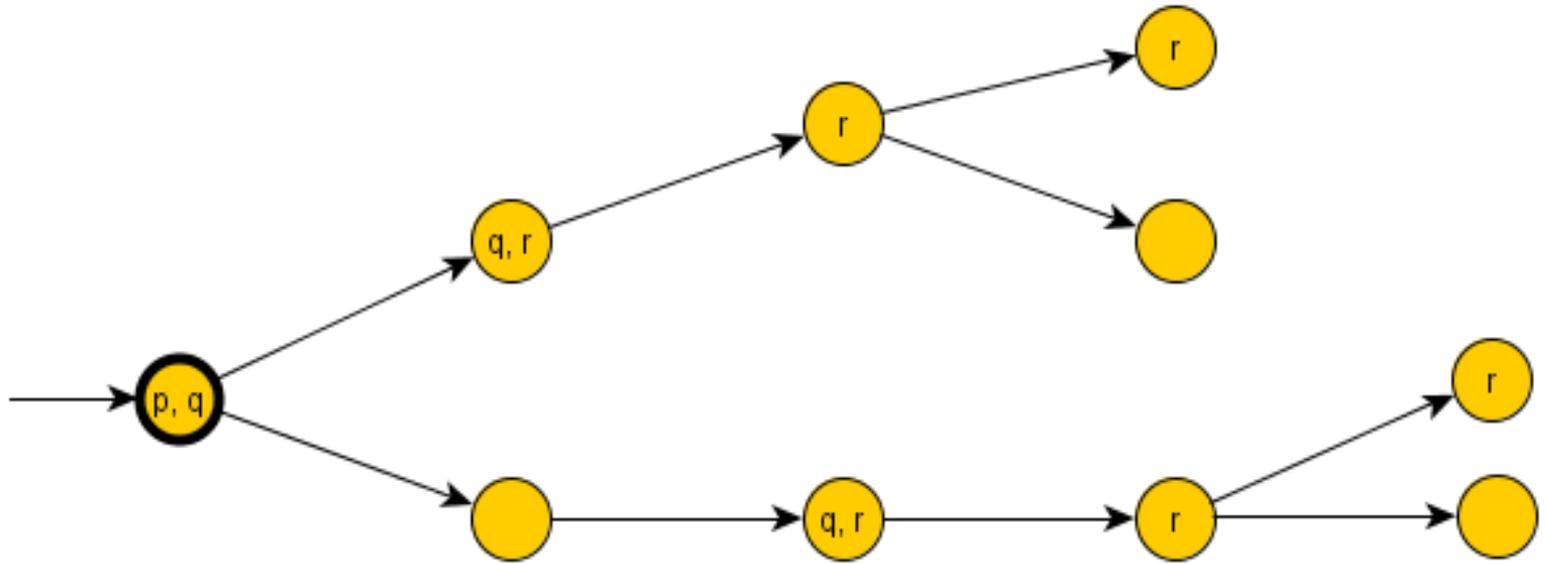
Структура доклада

- Введение
- Статические проверки
- Валидация модели во время компиляции
- Поддержка LINQ
- Модель состояний
- **Верификация модели состояний**

Структура Крипке



Структура Крипке



CTL, формулы состояний

CTL - Computation tree logic.

Формулы состояний:

- A **f** - All: **f** должен выполняться на всех путях из данного состояния;
- E **f** - Exists: существует хотя бы один путь из данного состояния, на котором выполняется **f**.

В этом определении **f** – формула пути.

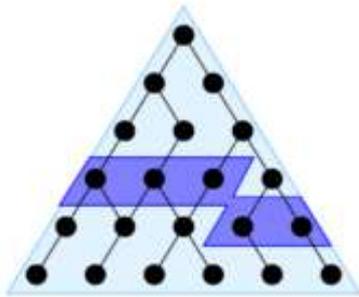
CTL, формулы пути

Формулы пути:

- $\underline{X} p$ - **Next**: p выполняется на следующем состоянии пути;
- $\underline{G} p$ - **Globally**: p выполняется на всех последующих состояниях пути;
- $\underline{F} p$ - **Finally** p выполняется на одном из последующих состояний пути;
- $p \underline{U} q$ - **Until**: p выполняется, пока на каком-то из состояний пути не выполнится q , причем q должен обязательно когда-нибудь выполниться в будущем.
- p – формула состояния или предикат

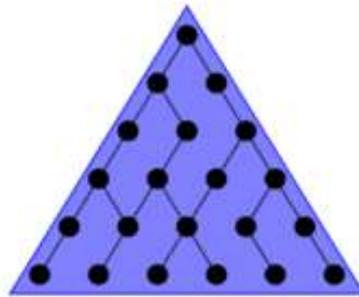
CTL

finally P



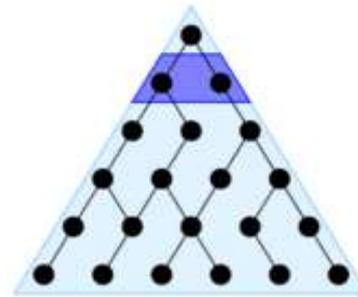
$AF P$

globally P



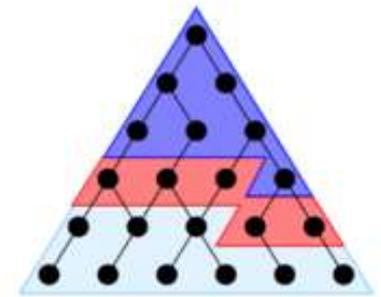
$AG P$

next P

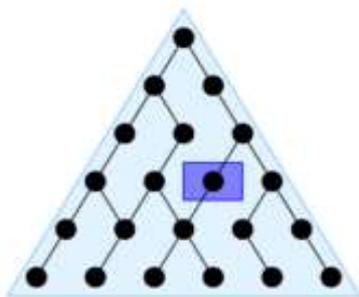


$AX P$

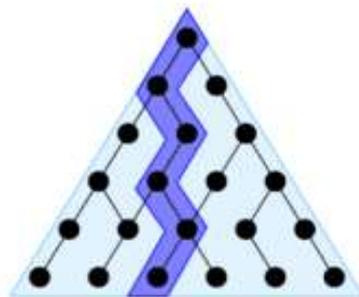
P until q



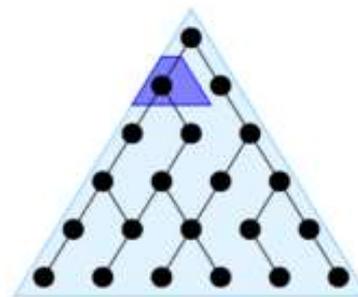
$A[P U q]$



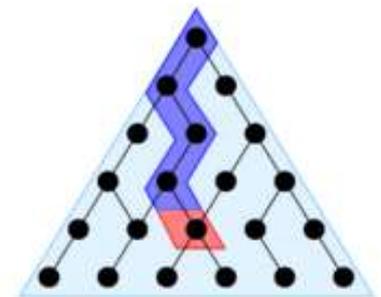
$EF P$



$EG P$



$EX P$



$E[P U q]$

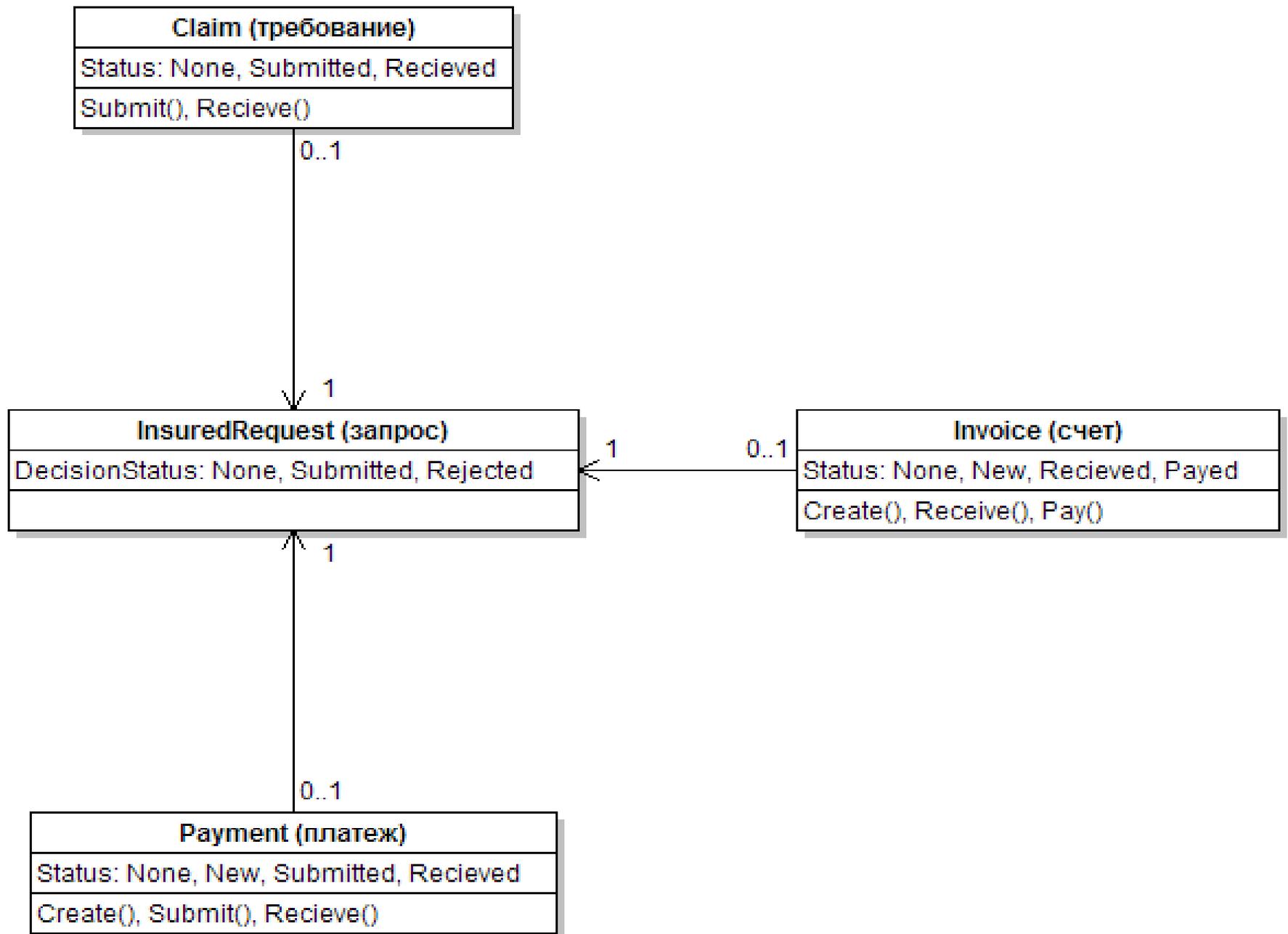
```
using CustIS.UniNet;
using CustIS.UniNet.Markup;

namespace Samples
{
    /// <summary> Сущность автомобиль с состояниями. </summary>
    [VerifyCTL("E G E X(!Stopped)", AutoState.Stopped)] // Можно тронуться
    [VerifyCTL("A G (! Winded & ! Driving => A X ! Driving)")] // Машина не может ехать, если не заведена
    [Entity]
    [TableStorage(TableName = "SomeTable")]
    public abstract class Auto : Entity
    {
        /// <summary> Идентификатор. </summary>
        [Attr(KeyPart = true)]
        [Column("id_column")]
        public abstract long Id { get; }

        /// <summary> Состояние машины. </summary>
        [Attr]
        [Column("state_column")]
        public abstract AutoState State
        {
            get;
            protected set;
        }

        /// <summary> Завести машину. </summary>
        [Method]
        [StateRestriction(AutoState.Stopped)]
        [StateTransition(AutoState.Stopped, AutoState.Stopped | AutoState.Winded)]
        public virtual void WindUp()
        {
        }

        /// <summary> Заглушить автомобиль. </summary>
        [Method]
        [StateRestriction(AutoState.Winded)]
    }
}
```



namespace Samples

{

```
// Счет не может быть оплачен, если принято решение отказать.  
[VerifyCTL("!E F (Rejected & Invoice.Payed)")]
```

```
///
```

```
[Entity]
```

```
[TableStorage("decision")]
```

```
public abstract class InsuredRequest : Entity
```

{

```
///
```

```
[Attr(KeyPart = true)]
```

```
[Column("Id")] struct System.Boolean  
public abstract Represents a Boolean value.
```

```
///
```

```
[Attr(InitialValue = DecisionStatus.None)]
```

```
[Column("status")]
```

```
public abstract DecisionStatus Status { get; protected set; }
```

|

```
///
```

```
[Attr]
```

```
[Column("garage")]
```

```
public abstract Garage Garage { get; set; }
```

```
///
```

```
[Attr]
```

```
[Column("insurance")]
```

```
public abstract Insurance Insurance { get; set; }
```

```
///
```

```
[BackReference("Request")]
```

```
public abstract Claim Claim { get; }
```

```
///
```

```
[BackReference("Request")]
```

```
public abstract Payment Payment { get; }
```

```
Payment.cs InsuredRequest.cs Auto.cs
Samples.Payment PayAdvanced()
[Method(AutoSave = true)]
[StateRestriction(PaymentStatus.New)]
[StateRestriction(DecisionStatus.Submitted, Chain = "Request")]
[StateTransition(PaymentStatus.New, PaymentStatus.Submitted)]
public virtual void Submit()
{
    Status = PaymentStatus.Submitted;
}

/// <summary> Получить платежу </summary>
[Method(AutoSave = true)]
[StateRestriction(PaymentStatus.Submitted)]
[StateTransition(PaymentStatus.Submitted, PaymentStatus.Recieved)]
public virtual void Recieve()
{
    Status = PaymentStatus.Recieved;
}

[Method]
[StateTransition(PaymentStatus.New, PaymentStatus.Submitted)]
public virtual void PayAdvanced()
{
    // Реализация супер-платежа
}
}
```

Error List

0 Errors 1 Warning 0 Messages

	Description	File	Line	Column	Project
1	Сущность "Сущность Samples.InsuredRequest (InsuredRequest)." не удовлетворяет ограничению "CTL Verification: formula = !E F (Rejected & Invoice.Payed)".	InsuredRequest	11	27	ADD2010

Список литературы

- Ю.Г. Карпов. Model Checking. Верификация параллельных и распределенных программных систем.
- <http://www-verimag.imag.fr/~sifakis/TuringAwardPaper-Apr14.pdf> Turing award. Model Checking: Algorithmic Verification and Debugging.
- <http://www.introducinglinq.com/files/folders/5/download.aspx> LINQ introduction.
- http://www.rsdn.ru/article/design/Code_Contracts.xml Проектирование по контракту.