Создание интерпретатора OCL для глубокого метамоделирования в REAL.NET

Шигаров Никита Алексеевич

научный руководитель: к.т.н., доц. Литвинов Ю. В.

рецензент: инженер-программист АО «ПФ «СКБ Контур» Перешеина А. О.

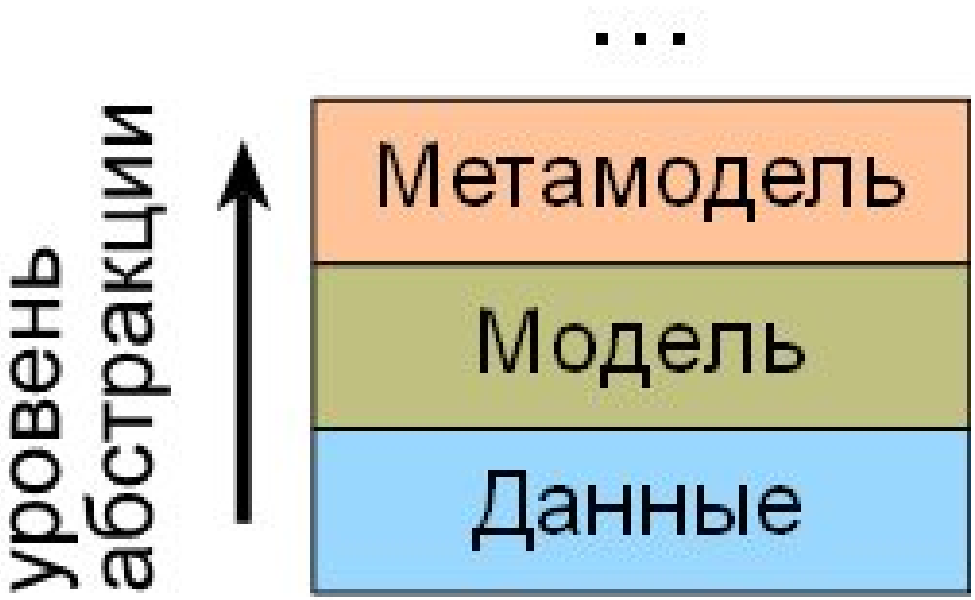
СПбГУ

12 июня 2019 г.

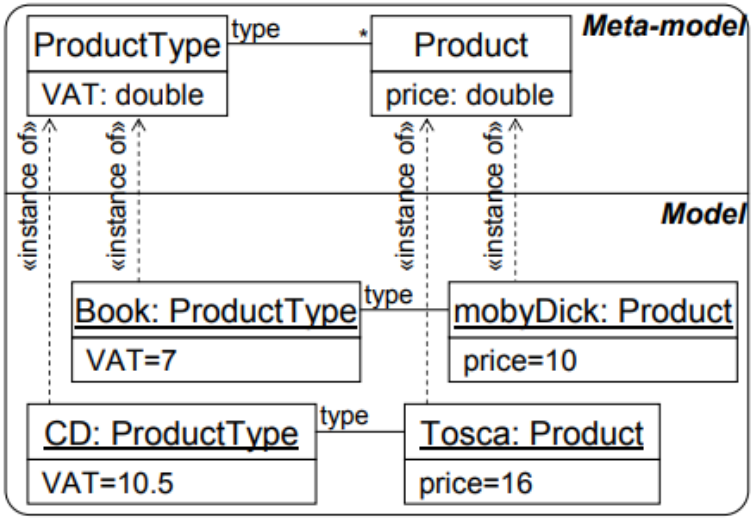
Метамоделирование

- данные — это множество простейших единиц информации, которые касаются не абстрактных, а конкретных сущностей (например, значение определённого свойства определённого элемента)
- модели — это описание структуры данных (например, информация о том какие есть свойства и связи между объектами)
- метамодели — это средства построения моделей (например, формальные языки или графические нотации для описания структуры классов, свойств и связей)

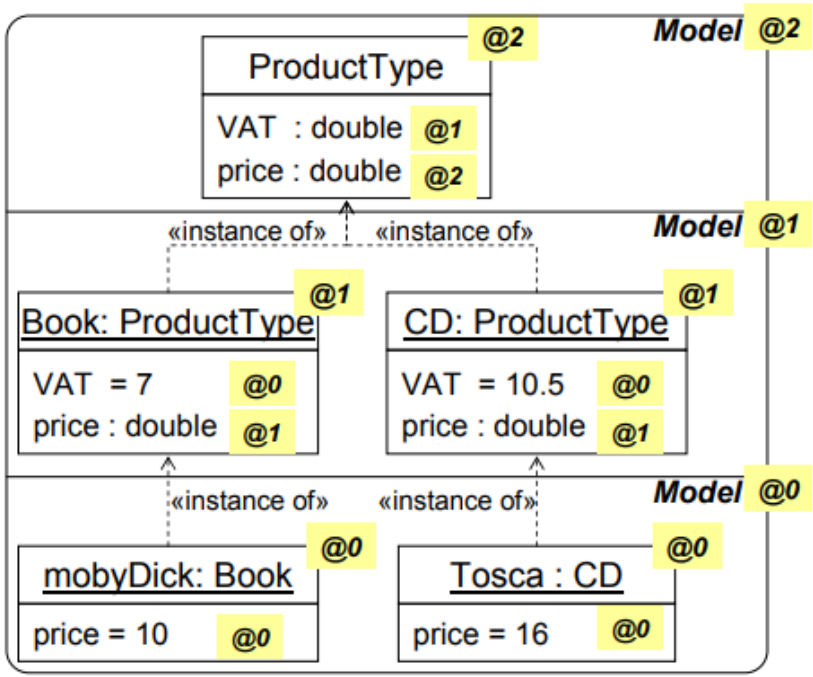
Уровни абстракций



Типы метамоделирования



Объект-тип



Глубокое метамоделирование

OCL для задания ограничений

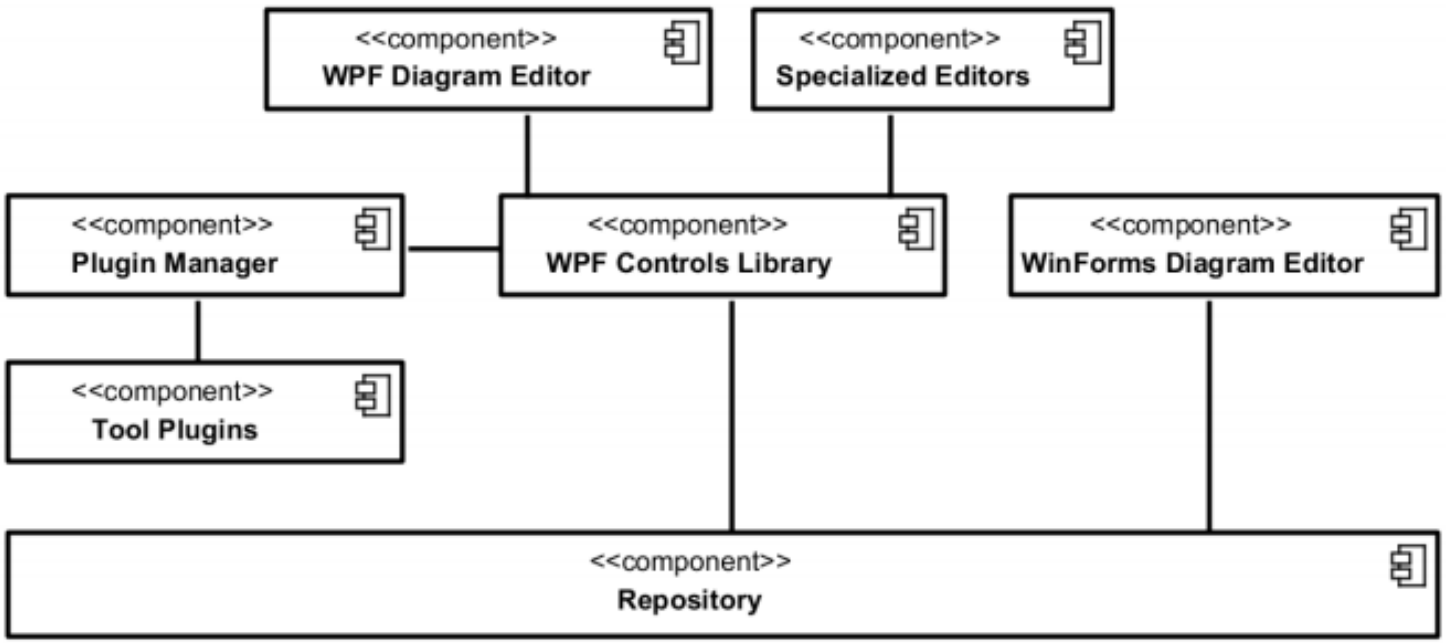
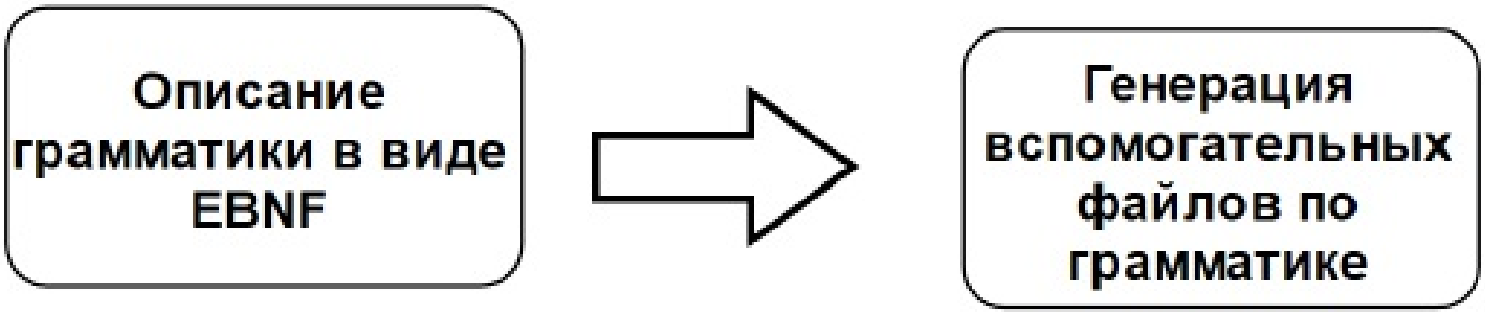
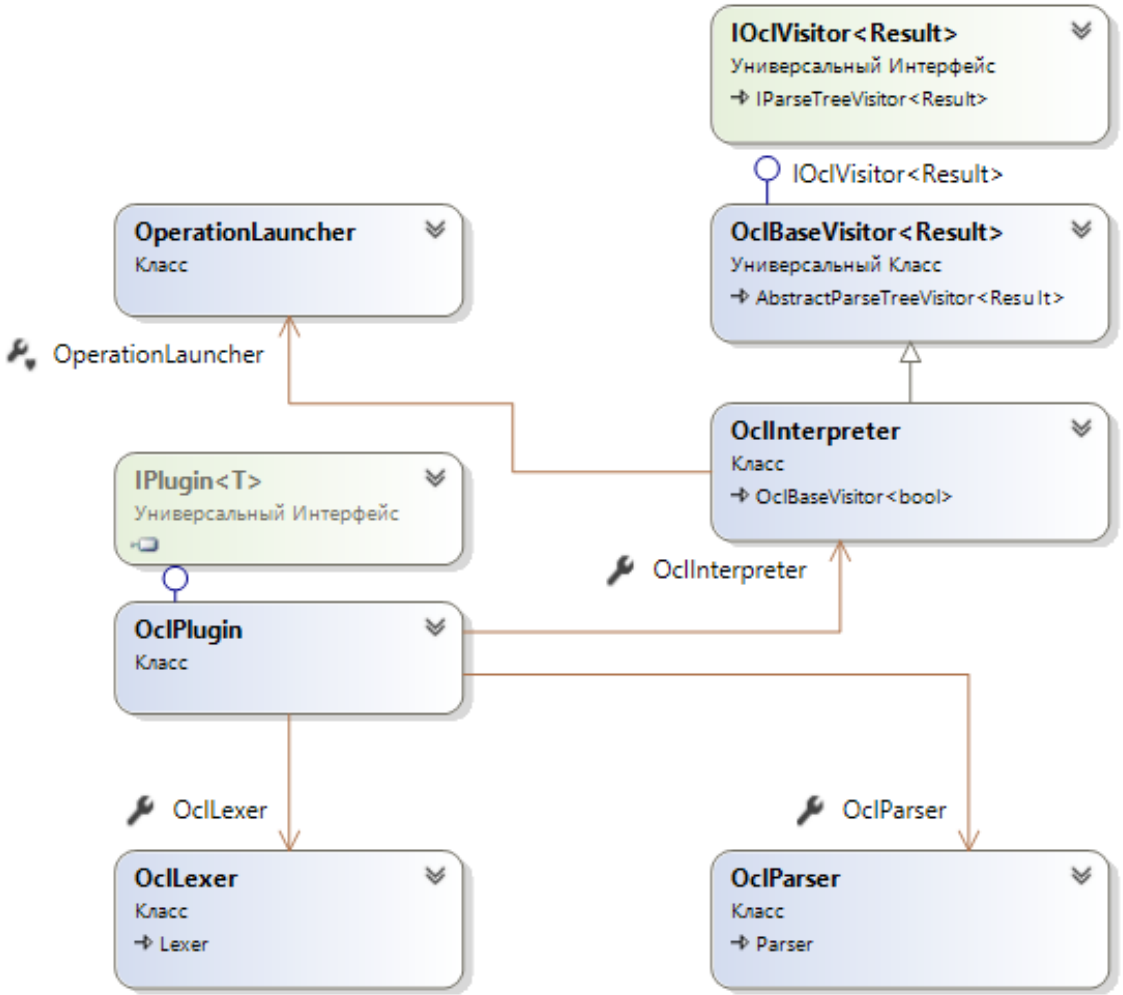
- Object Constraint Language (часть стандарта UML)
- декларативный язык описания ограничений
- ```
  1 context Person inv: self.age >= 0
  2 context Person inv: self.parents->forAll(p | p.age >
        self.age)
  ```
- нет современных open source реализаций OCL-интерпретаторов на .NET
  ```

- репозиторий для хранения моделей и метамоделей
- возможность написания плагинов
- редактор для визуализирования метаслоев



Задачи

- определить требования к интерпретатору
- выбрать подмножество языка OCL и модифицировать его в соответствии с требованиями
- реализовать интерпретатор OCL
- выполнить интеграцию с REAL.NET
- выполнить апробацию

ANTLR

- ANTLR 4 – генерация лексического и синтаксического анализатора по грамматике (на языки C#, Java, C++ и др.)
- модифицирована грамматика OCL для поддержки нескольких метаслоев



- использован паттерн Visitor

Требования

- возможность задавать ограничения на элементы более глубоких метамоделей, соответствующим данному
- возможность обращаться к полям из более глубоких метамоделей
- возможность комбинировать обращения к полям из разных метамоделей

Реализация требований

- помечать каждое ограничение номером самой глубокой метамодели относительно данного элемента через символ @
- помечать обращения к полям из более глубоких метамоделей номерами через символ @
- ```
  package P
  context Function
  inv@2:
     self.callParams@2.size() == self.params@1.size()
  endpackage
  ```
  ```

Диаграмма классов решения



Операции

- операции работы с числами (max, abs, round...)
- операции работы со строками (toReal, substring, concat...)
- операции работы с коллекциями (select, iterate, any...)
- возможность определять свои рекурсивные функции

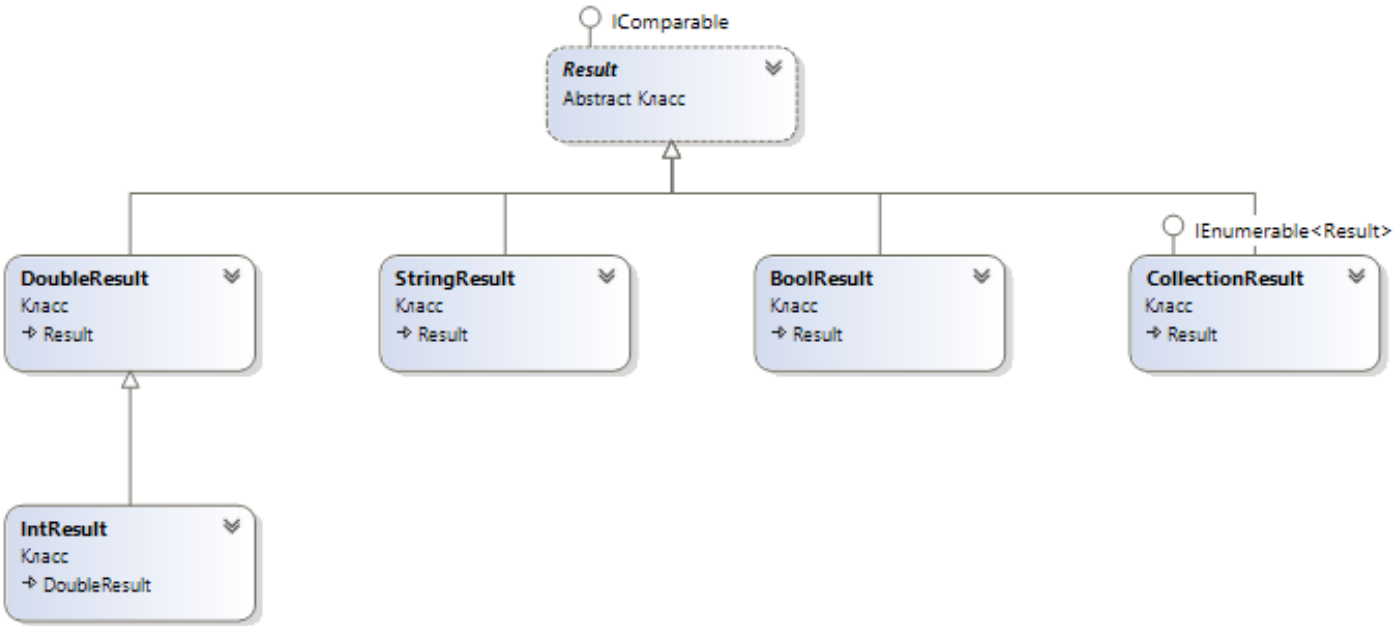
Рекурсивные функции

- определение и вызов рекурсивных функций

```
package P
context A
inv@0:
    let fact(n: var) =
        if n = 1 then n else n * fact(n - 1) endif
    let n = 5
    fact(n+1) = 720
endpackage
```

Система типов

- динамическая система типов
- **Sequence**{**Bag**{1}, **Set**{'a'}, **Set**{1.0}}->first()->
  asSequence()->first()+1



Окно редактора

Пример OCL-выражения

- Node -> Function -> Rotate -> FunctionCall

```
package Model
context Function
inv@2:
    self.callParams@2.size() == self.params@1.size()
endpackage
```

Пример OCL-выражения

- Node -> Function -> Rotate -> Type -> FunctionDefine

```
package Model
context Function
inv@2:
    self.allInstances@2->select(x.type == ”Call”)->
    forAll(true implies self.allInstances@2->
    select(y.type == ”Def”)->exists(z.name == x.name))
endpackage
```

Апробация

- тест на 5 пользователях
- ограничение из последнего примера
- System Usability Scale
- 90, 70, 57.5, 77.5 и 85 баллов

Результаты

- модифицирована грамматика OCL для поддержки нескольких метаслоев в соответствии с составленными требованиями
- реализован интерпретатор модифицированного языка OCL на языке C#
- выполнена интеграция с REAL.NET с помощью плагина и создания редактора
- для проверки корректности проведены опросы пользователей с оценкой по метрике System Usability Scale