Автоматическая генерация графических пользовательских интерфейсов доступа к интегрированным данным на основе диаграмм классов UML*

Д.Р. Ширяев

1. Введение

Система BizQuery [1], разрабатываемая группой MODIS ИСП РАН, позволяет интегрировать разнородные источники данных. В системе применяется подход виртуальный интеграции. При виртуальной интеграции данные из различных источников приводятся к одной схеме, и пользователь "видит" данные в терминах этой схемы. При этом внутри интеграционной системы данные не хранятся. Для выполнения запроса система выделяет из исходного запроса подзапросы к источникам, и после выполнения этих частичных запросов в интеграционной системе происходит соединение полученных результатов. Запросы к системе можно формулировать на одном из двух языках запросов - XQuery [2]и UQL [3]. Запросы на XQuery формулируются в терминах XML [4], а запросы на UQL – в терминах UML-модели [5].

Для удобной работы с интегрированными данными требуется обеспечить высокоуровневый доступ к ним в терминах предметной области. Для представления предметной области хорошо подходят и широко используются диаграммы классов UML – простое, мощное и выразительное средство для представления сущностей предметной области и связей между ними.

Как было установлено при выполнении проекта BizQuery, информации, содержащейся в диаграмме классов UML, оказывается достаточно для построения высокоуровневого интерфейса доступа к интегрированным данным. Очевидно, что строить интерфейс вручную каждый раз для новой предметной области проблематично. В проекте BizQuery был предложен подход, при котором графические пользовательские интерфейсы (Graphic User Interfaces - GUI) автоматически строятся по диаграмме классов UML-модели. Затем работа пользователя с данными производится в терминах предметной области,

* Работа частично поддерживалась грантом РФФИ 02-07-90300-в

понятных пользователю. При этом на каждом шаге навигации по модели интерфейс динамически перестраивается.

В системе BizQuery за обеспечение взаимодействий с пользователями отвечает компонент Frontend. Этот компонент Frontend позволяет строить три различных вида интерфейсов: Form Generator (Forms, FG), Catalog Generator (Catalogs, CG) и GraphicMap (GM). Forms и Graphic Map являются декларативными интерфейсами, а Catalogs – навигационным. Используя декларативный интерфейс, пользователь формирует запрос путем композиции ограничений на значения атрибутов классов модели, и получает результат в виде экземпляров классов модели, удовлетворяющих наложенным ограничениям. С другой стороны, используя навигационный интерфейс, пользователь может получать все экземпляры заданного класса модели с последующей возможностью перехода по связям от данного класса к другим классам. Связи между классами определяются в диаграмме классов UML-модели.

В графическом интерфейсе Forms пользователь выбирает класс, экземпляры которого он хочет получить (целевой класс), и формирует ограничения на значения атрибутов этого класса, а возможно, и на значения атрибутов классов, которые с ним связаны. В результате пользователь получает все экземпляры целевого класса, удовлетворяющие установленным ограничениям.

В графическом интерфейсе Catalogs пользователь также выбирает класс, экземпляры которого он хочет получить (целевой класс). При этом пользователь получает все экземпляры данного класса. После этого можно получить все экземпляры одного из классов, связанных с данным. Таким образом, используя связи между классами, определенными в диаграмме классов UML-модели, пользователь может осуществлять "навигацию" по интегрированным данным.

Важной особенностью интерфейсов Forms и Catalogs возможность их настройки в соответствии с потребностями пользователей. Пользователь может сам решить, как должен выглядеть его интерфейс, и написать соответствующий XSL-стиль.

При использовании графического интерфейса Graphic Map пользователь видит всю диаграмму классов предметной области, может выбрать целевой класс и определить условия, которым должны удовлетворять интересующие его объекты целевого класса и объекты классов, связанных с целевым.

2. Генерация интерфейсов

Первый этап. Подготовка UML-модели, соответствующей данной предметной области.

1. После обсуждения всех аспектов предметной области в среде Rational Rose [8] строится диаграмма классов UML-модели (Рис. 1), соответствующая данной предметной области.
2. Диаграмма классов UML-модели преобразуется в XMI-представление [9]. (Чтобы обеспечить возможность данного преобразования, было установлено

дополнение к Rational Rose «UnisysRoseXMLTools»). XMI-представления UML-моделей, соответствующие различным предметным областям, сохраняются в репозитории BizQuery.

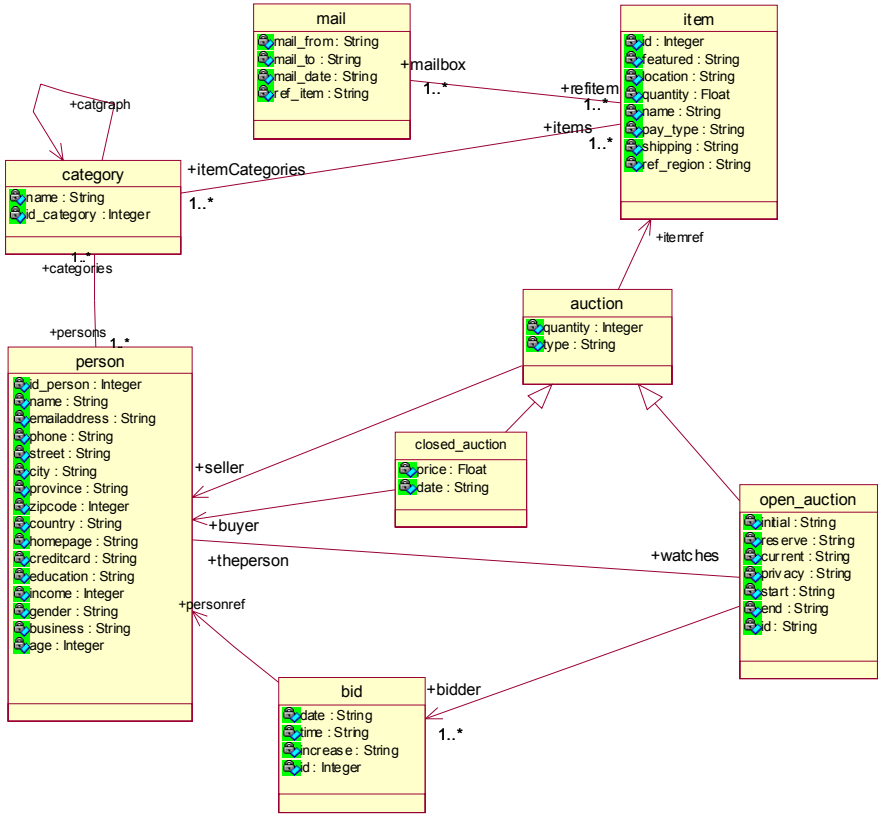

Рис. 1. *Диаграмма классов UML-модели.*

Второй этап. Динамическая генерация интерфейсов по заранее построенным XMI-представлениям диаграмм классов.

Процесс динамической генерации и использования интерфейсов показан на Рис. 2.

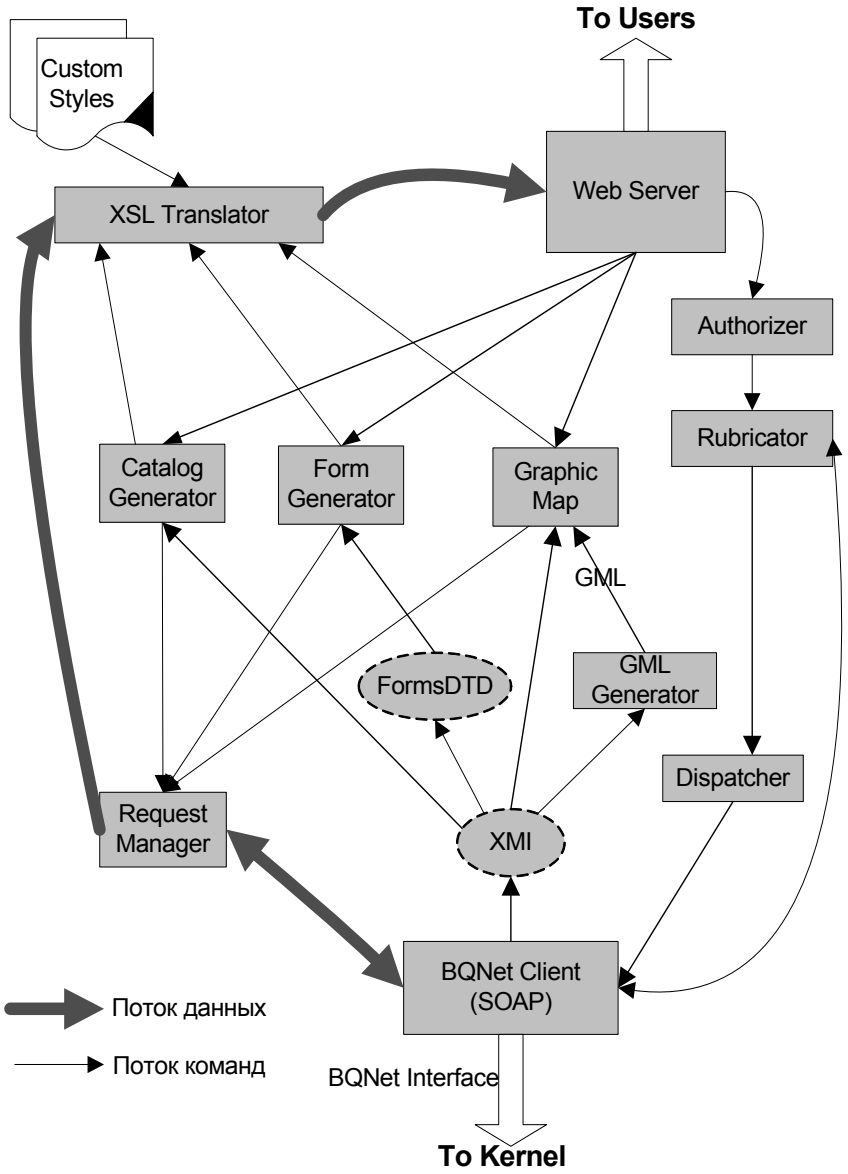

Рис. 2. *Динамическая генерация и работа графических интерфейсов компонента Frontend системы BizQuery.*

2.1. Основные модули компонента Frontend системы BizQuery

WebServer– поддерживает взаимодействия компонента Frontend системы BizQuery с пользователями. При реализации компонента Frontend использовался веб-сервер Jakarta Tomcat 3.3.1

Authorizer – модуль для авторизации пользователя. В форме авторизации вводится регистрационное имя пользователя и его пароль.

Rubricator –обращается к ядру и получает дерево UML-моделей, доступных пользователю в соответствии с авторизацией. Пользователь выбирает UML-модель, соответствующую интересующей его предметной области.

Dispatcher – модуль, в котором загружается форма со списком классов выбранной метамодели. Из этого списка пользователь выбирает целевой класс. Также пользователь выбирает один из трёх интерфейсов (Forms, Catalogs и GraphicMap), в котором он будет работать.

BQNet Client SOAP [11] – SOAP клиент для связи с ядром системы BizQuery.

Catalog Generator – основной компонент интерфейса Catalogs.

Form Generator – основной компонент интерфейса Forms.

Graphic Map – основной компонент интерфейса GM.

Request Manager – принимает UQL-запрос и через BQNet Client передаёт его ядру, получает результат запроса и передаёт его XSLT процессору для отображения.

XSLT Translator [6] – используется для различных XSL преобразований в компоненте Frontend системы BizQuery, в том числе:

- преобразует DOM (Document Object Model) [7] представление формы в HTML для отображения через Web Server;
- преобразует XML файл результатов запроса в HTML для отображения через Web Server.

3. Сценарии динамической генерации и работы графических интерфейсов компонента Frontend системы BizQuery

1. Пользователь через Web Server загружает Authorizer, вводит логин и пароль.
2. Загружается Rubricator, через BQNet Client обращается к ядру и получает дерево UML-моделей, доступных пользователю в соответствии с авторизацией.
3. Пользователь выбирает UML-модель, соответствующую интересующей его предметной области, эта модель загружается с сервера в виде заранее построенного XMI [9] представления диаграммы классов UML и переводится в специальное внутреннее представление на основе хэш-таблиц.

4. Загружается Dispatcher. Из списка классов выбранной UML-модели пользователь выбирает целевой класс. Кроме того, пользователь выбирает один из трёх интерфейсов (Forms, Catalogs, GraphicMap), который собирается использовать.

Если выбран интерфейс Forms:

5f. По внутреннему представлению в соответствии с Forms DTD (см. приложение 1) строится DOM-представление выбранной метамодели.

6f. По DOM с учетом одного из стилей Custom Styles с помощью XSLT строится форма, соответствующая текущему классу, т.е. классу, информация о котором отображается в форме в данный момент (рис. 3). Первым текущим классом является целевой класс.

7f. Заполняется форма, соответствующая текущему классу. При переходе по связи сохраняется информация о заполненных значениях атрибутов текущего класса, имя связи, по которой происходит переход на другой класс, меняется текущий класс и осуществляется перестройка формы. В строку UQL-запроса добавляется запись.

8f. При нажатии кнопки Submit строится запрос и передаётся модулю Request Manager.

9f. Request Manager через BQNet Client передаёт запрос ядру и получает от него результат запроса. В соответствии с одним из стилей Custom Styles с помощью XSLT результат преобразуется в HTML и передаётся пользователю.

Если выбран интерфейс Catalogs:

5c. Catalog Generator формирует запрос на выдачу всех экземпляров текущего класса и отдаёт его модулю Request Manager. Первым текущим классом является выбранный в Dispatcher целевой класс.

6c. Request Manager через BQNet Client передаёт запрос ядру, получает результат запроса, и отдаёт его модулю Catalog Generator.

7c. Catalog Generator, с помощью XSLT в соответствии со специальным стилем преобразует результат в XML документ, соответствующий Catalog DTD (см. приложение 2.) Это делается для того, чтобы пользователям было более удобно писать свои стили для преобразования результата в HTML.

8c. Далее с помощью XSLT в соответствии с одним из Custom Styles результат преобразуется в HTML и передаётся пользователю. Если количество выдаваемых экземпляров текущего класса велико, то они выдаются порциями. Для получения следующей порции достаточно нажать Next Portion.

7c. При каждом переходе по связи меняется текущий класс, строится запрос, производится обращение к модулю Request Manager, и всё повторяется, начиная с пункта 6c.

Если выбран интерфейс Graphic Map:

5g. Из Dispatcher загружается апплет интерфейса GM, строящийся на основе специального внутреннего представления выбранной UML-модели и заранее сгенерированного gml-файла (см. приложение 3).

6g. Пользователь, осуществляя навигацию по метамодели, генерирует запрос, и далее, также как и в интерфейсе Forms, получает результат запроса.

4. Полученные в результате генерации графические пользовательские интерфейсы

4.1. Интерфейс Forms

Интерфейс Forms является наиболее привычным для пользователей. Пользователь заполняет формы в терминах предметной области и получает интересующую его информацию опять же в терминах предметной области.

Внешний вид этого интерфейса зависит от используемого XSL стиля. Приведем пример формы, построенной с использованием стиля по умолчанию. В правой части формы (Рис. 3) находится информация о текущем классе (current business class) и список атрибутов текущего класса. Ограничения на текущий класс накладываются путём заполнения полей атрибутов.

Рис.3. *Интерфейс Forms.*

В левой части формы располагается список классов, с которыми есть связи у текущего класса. В этом списке два столбца: имя роли связи и имя класса, к которому идет данная связь в соответствии с метамоделью. По одной из связей можно перейти к соответствующему классу.

Внизу формы расположены 3 кнопки:

- `submit` – строит UQL запрос, и получает результат запроса,
- `back` – возвращает к предыдущему классу, стирает последний критерий из запроса,
- `reset` – стирает введённые значения атрибутов.

4.2. Интерфейс Catalogs

Внешний вид этого интерфейса также зависит от используемого XSL стиля. Приведем пример интерфейса, построенного с использованием стиля по умолчанию (Рис. 4).

В окне интерфейса Catalogs пользователь получает все экземпляры текущего класса. Если экземпляров много, то они выдаются порциями. Для получения следующей порции экземпляров необходимо нажать “Next Portion”.

Каждый экземпляр класса представлен в виде отдельной таблицы. В левой части таблицы изображены атрибуты классов, а в правой части ссылки на связанные классы. Ссылка содержит имя роли и имя целевого класса. Экземпляры можно фильтровать по имени класса или по значению атрибутов.

В процессе навигации по каталогам строится запрос, отправляется на сервер, и результаты запроса отображаются в том же окне.

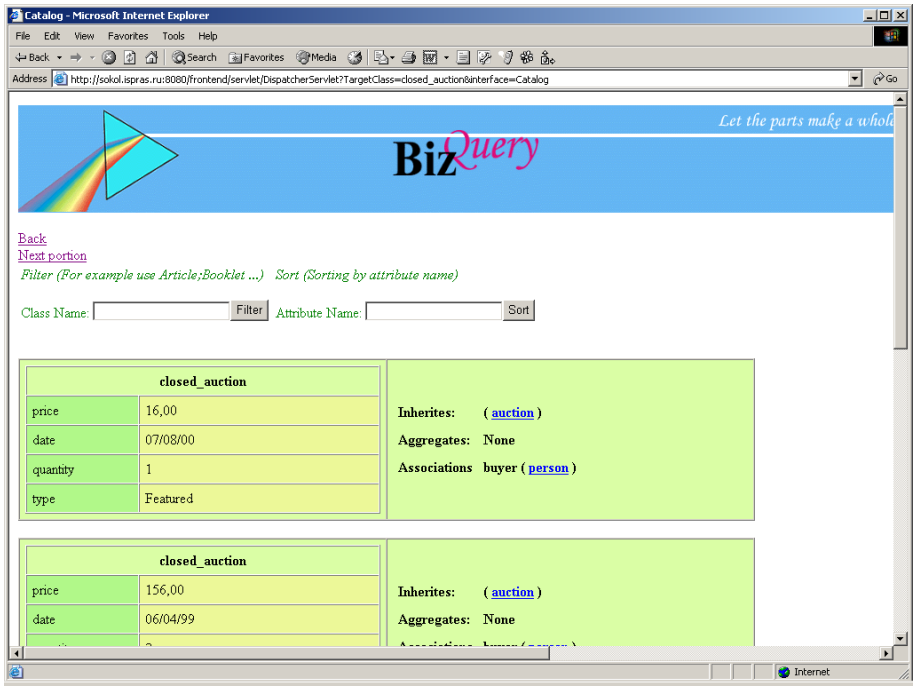

Рис. 4. Интерфейс Catalogs

4.3. Интерфейс Graphic Map

Graphic Map – оригинальный графический интерфейс, в котором предметная область представляется в виде, схожем с диаграммой классов UML модели, но более удобном, компактном и понятном для конечных пользователей.

4.3.1. Представление GM

Классы UML-модели распределены по боксам. Бокс – это графический объект карты, представляющий собой прямоугольник с закладками. Каждой закладке в боксе соответствует один класс UML-модели. При этом классы, объединенные в одно дерево наследования, размещаются в одном общем боксе. В каждый момент времени один из классов бокса является активным. При выборе определенной закладки бокса соответствующий класс становится активным.

Между боксами карты могут иметься связи разных типов (кроме наследования), соответствующие связям диаграммы классов UML между соответствующими классами, содержащимися в боксах. Таким образом, каждое дерево наследования помещается в отдельный бокс, куда попадает главный родитель этого дерева вместе со всеми наследниками. В частном случае, если класс не принадлежит никакому дереву наследования, то он попадает в отдельный бокс один. С каждым классом модели может быть связан рисунок.

4.3.2. Расположение GM

Окно браузера делится на 4 фрейма (Рис. 5). Центральный фрейм содержит графическое представление UML-модели. В правом фрейме находятся атрибуты текущего класса с текстовыми полями для ввода значений атрибутов. В левом нижнем фрейме собираются критерии для поиска. В правом нижнем фрейме находится дерево наследования текущего бокса. Вверху фрейма расположены инструменты для формулировки запроса.

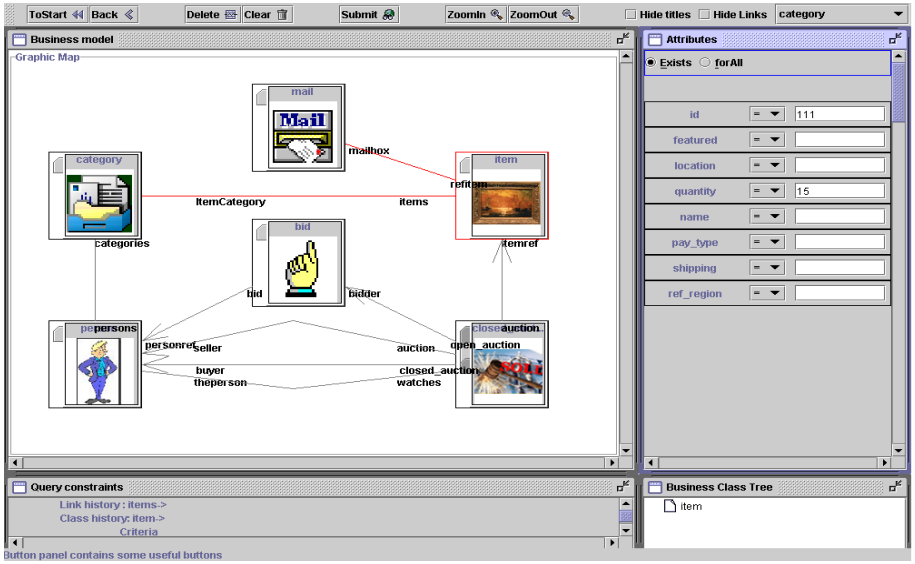

Рис. 5. *Интерфейс Graphic Map.*

5. Заключение

В данной статье рассмотрены основные аспекты автоматической генерации GUI на основе диаграммы классов UML модели. Все идеи, описанные в этой статье, были реализованы в рамках компонента Frontend системы BizQuery, над которой работает наша группа MODIS [1]. Тестирование и использование системы показало удобство и эффективность разработанного подхода.

Рассмотренной в данной статье темой также занимаются исследователи группы различных университетов мира, что подчёркивает ее актуальность.

Появляются также первые коммерческие продукты, например Polen E-companion Software[10].

Литература

1. Система BizQuery http://www.ispras.ru/groups/modis/bizquery.html
2. XQuery 1.0: An XML Query Language http://www.w3.org/TR/xquery/
3. UQL: A Query Language on Integrated Data in Terms of UML (in Russian, English) Maxim Grinev, Sergey Kuznetsov Programming and Computer Software, Vol. 28, No. 4, 2002, pp. 189-196.
4. Extensible Markup Language (XML) 1.0 (Second Edition) http://www.w3.org/TR/REC-xml
5. Unified Modeling Language (UML), version 1.4 http://www.omg.org/technology/documents/formal/uml.htm
6. XSL Transformations (XSLT) Version 1.0 http://www.w3.org/TR/xslt
7. Document Object Model (DOM) http://www.w3.org/DOM/
8. Rational Rose documentaion http://www.rational.com/
9. XML Metadata Interchange (XMI), version 1.2 http://www.omg.org/technology/documents/formal/xmi.htm
10. Pollen UML-GUI Designer http://www.e-companionsoftware.com
11. Simple Object Access Protocol (SOAP) 1.1 http://www.w3.org/TR/SOAP/

Приложение 1. Описание Forms DTD.

Forms.DTD

```
<!ELEMENT    model    (class+)>
<!ELEMENT    class    (attribute*, link*)>
<!ELEMENT    attribute EMPTY>
<!ELEMENT    link     (multiplicity*)>
<!ELEMENT    multiplicity     EMPTY>
<!ATTLIST    class
                      name     ID       #REQUIRED
                      selected (yes |no)#REQUIRED
<!ATTLIST    attribute
                      type     (integer | double | string | date)     #REQUIRED
                      name     CDATA#REQUIRED
                      label    CDATA#REQUIRED>
<!ATTLIST    link
```

```
                    roleName        CDATA#IMPLIED
                    name            CDATA#REQUIRED
                    type    (association | aggregation | composition |
inheritance)    #REQUIRED
                    targetClass     IDREF   #REQUIRED>
<!ATTLIST       multiplicity
                    lower   CDATA#IMPLIED
                    upper   CDATA#IMPLIED>
```

Элемент **model** является тэгом верхнего уровня XML документа, соответствующего Forms DTD. Он содержит внутри себя информацию о метамодели, определяемой данным документом.

Элементы **class**, **attribute** и **link** соответствуют понятиям класс, атрибут и связь в диаграмме классов UML модели.

Элемент **class** содержит следующие атрибуты:

- **name –** имя класса,
- **selected –** атрибут для определения текущего класса. Для текущего класса значение атрибута ”yes”, иначе - “no”.

Элемент **attribute** содержит следующие атрибуты:

- **type** – тип атрибута. Значение должно быть одним из нижеследующих: integer, double, string или date,
- **name**– имя атрибута. Имя атрибута должно иметь следующий формат: ИмяКласса.ИмяАтрибута,
- **label** – имя атрибута, которое отображается в форме.

Элемент link содержит следующие атрибуты:

- r**oleName** – имя роли связи,
- **name** – имя связи.

Тэг **link** содержит элемент **multiplicity.** Элемент **multiplicity** описывает, сколько сущностей класса, к которому идёт связь, могут соответствовать элементу текущего класса. Содержание тэгов **lower** и **upper** описывает соответственно максимальное и минимальное возможные значения возможного количества классов.

Приложение 2. Описание Catalog DTD.

Catalog DTD определяет структуру XML документа, представляющего результат запроса, вместе с информацией, извлекаемой из UML-модели.

Catalog DTD:

```
<!ELEMENT result (classinstance*)>
<!ELEMENT classinstance (attribute*, model)>
<!ELEMENT attribute     (#PCDATA)>
<!ELEMENT model         (reference*)>
<!ELEMENT reference (class*)>
```



```
<!ELEMENT classEMPTY>
<!ATTLIST classinstance
                    cname   CDATA#REQUIRED
                    id      ID      #REQUIRED>
<!ATTLIST attribute
                    aname CDATA   #REQUIRED>
<!ATTLIST reference
                    roleName CDATA #REQUIRED
                type(association|aggregation|
composition|inheritance) #REQUIRED>
<!ATTLIST class
                    cname CDATA #REQUIRED>
```

Верхним элементом документа является элемент result. В этом элементе содержатся экземпляры классов UML модели, полученные в результате запроса.

Описание экземпляра класса содержится в элементе **classinstanse**. Этот элемент содержит следующие атрибуты:

- **cname** - имя класса,
- ID- идентификатор экземпляра.

Также элемент **classinstanse** содержит элементы **attribute** и **reference**.

Элемент **attribute** имеет атрибут **‘aname’** – имя атрибута.

Элемент **reference** содержит следующие атрибуты:

- Type – тип связи. Значение должно быть одним из нижеследующих: association, aggregation, composition, inheritance,
- rolename – имя роли связи.

Элемент **reference** содержит элемент **class**. Элемент **class** содержит атрибут **cname –** имя класса, на который ссылается связь.

Приложение 3.

Описание GML

```
<element map (box*, link*)>
<attlist map sizeX CDATA #REQUIRED
            sizeY CDATA #REQUIRED >

<element box (tab+)>
<attlist box sizeX CDATA #REQUIRED
            sizeY CDATA #REQUIRED
            x CDATA #REQUIRED
            y CDATA #REQUIRED >
<element tab (reference*)>
<attlist tab className CDATA #REQUIRED
            xmi_id CDATA #REQUIRED
            label CDATA #REQUIRED
```

```
        active (yes|no) #REQUIRED>
<element reference empty>
<attlist reference link IDREF #REQUIRED
                xmi_id CDATA #REQUIRED>

<element link (turn*)>
<attlist link id ID #REQUIRED
        color CDATA #IMPLIED>
<element turn empty>
<attlist turn x CDATA #REQUIRED
        y CDATA #REQUIRED>
```

Элемент map содержит информацию о всей графической карте.

Элемент box описывает box.

Элемент link описывает линию на карте, представляющую определенную связь между классами.

Элемент reference описывает ссылку от данного класса на другой класс (важно направление) и также ссылается на тэг link по id, который описывает отображение данной связи на карте.

Элемент turn определяет координаты точки перегиба линии на карте.

Определения графической карты (GML) ссылаются на объекты XMI по идентификаторам:

- Элемент tab содержит атрибут-ссылку по id на соответствующий класс UML модели в файле XMI.
- Элемент reference содержит атрибут-ссылку по id на соответствующую ссылку UML модели в файле XMI.