Представление Трансформаций MDA в Виде Логических Объектов

Евгений Черкашин, **Алексей Шигаров**, **Вячеслав Парамонов***

Институт динамики систем и теории управления СО РАН,
Иркутский научный центр СО РАН
ул. Лермонтова, д. 134, Иркутск, Россия, 664033

*eugeneai@icc.ru

Международная конференция
Знания-Онтологии-Теории (ЗОНТ 2019)
7–11 Октября, 2019, Новосибирск, Россия

Цель исследования

Цель исследования – создать технологии MDA, которые основаны на современной системе визуальных языков моделирования (SysML, BPMN, CMMN, и др.), существующих средствах Семантического Веба: ***форматах***, ***словарях*** и ***технологиях***, а также объектно-ориентированном логическом программировании. Разрабатываются технологии и программное обеспечение:

1. Представление CIM в виде комплекса моделей SysML, BPMN, CMMN, а также результатов анализа исходного кода,
2. Представление CIM, PIM, PSM в формате RDF при помощи известных онтологий,
3. Реализация трансформации при помощи языка объектного логического программирования Logtalk,
4. Использование данных серверов LOD для обеспечения дополнительных семантических данных,
5. Порождение документов и интерфейсов пользователя, размеченных LOD (интеграция ИС в СВ).

Технологии-аналоги и стандарты

- Наиболее широко используется язык трансформации ATL и его предшественник QVT – стандарты OMG; разработаны для преобразования XMI в XMI;
- Использование ATL постепенно восходит на уровень CIM, например, диаграмма BPMN (CIM) преобразуется в набор диаграмм UML (PIM) классов и состояний, используемых для реализации web-приложений.
- Использование MDA становится шире, например, для анализа аспектов безопасности в распределенных приложениях, при этом результат получается на основе логического вывода над MDA–моделью;
- UML используется в качестве CIM/PIM при моделировании онтологий; есть специальный стандарт OMG;

Трансформация программируется в языке Logtalk, при этом CIM и PIM представляют собой граф RDF, PSM – это объекты LogTalk. В результате можно использовать не только модели, представленные в стандарте XMI, но и организовывать многоэтапные трансформации, формируемые из модулей.

Logtalk как язык задания трансформаций

Язык Logtalk выбран, так как он

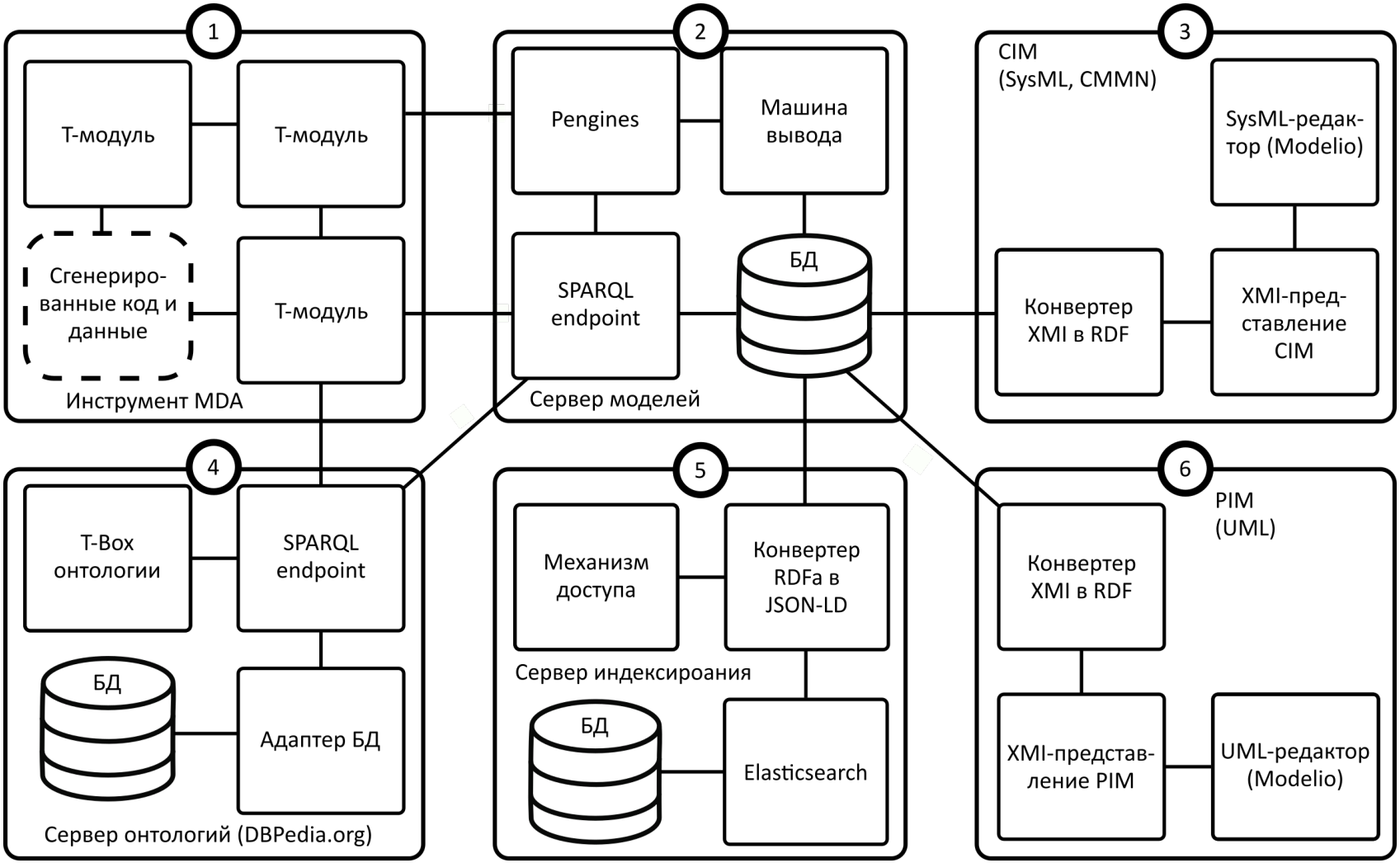
- наследует синтаксис и среду исполнения программ Prolog;
- реализован как макропакет, при этом функционирует всего на 1.5% медленнее;
- имеет гибкую семантику: можно задавать трансформации и ограничения в рамках одного синтаксиса;
- реализует объектно-ориентированную структуризацию знаний (правил), а также инкапсуляцию и модификацию;
- реализует композицию объектов;
- механизм установки ограничений на сообщения между объектами (события);
- существуют версии для разных реализаций Prolog.

Языки программирования общего назначения позволяют использовать библиотеки, которые не имеют прямого отношения к трансформациям MDA.

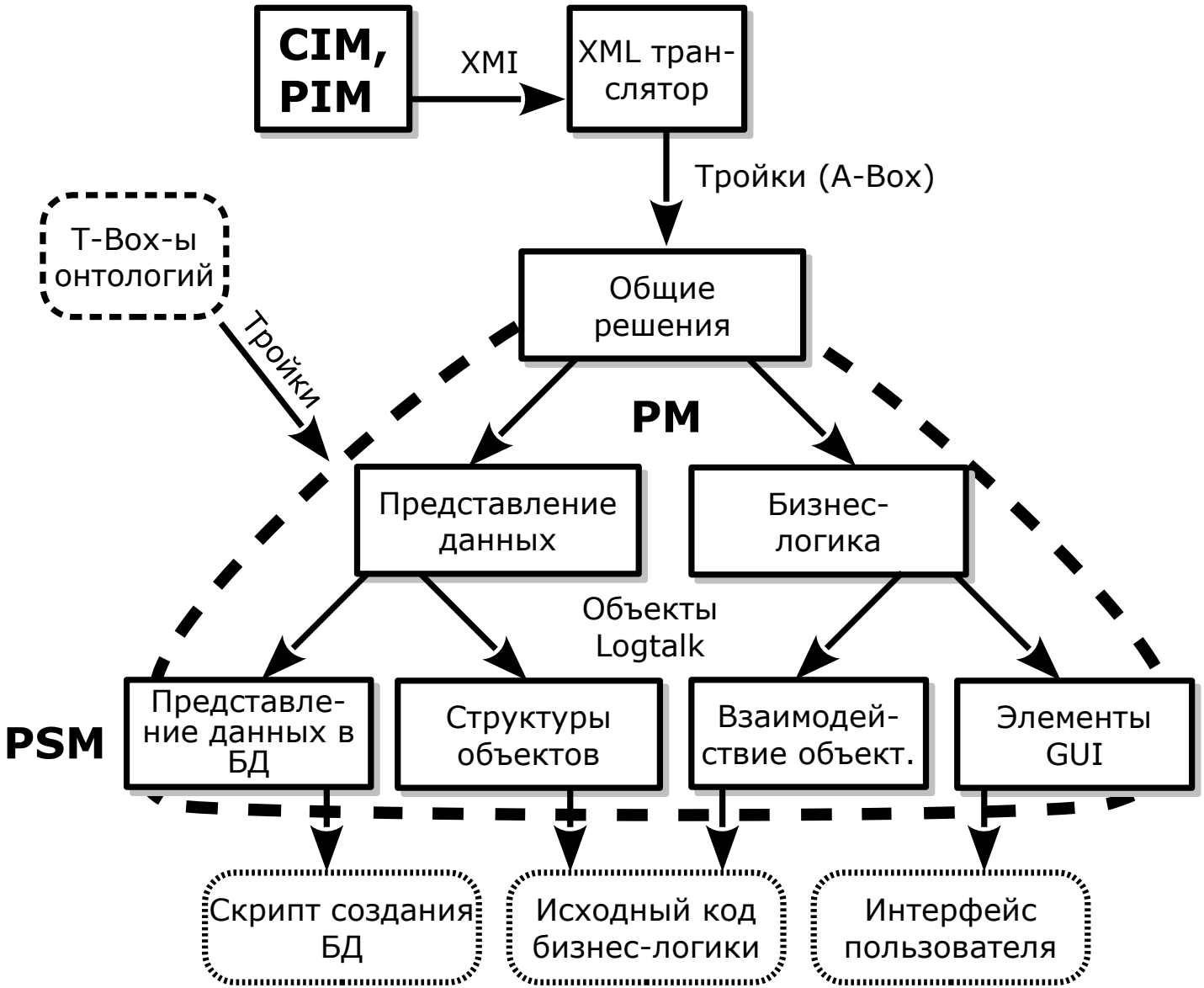
Использование средств Самантического Веба при представлении исходных моделей

- Использует опыт исследователей предметных областей, в том числе возведенный в стандарт;
- Наборы троек определяют граф (T-Box, A-Box);
- Стандартные онтологии заданы формально (`rdfs:domain`, `rdfs:range`);
- Поддерживаются практически любыми системами программирования (библиотеки, механизмы вывода/верификации, SPARQL);
- В RDF реализован способ глобальной идентификации сущностей в виде URI;
- SWI-Prolog поддерживает прямой доступ к графам RDF, интерпретацию семантики некоторых отношений (`rdfs:label`, `dc:title`); сервер онтологий ClioPatria;
- Существует простой способ реализации защиты данных (`rdfs:seeAlso`);
- При помощи средств СВ и принципов LOD можно организовывать интеграцию модулей в гетерогенной информационной среде.

Инфраструктура MDA



Архитектура модульной системы трансформации

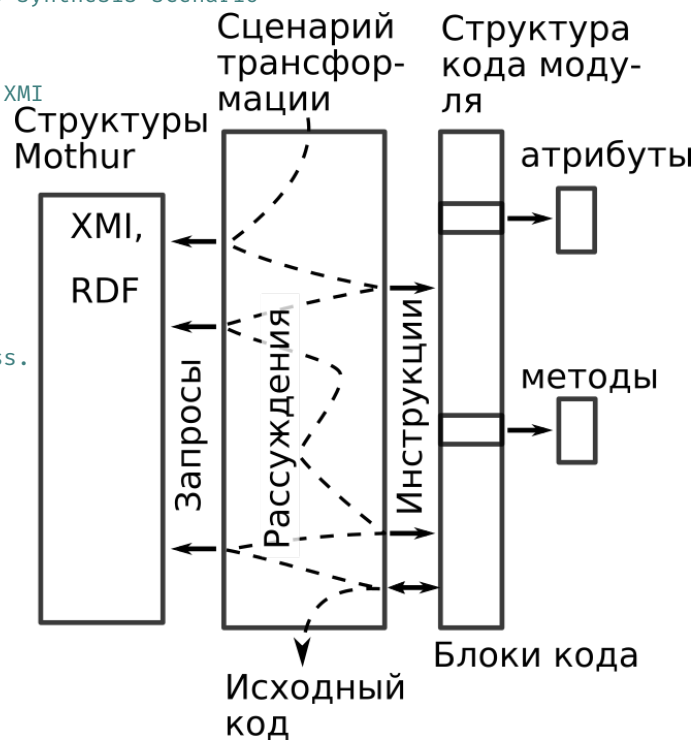


PSM: Сценарий синтеза класса

```
:- object(direct(_Package,_LocalProf,_CodeProf)).    % Transformation driver object
:- public([tr/4,tr/3]).                              % Public interface of a class synthesis scenario
% . . . . . .
tr(class, Class, ClassID):- ::package(Package),      % Synthesize a class
    query(Package)::class(Name, ClassID),            % Query package structure in XMI
    create_object(Class,      % . . . . .            % Create a «Class» object
    create_object(Attributes, % . . . . .            % Create «Attributes» object
    create_object(Methods,    % . . . . .            % ...«Methods».
    Class::name(Name),                               % Name the class.
    % Generate attributes of the class,
    % organizing them in a local database.
    % ...methods...
    Class::attributes(Attributes),                   % Set the attributes for class.
    Class::methods(Methods).                         % ...methods.

tr(attribute, Attribute, ClassID, AttributeID):-     % Attribute transformations
    ::package(Package),
    query(Package)::attribute(Name,ClassID,AttrID),
    create_object(Attribute,  % . . . . .
    Attribute::name(Name).                           % Name the attribute.

tr(method, Method, ClassID, MethodID):-              % Transformation of methods
    ::package(Package),
    query(Package)::method(Name,ClassID,MethodID),
    create_object(Method,     % . . . . .
    Method::name(Name).                              % Name of the method
:- end_object.
```

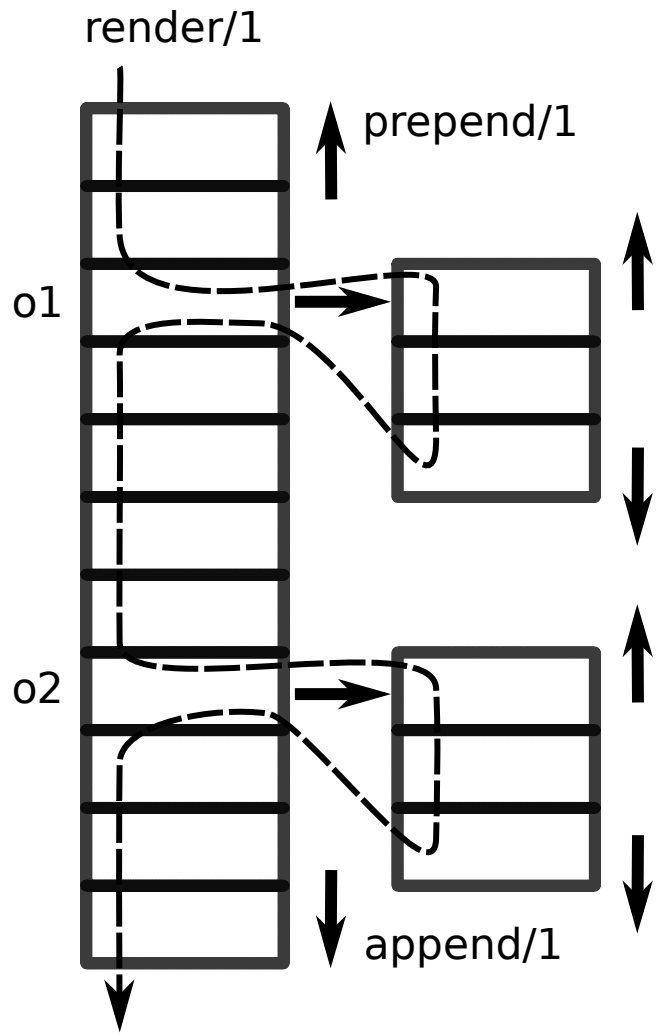


Реализация объекта Query

```
:- object(query(_XMI)).
:- protected(xmi/1).
:- public([class/2, attribute/3, method/3]).
xmi(XMI) :- parameter(1, XMI).
class(Name, ID):-                          % Recognition of Class in RDF
    ::xmi(XMI),
    XMI::rdf(ID,rdf:type,uml:'Class'),
    XMI::rdf(ID,rdfs:label, literal(Name)).
attribute(Name, ClassID, ID):-             % ...attribute...
    ::xmi(XMI),
    XMI::rdf(ClassID, xmi:ownedAttribute, ID),
    XMI::rdf(ID, rdfs:label, literal(Name)).
method(Name, ClassID, ID):-                % ...method...
    ::xmi(XMI),
    XMI::rdf(ClassID, xmi:ownedOperation, ID),
    XMI::rdf(ID, rdfs:label, literal(Name)).
% . . . . . .
:- end_object.
```

Класс Code Block (идея взята в llvmlite*)

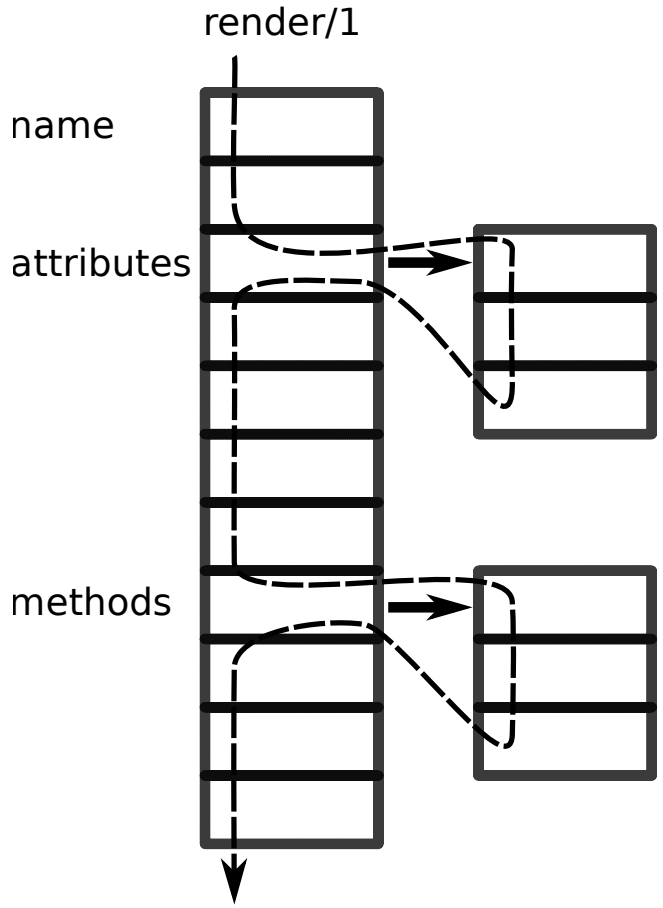
```
:- object(code_block, specializes(root)).
% Public interface of the object
:- public([append/1, prepend/1, clear/0,
   render/1, render_to/1, remove/1,
   item/1, items/1]).
% Code block items
:- dynamic([item_/1]).
:- private([item_/1]).
% Methods specialized during inheritance
:- protected([renderitem/2, render_to/2]).
% . . . . . .
% Delegate rendering to object itself
renderitem(Object, String):-
    current_object(Object), !,
    Object::render(String).
% Convert a literal to its string
% representation
renderitem(literal(Item), String):-!,
    atom_string(Item, String).
% Just print the item (debugging).
renderitem(Item, String):-
    root::iswritef(String, '%q', [Item]).
:- end_object.
```



*) https://github.com/numba/llvmlite

PSM класса Python, специализации Code Block

```
:- object(class, specializes(code_block),
   imports([named])). % Category of named entities
:- public([classlist/1, methods/1, attributes/1]).
% . . . . . .
renderitem(Item, Result):-      % proceed with default
    ^^renderitem(Item, Result). % rendering
render(Result):-          % Source generator
    ^^render(Name),       % implemented in a category
    ( ::item(classlist(List)) ->
     % . . . . . .
        [Name]) ),
    ( ::item(attributes(Attributes))->
     % . . . . . .
        [DefAttrList]),
      Attributes::items(InstanceAttrs),
      findall(S, ( % initialize attributes
         % . . . . . .
         ), AttrAssigns),
        root::unindent,
        AttrList=[ConstructorDef|AttrAssigns];
         % . . . . . .
        AttrList=[ConstructorDef, Pass] ),
    ( ::item(methods(Methods))-> % If any ...
      Methods::render(MethodList);
      MethodList=[] ),
    lists::append(AttrList,MethodList,StringList),
    root::unindent, Result=[Signature|StringList].
:- end_object.
```



Категории Logtalk

Категория поименованных сущностей

```
:- category(named).
:- public([name/1, render/1]).
:- protected([renderitem/2]).
name(Name):- ::prepend(name(Name)).
renderitem(name(Name), String):-!, atom_string(Name, String).
render(String):-  % What is code generation from items
    ::item(name(Name)), ::renderitem(name(Name), String).
:-end_category.
```

Категория поименованных сущностей некоторого типа

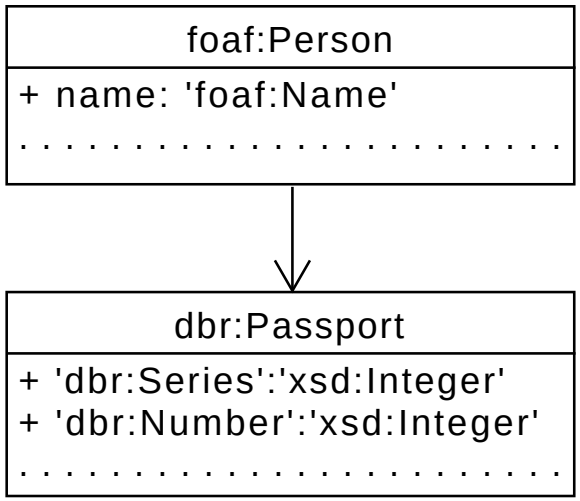
```
:- category(namedtyped, extends(named)).
:- public([type/1,render/2, separator_option/2,list_separator/1]).
:- protected([renderitem/2]).
type(Type):- ::append(type(Type)).
renderitem(Item, String):- ^^renderitem(Item, String),!.
renderitem(type(Type),String):-!, ::list_separator(Separator),
    writef::swritef(String, '%w%w', [Separator, Type]).
render(Middle, String):- ^^render(SName),
    (   ::item(type(Type)) ->
        ::renderitem(type(Type), SType),
        string_concat(SName, Middle, _1),
        string_concat(_1, SType, String) ;
        SName = String  ).
render(String):-  ::render("", String).
list_separator(Separator):-
    ::separator_option(Name, Default),!, % Global options
    root::option(Name, Separator, Default).
:- end_category.
```

Доступ к данным LOD

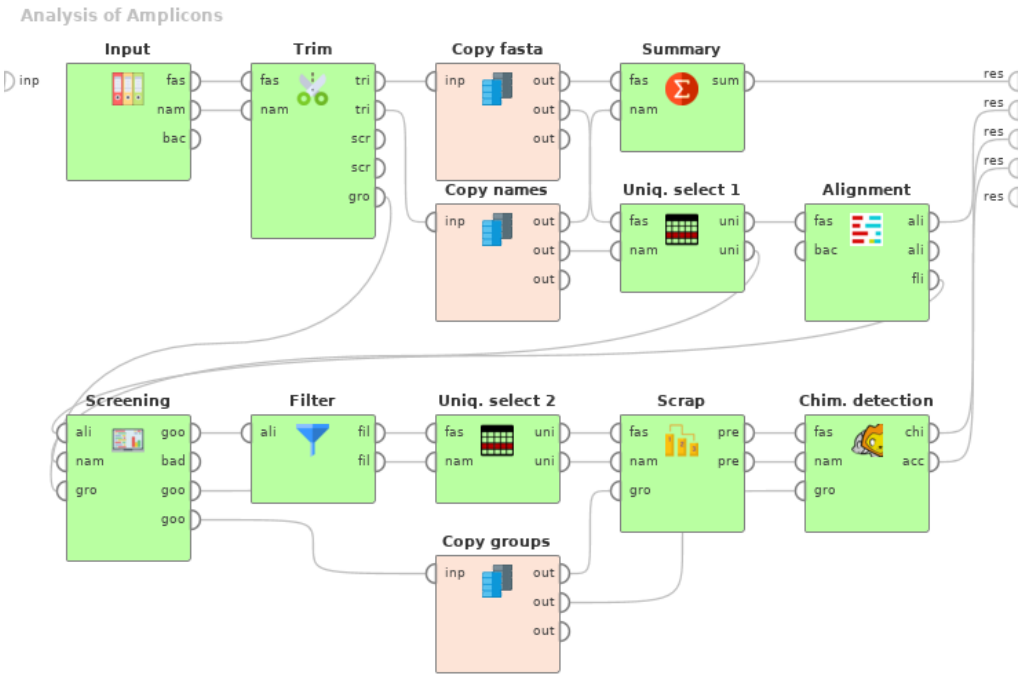
```
:- category(sparql).
:- public(query/2).
query(Pattern,Parameters,Row):-
    prepare(Pattern,Parameters,Query),
    server(Host,Port,Path),
    sparql_query(Query, Row,
        [host(Host),port(Port),path(Path)]).
:- protected(server/3).  % must be implemented
                         % by a subclass.
:- protected(prepare/3). % prepares a query
% . . . . . .    %              string.
:- end_category.

:- object(dbpedia, imports(sparql)).
:- protected(server/3).
server('dbpedia.org',80,'/sparql').
:- public(entity_name/2).
entity_name(Entity,Language,Name):-
    query('select ?name where { '
          ' %w rdfs:label ?name. '
          'FILTER langMatches( lang(?label),'
          ' "%w" )}', [Entity, Language],
          row(Name)).
:- end_object.

% ?- dbpedia::entity_name(dbr:'Passport', 'ru', Name).
```



Приложение: Представление анализа ампликонов в NGS в виде dataflow-диаграмм



Term	Расшифровка
NGS	New Generation Sequencing
Amplicon	Часть ДНК или РНК, скопированная много раз
Mothur	Прикладной пакет для исследований в NGS
Rapidminer	Инструмент визуального моделирования процедур анализа данных и их исполнения

Зеленые блоки – модули Mothur.
Остальные – модули Rapidminer.

Дискуссия

Получен интересный позитивный опыт:

- Logtalk и RDF очень гибки, универсальны и удобны при реализации трансформаций в MDA;
- Наиболее полезный приемы программирования – это инкапсуляция запросов к RDF в предикаты–методы и инкапсуляция наборов правил в объекты Logtalk;
- Не все инструменты и приемы программирования Logtalk освоены: необходимо их изучить и выработать методики их использования, например, для инструментов перехвата сообщений (watchers).

Некоторые технические проблемы немного портят общий оптимизм:

- Очень простые задачи при использовании MDA требуют несоизмеримых усилий, например, обработка текстов: преобразование идентификатора в CamelCase;
- Много времени занимает поиск подходящей онтологии в интернете, но это все же продуктивнее, чем собственная разработка;
- Prolog не самый популярный язык программирования в MDA, то же и для LogTalk.

Заключение

К настоящему времени получены следующие результаты:

- Методика представления моделей разработана и в некоторой степени протестирована.
- Создана методика объектно-ориентированного программирования процедур трансформации для языка Logtalk.
- Реализованы прототипы для некоторых трансформаций.
- Средства трансформаций апробированы в прикладной области, сложных проблем пока не выявлено.

Дальнейшее развитие видим по следующим направлениям:

- Разработка методики автоматизации разметки отчетных документов LOD.
- Развитие методики реализаций модулей трансформаций с минимальным использованием динамических объектов, используя свойства Logtalk как макропакета.
- Сформировать современный инструментарий разработки ИС из существующего прототипа.

Исходный код хранится на сервере
https://github.com/isu-enterprise/icc.xmitransform,
https://github.com/eugeneai/icc.mothurpim.

Спасибо за проявленный интерес к нашему проекту!

Исходный код модуля NGS в Rapidminer

```
vector<string> AlignCommand::setParameters(){ // PART OF MODULE SOURCE
try {
  CommandParameter ptemplate("reference", "InputTypes", "", "", "none", "none", "none","",false,true,true); parameters.push_back(ptemplat
  CommandParameter pcandidate("fasta", "InputTypes", "", "", "none", "none", "none","fasta-alignreport-accnos",false,true,true); paramete
  CommandParameter psearch("search", "Multiple", "kmer-blast-suffix", "kmer", "", "", "","",false,false,true); parameters.push_back(psear
  CommandParameter pksize("ksize", "Number", "", "8", "", "", "","",false,false); parameters.push_back(pksize);
  CommandParameter pmatch("match", "Number", "", "1.0", "", "", "","",false,false); parameters.push_back(pmatch);
// . . . . . .
package com.rapidminer.ngs.operator; // GENERATED JAVA MODULE
// imports

class MothurChimeraCcodeOperator extends MothurGeneratedOperator {
  private InputPort fastaInPort = getInputPorts().createPort("fasta");
  private InputPort referenceInPort = getInputPorts().createPort("reference");
  private OutputPort chimeraOutPort = getOutputPorts().createPort("chimera");
  private OutputPort mapinfoOutPort = getOutputPorts().createPort("mapinfo");
  private OutputPort accnosOutPort = getOutputPorts().createPort("accnos");

  public MothurChimeraCcodeOperator (OperatorDescription description) {
    super(description);
  }
  @Override
  public void doWork() throws OperatorException {
    super();
    // . . . . . .
  }
  @Override
  public List<ParameterType> getParameterTypes() {
    super();
        // . . . . . .
  }
  @Override
  public String getOutputPattern(String type) {
    if (type=="chimera") return "[filename],[tag],ccode.chimeras-[filename],ccode.chimeras";
    if (type=="mapinfo") return "[filename],mapinfo";
    if (type=="accnos") return "[filename],[tag],ccode.accnos-[filename],ccode.accnos";
    return super.getOutputPattern(type);
  }
}
```

Представление исходного кода Mothur в RDF (TTL) и его объект-фасад query

```
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
ngsp:spec a ngsp:Specification ;
    ngsp:module mothur:NoCommand,
        mothur:align-check,
        mothur:align-seqs,
# . . . . .
mothur:align-check a ngsp:Module ;
    ngsp:outputPattern [ a cnt:Chars ;
            ngsp:parameterName "type" ;
            ngsp:pattern [ ngsp:patternString
                    "[filename],align.check" ;
                    dc:identifier "aligncheck" ] ;
            cnt:chars # . . . .
# . . . . .
mothur:align-check-idir-parameter a ngsp:Parameter ;
    ngsp:important false ;
    ngsp:multipleSelectionAllowed false ;
    ngsp:optionsDefault "" ;
    ngsp:required false ;
    ngsp:type mothur:String ;
    dc:title "inputdir" .

mothur:align-check-map-parameter a ngsp:Parameter ;
    ngsp:important true ;
    ngsp:multipleSelectionAllowed false ;
    ngsp:optionsDefault "" ;
    ngsp:required true ;
    ngsp:type mothur:InputTypes ;
    dc:title "map" .

mothur:align-check-name-parameter a ngsp:Parameter ;
    ngsp:chooseOnlyOneGroup "namecount" ;
    ngsp:important false ;
    ngsp:multipleSelectionAllowed false ;
```

```
:- object(queryparam(_RDF,_Parameter),
          extends(ngsquerybase)).

:- public(type/1).
type(Type) :-
    ::attr(type, Type).
:- public(name/1).
name(Name) :- ::attr(dc:title, literal(Name)).
:- public(options/1).
options(Value):- ::attr(options, Value).
:- public(options_default/1).
options_default(Value):-
    ::attr(optionsDefault, Value).
% . . . . . .
:- public(multiple_selection_allowed/0).
multiple_selection_allowed:-
    ::bool_attr(multipleSelectionAllowed).
:- public(required/0).
required:-
    ::bool_attr(required).
:- public(important/0).
important:-
    ::bool_attr(important).
:- protected(attr/2).
attr(NS:Name, Value):-
    ::ngs(RDF),
    ::second(Parameter),
    rdf_db::rdf_global_object(Value, V),
    RDF::rdf(Parameter, NS:Name, V).
attr(Name, Value):-
    \+ Name=_:_,!,
    ::ngs(RDF),
    ::second(Parameter),
    rdf_db::rdf_global_id(Value, V),
    RDF::rdf(Parameter, ngsp:Name, V).
```