Реализация трансформации моделей на основе объектно-ориентированного логического программирования

Евгений Черкашин
eugeneai@icc.ru

г. Иркутск, Институт динамики систем и теории управления им. В. М. Матросова СО РАН

III Всероссийская научно-практическая конференция «Современное программирование»

Цели исследования

Основная цель исследования – разработка технологии MDA (Model Driven Architecture), где используются UML, SysML, BPMN, CMMN и Семантический Web (SW) в качестве абстрактных нотаций. Решаются следующие основные задачи:

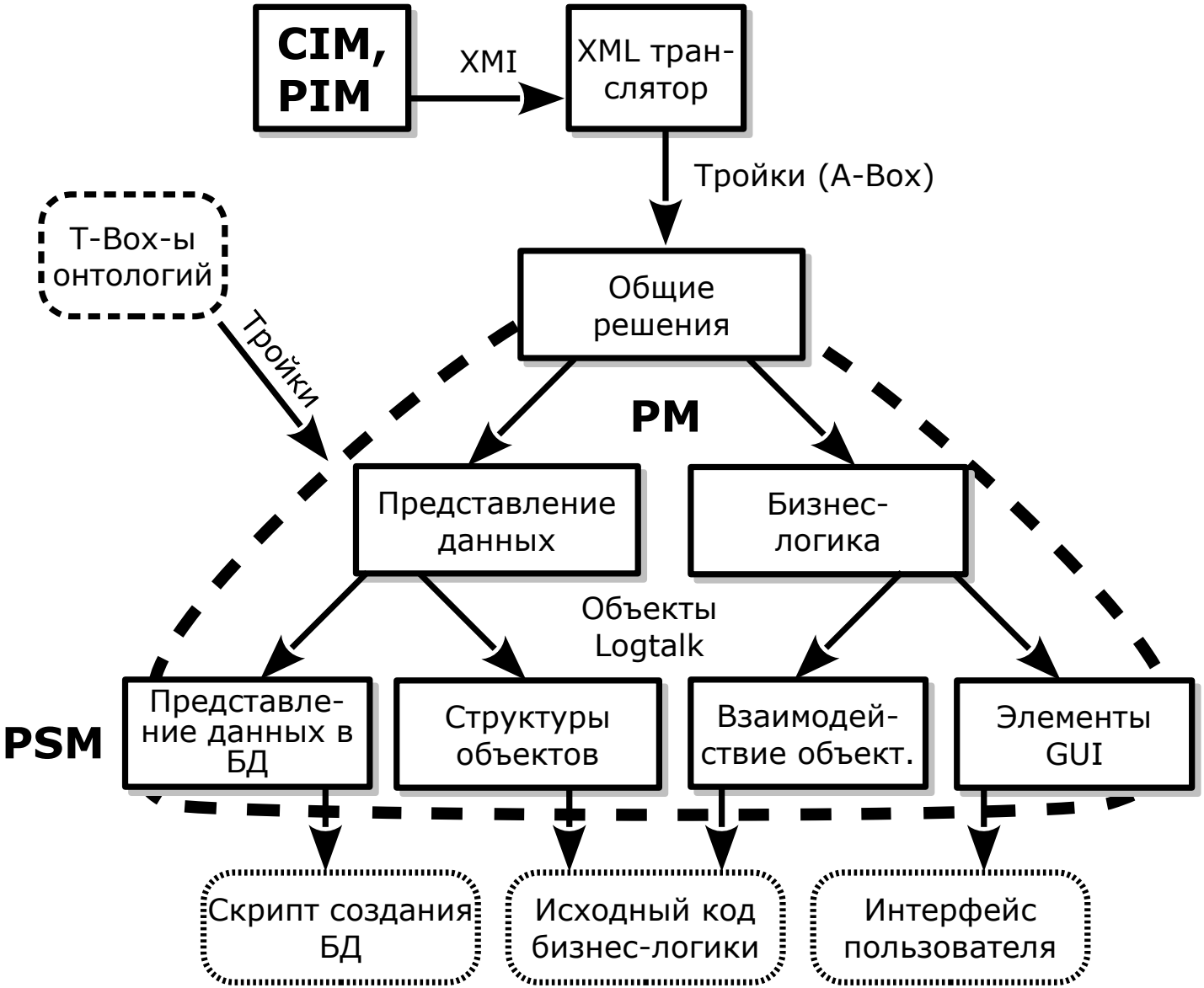
1. представление CIM при помощи SysML, BPMN, CMMN, SW,
2. представление CIM, PIM, PSM при помощи RDF и стандартизованных онтологий,
3. реализация трансформаций на логическом OO-языке Logtalk,
4. использование данных LOD-хранилищ для получения дополнительной семантической информации,
5. порождение документов и интерфейсов пользователя с применением LOD-разметки.

Современное состояние

- Стандартами OMG для представления трансформаций UML являются среды ATL и QVT; преобразование XMI в XMI;
- Использование ATL наблюдается и на уровне CIM, например, диаграммы BPMN преобразуют в UML (PIM); используются для создания web-приложений, где CIM представляют при помощи UML-диаграмм «State Machine» и «Use case».
- Класс задач, решаемых MDA, расширяется, например, проводится анализ аспектов защищенности распределенных приложений (логический вывод на PIM, представляющей распределенную систему);
- UML используется как язык онтологического моделирования (CIM/PIM); существует спецификация OMG;
- XML используется для структурного и семантического описания сервисов, например, WSDL, WS-BPEL;
- MDA – это противоположный подход концептуальному программированию Э.Тыугу.

Использование языка ОО логического программирования Logtalk и SW позволяет выйти за рамки XMI: использовать все доступные источники данных и библиотеки.

Архитектура модулей трансформации

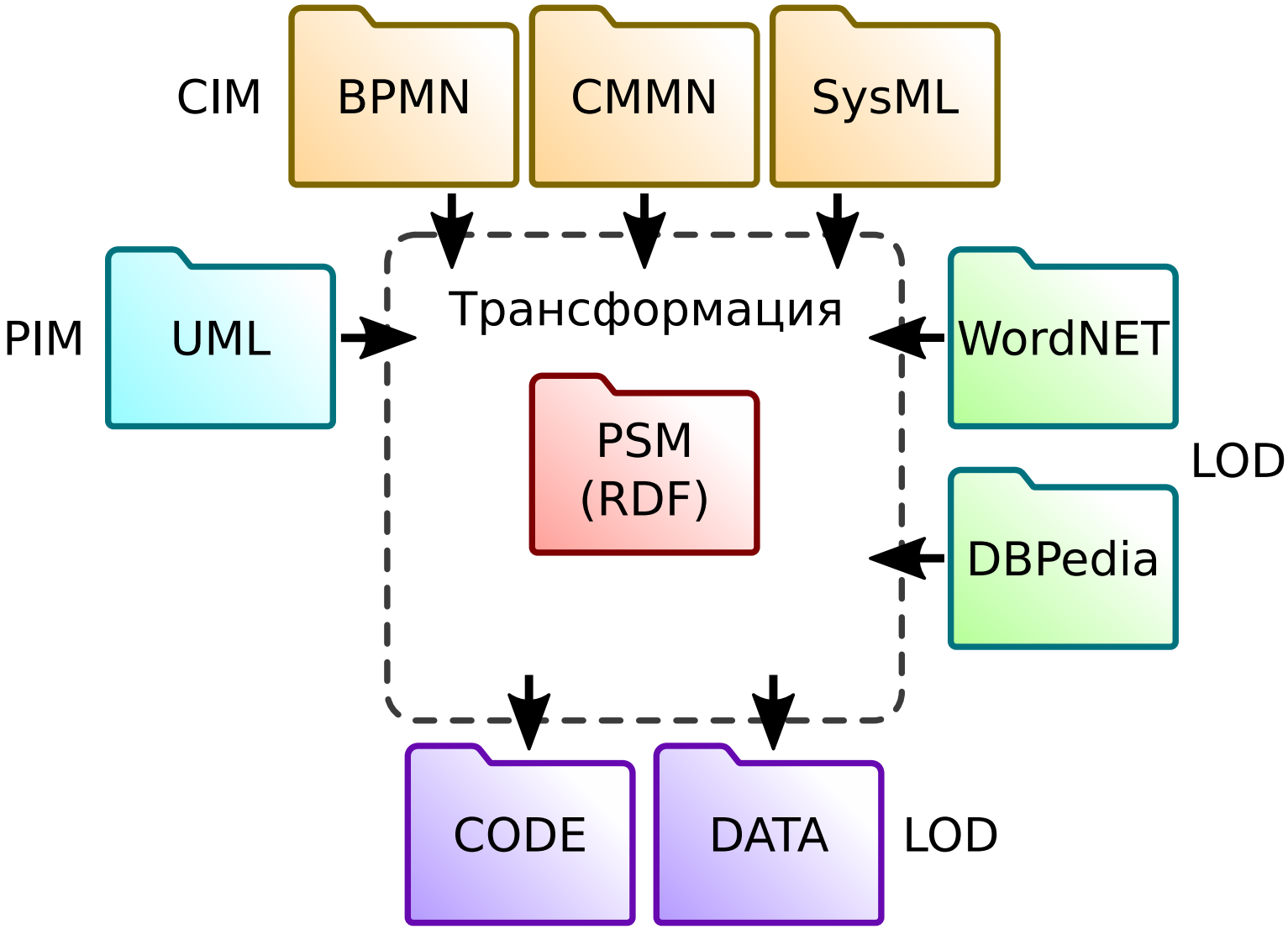


Язык Logtalk

Язык Logtalk выбран в качестве языка представления трансформаций по следующим причинам:

- наследует все свойства сред программирования **ISO-Prolog**;
- реализован в виде **макропакета**; накладные расходы в случае использования только статических объектов – 1.5%;
- гибкий подход к представлению трансформаций: синтаксически одинаково задаются как трансформации, так и ограничения;
- параметрические объекты (`circle(0,0,10)::square(S)`).
- реализация ОО-представления базы знаний (правил): структуризация, инкапсуляция, замена части и т.п.;
- представление объектов-трансформаций как композиций при помощи категорий;
- механизмы фильтрации при передаче сообщений;
- представлены реализации для разных ISO-Prolog-систем.

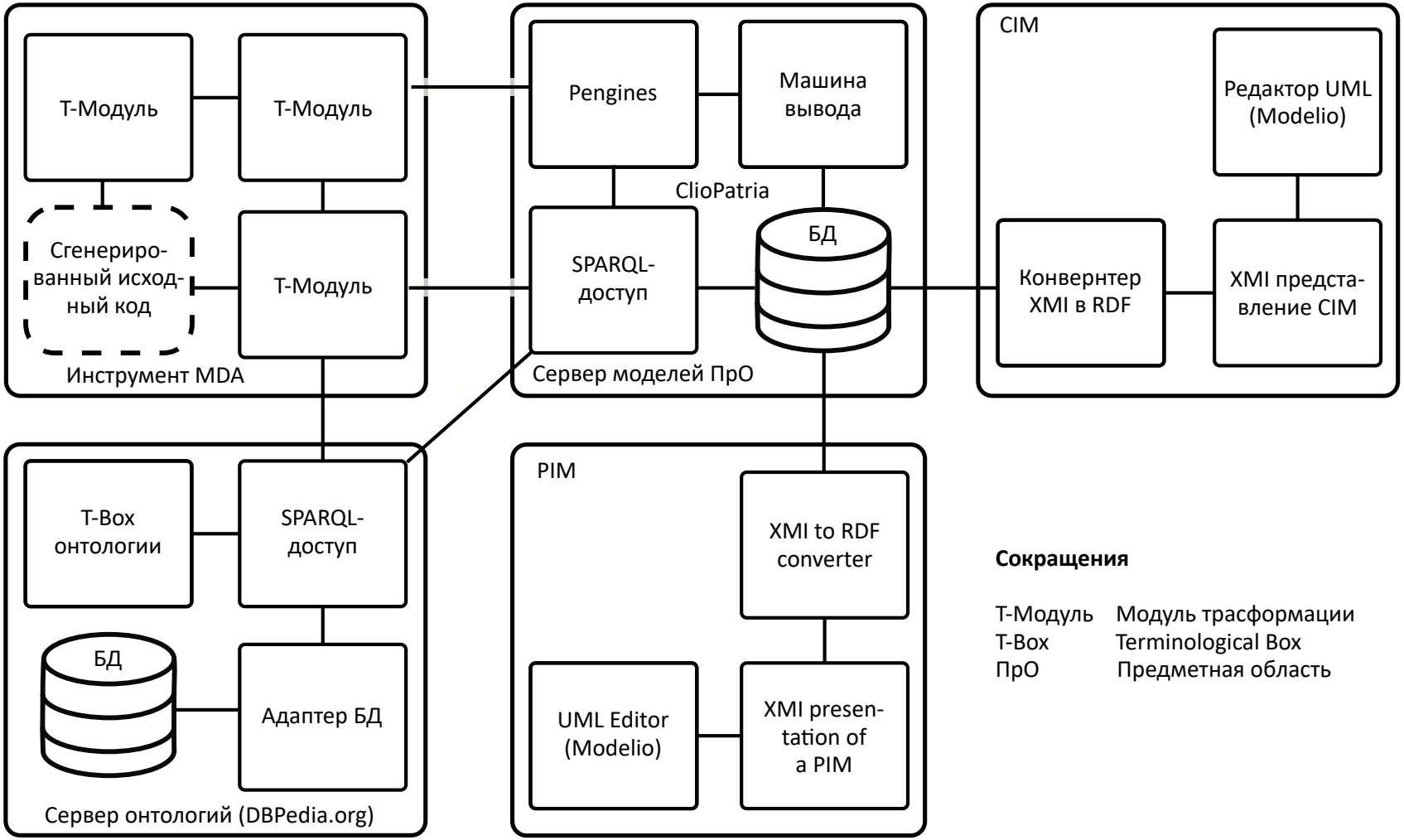
Model Driven Architecture и Linked Open Data



Применение SW для представления моделей

- Использование накопленного опыта и стандартов формализации предметных областей;
- Представление T-Box и A-Box при помощи множества троек (графа) <субъект, отношение, объект>;
- Элементы стандартных онтологий формально описаны (rdfs:domain, rdfs:range);
- Поддерживаются большинством программных систем (библиотеки, системы логического вывода, SPARQL);
- Предлагает способ глобальной идентификации объектов в различных независимых системах;
- SWI-Prolog включает библиотеку, позволяющую осуществлять запросы к графу, интерпретацию семантики некоторых отношений (rdfs:label, dc:title), инкапсуляцию BNode в многоаргументные предикаты; сервер онтологий ClioPatria;
- Предоставляется простой способ разделения уровней доступа к информации: (rdfs:seeAlso);
- Разметка RDF/LOD позволяет интегрировать гетерогенные системы.

Инфраструктура сервисов MDA

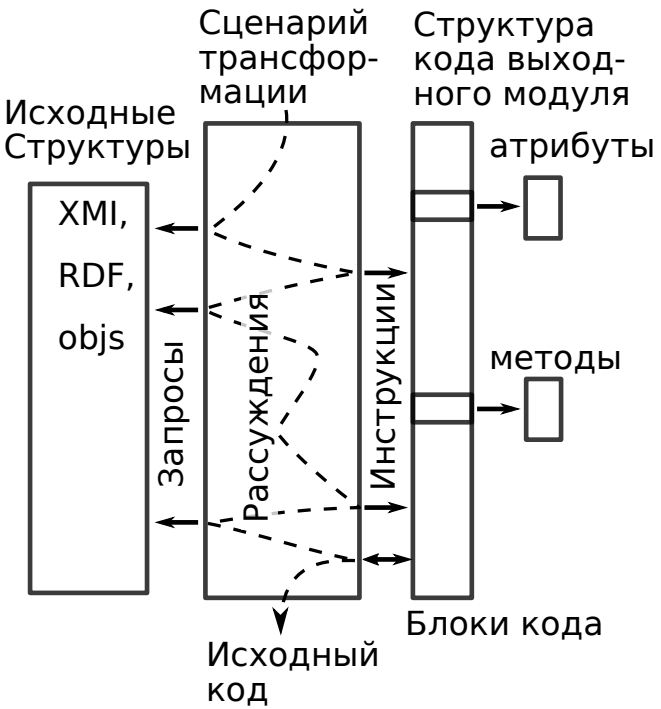


Сокращения

Т-Модуль	Модуль трасформации
Т-Box	Terminological Box
ПрО	Предметная область

Сценарий синтеза класса по PSM

```
:- object(script(_Package_,_LocalProf_,_CodeProf_)). % Трансформационный профиль
  :- public([tr/4,tr/3]).                           % Публичный интерес сценария
  % . . . . . . . . . . .
  tr(class, Class, ClassID):-   % Синтез класса
    % Запрос к структурам пакета
    query(_Package_)::class(Name, ClassID),
    create_object(Class, . . . . .% Создание объекта «Класс»
    create_object(Attributes,. . .% Создание атрибута
    create_object(Methods, . . . .% ... метода
    Class::name(Name),             % Поименование класса
    % Порождение атрибутов класса,
    % Представление их в виде локальной базы данных.
    % ... то же с методами ...
    Class::attributes(Attributes),  % Ассоциация атрибутов с классом
    Class::methods(Methods).        % ... и методов тоже..

    % Трансформация атрибутов
  tr(attribute, Attribute, ClassID, AttributeID):-
    query(_Package_)::attribute(Name,ClassID,AttrID),
    create_object(Attribute,  % . . . . .
    Attribute::name(Name).    % Поименование атрибута

    % Трансформация метода
  tr(method, Method, ClassID, MethodID):-
    query(_Package_)::method(Name,ClassID,MethodID),
    create_object(Method,     % . . . . .
    Method::name(Name).       % Поименование метода
:- end_object.
```



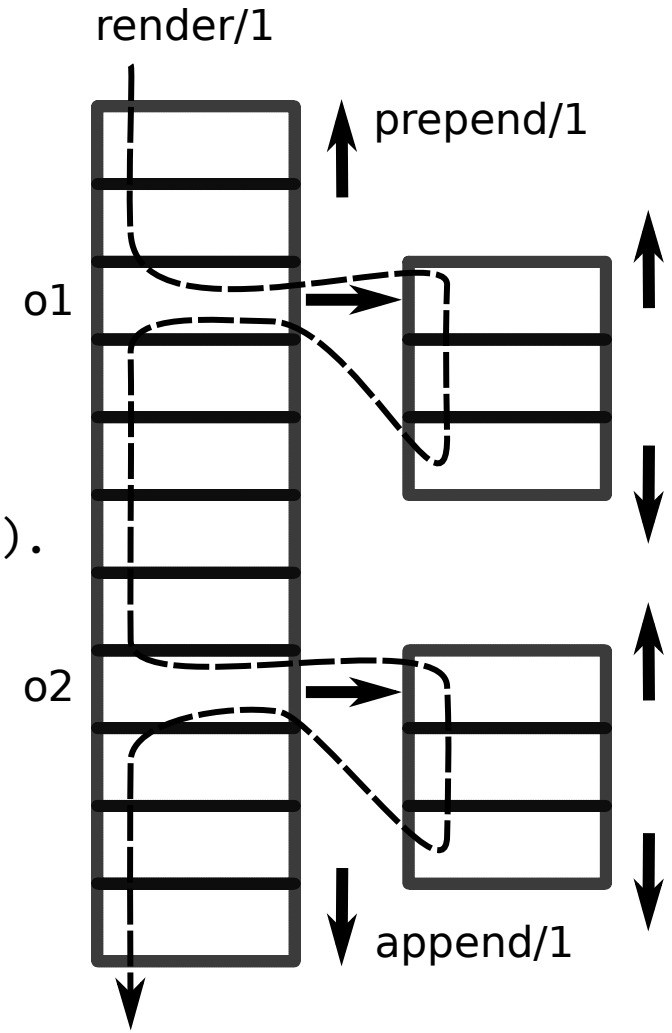
Реализация объекта-фасада query

```
:- object(query(_XMI_)).

  :- public([class/2, attribute/3, method/3]).
  class(Name, ID):-                                % Рассознавание
    _XMI_::rdf(ID,rdf:type,uml:'Class'),           % класса
    _XMI_::rdf(ID,rdfs:label, literal(Name)).

  attribute(Name, ClassID, ID):-                   % ...атрибута...
    _XMI_::rdf(ClassID, xmi:ownedAttribute, ID),
    _XMI_::rdf(ID, rdfs:label, literal(Name)).

  method(Name, ClassID, ID):-                      % ...метода...
    _XMI_::rdf(ClassID, xmi:ownedOperation, ID),
    _XMI_::rdf(ID, rdfs:label, literal(Name)).
  % . . . . . .
:- end_object.
```

Блок кода

Идея реализации взята из `llvmlite`*)

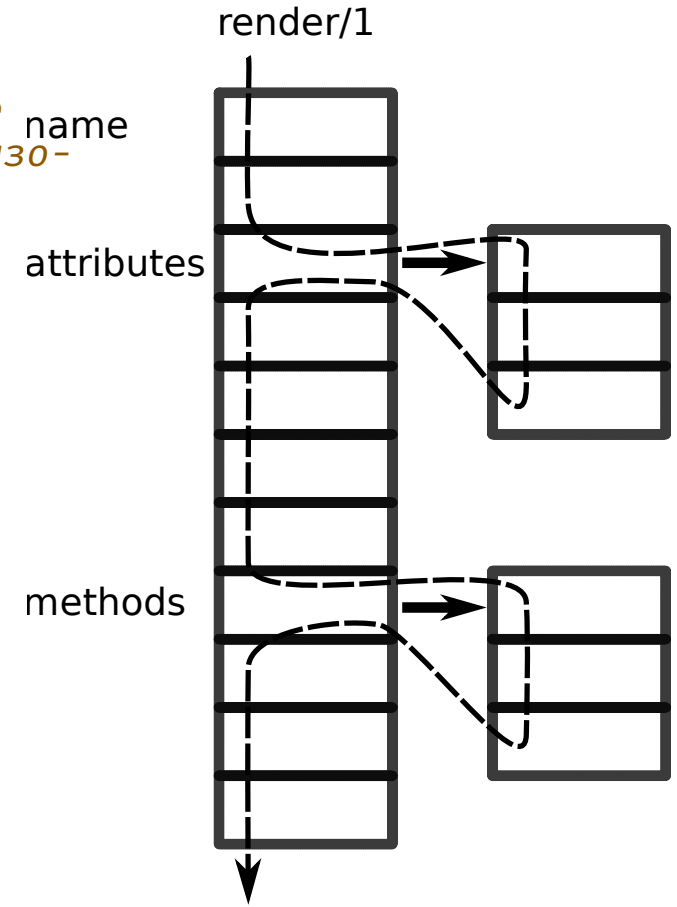
```
:- object(code_block, specializes(root)).
  % Публичный интерфейс объекта
  :- public([append/1, prepend/1, clear/0,
    render/1, render_to/1, remove/1,
    item/1, items/1]).
  % Элементы блока
  :- dynamic([item_/1]).
  :- private([item_/1]).
  % Специализации методов при наследовании
  :- protected([renderitem/2, render_to/2]).
  % Позволить объекту сгенерировать
  % свое представление самостоятельно
  renderitem(Object, String):-
    current_object(Object), !,
    Object::render(String).
  % Преобразовать литерал в строку
  renderitem(literal(Item), String):-!,
    atom_string(Item, String).
  % Отобразить как есть (для отладки).
  renderitem(Item, String):-
    root::iswritef(String, '%q', [Item]).
:- end_object.
```



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

PSM для класса Python как пример блока кода

```
:- object(class, specializes(code_block),
   imports([named])). % Категория поименованных сущностей
  :- public([classlist/1, methods/1, attributes/1]).
  % . . . . . .
  renderitem(Item, Result):-    % Стандартное
    ^^renderitem(Item, Result). % преобразование
  render(Result):-       % Генератор кода, реализо-
    ^^render(Name),      % ванный в категории
    ( ::item(classlist(List)) ->
     % . . . . . .
        [Name]) ),
    ( ::item(attributes(Attributes))->
     % . . . . . .
        [DefAttrList]),
      Attributes::items(InstanceAttrs),
      findall(S, ( % Инициализация атрибутов
         % . . . . . .
         ), AttrAssigns),
        root::unindent,
        AttrList=[ConstructorDef|AttrAssigns];
         % . . . . . .
        AttrList=[ConstructorDef, Pass] ),
    ( ::item(methods(Methods))-> % Если есть ...
      Methods::render(MethodList);
      MethodList=[] ),
    lists::append(AttrList,MethodList,StringList),
    root::unindent, Result=[Signature|StringList].
:- end_object.
```



Категории Logtalk

Категория поименованных сущностей

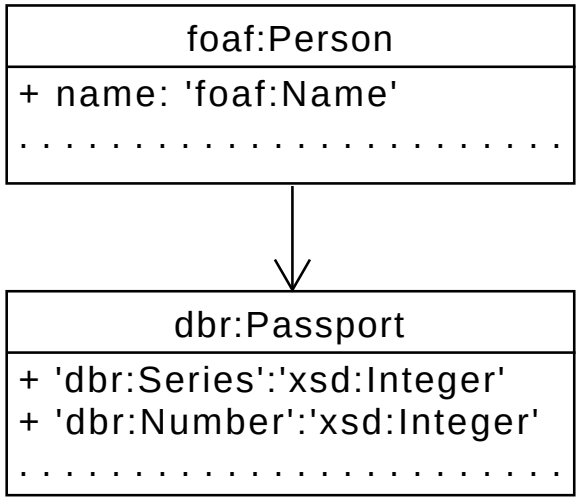
```
:- category(named).
  :- public([name/1, render/1]).
  :- protected([renderitem/2]).
  name(Name):- ::prepend(name(Name)).
  renderitem(name(Name), String):-!, atom_string(Name, String).
  render(String):-  % Порождение кода
    ::item(name(Name)), ::renderitem(name(Name), String).
:-end_category.
```

Категория поименованных типизированных сущностей

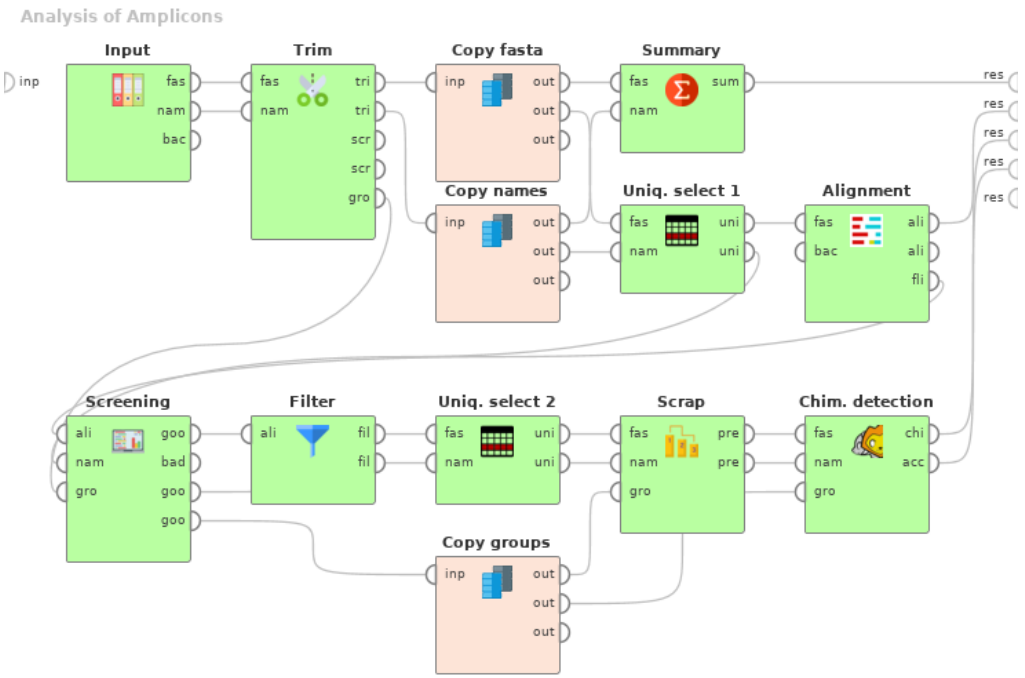
```
:- category(namedtyped, extends(named)).
  :- public([type/1,render/2, separator_option/2,list_separator/1]).
  :- protected([renderitem/2]).
  type(Type):- ::append(type(Type)).
  renderitem(Item, String):- ^^renderitem(Item, String),!.
  renderitem(type(Type),String):-!, ::list_separator(Separator),
    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),!, % Глобальные настройки
      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).  % реализовать
                           % при наследовании.
  :- protected(prepare/3). % подготовка запроса
  % . . . . . .           % в виде строки.
  :- end_category.

  :- object(dbpedia, extends(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 NGS в виде диаграммы DataFlow



Термин	Определение
NGS	Секвенирование нового поколения
Amplicon	Часть ДНК или РНК, скопированная много раз
Mothur	Пакет для исследований в NGS
Rapidminer studio	Визуальный редактор Dataflow-диаграмм

Зеленые блоки – модули Mothur, другие – модули Rapidminer studio.

Использование MDA позволяет актуализировать структуру ПП Mothur (в н.в. 144 модуля).

Обсуждение

Интересные замечания по поводу использования Logtalk:

- Logtalk и RDF гибки и достаточно универсальны для удобной реализации инфраструктуры MDA;
- Наиболее полезным средством Prolog и Logtalk – предикатная и объектная инкапсуляция;
- Не все средства Logtalk исследованы и протестированы: есть возможность разработки специальных методик программирования трансформаций, например, на основе перехвата сообщений.

Свойства языков, которые создают некоторые проблемы:

- Совсем простые задачи решаются трудно, например, обработка текста: преобразовать идентификатор в «CamelCase»;
- Поиск в Internet и исследование спецификаций необходимых онтологий занимает много времени, но оно оправдано по сравнению со временем разработки новой;
- Prolog и Logtalk не является обычным языками в MDA.

Заключение

Результаты, полученные к настоящему времени:

- Разработаны и протестированы методики представления моделей CIM, PIM, PSM, PM.
- Разработана методика представления сценариев трансформации в виде логических объектов.
- Создано ядро библиотеки доступа к модельным данным на основе объектов-фасадов.
- Программы трансформации протестированы, существенных технических проблем не выявлено.

Дальнейшее развитие проекта:

- Создание удобных для программиста инструментов семантической LOD-разметки форм и документов.
- Разработка методик программирования с использованием только статических объектов Logtalk.
- Реализовать библиотеки модулей трансформации для популярных сред, например, web-сред.

Исходный код проекта доступен по ссылкам:
https://github.com/isu-enterprise/icc.xmitransform,
https://github.com/eugeneai/icc.mothurpim.

Спасибо за интерес к проекту!

Сгенерированный модуль Rapidminer studio

```
vector<string> AlignCommand::setParameters(){ // PART OF MODULE SOURCE
try {
  CommandParameter ptemplate("reference", "InputTypes", "", "", "none", "none", "none","",false,true,true); para
  CommandParameter pcandidate("fasta", "InputTypes", "", "", "none", "none", "none","fasta-alignreport-accnos",f
  CommandParameter psearch("search", "Multiple", "kmer-blast-suffix", "kmer", "", "", "","",false,false,true); p
  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);
  }
}
```

CIM Mothur и один из ее фасадных объектов

```
@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):-
    ::second(Parameter),
    rdf_db::rdf_global_object(Value, V),
    _RDF_::rdf(Parameter, NS:Name, V).
  attr(Name, Value):-
    \+ Name=_:_,!,
    ::second(Parameter),
    rdf_db::rdf_global_id(Value, V),
    _RDF_::rdf(Parameter, ngsp:Name, V).
  % . . . . .
:- end_object.
```