Model Driven Architecture Implementation using Linked Data and Digital Archives

Evgeny Cherkashin[*], **Alexey Kopaygorodsky, Ljubica Kazi, Alexey Shigarov, Viacheslav Paramonov**

Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, Irkutsk, Russia
University of Novi Sad, Technical faculty «Mihajlo Pupin», Zrenjanin, Serbia

[*]eugeneai@icc.ru

The 3th August 2019
Irkutsk, Russia

Research objectives

Main objective of the research is to construct a MDA technology based on nowadays system modeling visual languages (SysML, BPMN, CMMN) and existing Semantic Web **vocabularies** and **technologies**. The following techniques and software are under development:

1. CIM representation with SysML, BPMN, CMMN, and results of source code processing,
2. CIM, PIM, PSM representation in RDF with existing vocabularies,
3. transformation implementation with logical language Logtalk,
4. usage of LOD sources in transformations for obtaining additional semantic data,
5. generation of documents and user interfaces with LOD markup.

Related technologies and standards

- The most widely used technologies **ATL** and its predecessor QVT, an OMG standard; developed for XMI to XMI conversion;
- Usage of ATL trending to shift to CIM level, *e.g.*, BPMN diagram is converted into set of UML diagrams (PIM); used for construction web-applications, CIM is represented with State and Use case UML-Diagrams.
- MDA usage is widening, *e.g.*, for analysis of security aspects of distributed application with logical inference over MDA represented model;
- UML is used as CIM/PIM in modeling vocabularies (ontologies); there is an OMG standard specification;
- XML specifications are used to describe services, *e.g.* WSDL, WS-BPEL;
- MDA is opposed to conceptual programming by Enn Tyugu.

We describe transformation in Logtalk, representing CIM, PIM and PSM as RDF graphs, allowing non-XMI sources to be used, organizing multi-stage modular transformations.

Logtalk as transformation definition language

We have chosen Logtalk as it

- inherits widely known Prolog language syntax and runtime;
- implemented as macro package, performance penalties are about 1.5%;
- has flexible semantics: we can define transformations and constraints within the same syntax;
- implement object-oriented knowledge (rules) structuring, encapsulation and replacement;
- compositional way of transformation implementation;
- powerful engine to post constraints on object-to-object messages (events);
- has implementation for many Prolog engines.

The «regular» language allow us to use its libraries not directly related to MDA transformations.

Semantic Web technologies in representation of models during transformation

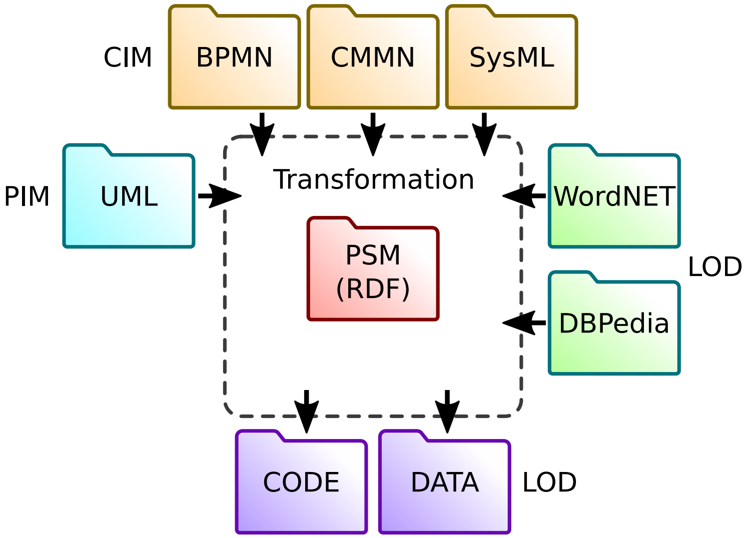
- Assimilates experience of domain basic researches trending to standardization;
- Regular set of triples denote a graph (T-Box, A-Box);
- Standard vocabularies are formally described (`rdfs:domain`, `rdfs:range`);
- Supported with most programming systems (libraries, inference engines, SPARQL);
- RDF has a way of global element identification, *i.e.* we can refer the same object from different software systems;
- SWI-Prolog supports direct queries to a graph, as well as interpreting some predicates (`rdfs:label`, `dc:title`), wraps sparse RDF structure into a predicate arguments; ontological server ClioPatria;
- There is simple way of data security implementation (`rdfs:seeAlso`);
- By means of Semantic Web & LOD we are able to organize data transfer between heterogeneous information systems.

Linked Open Data, LOD

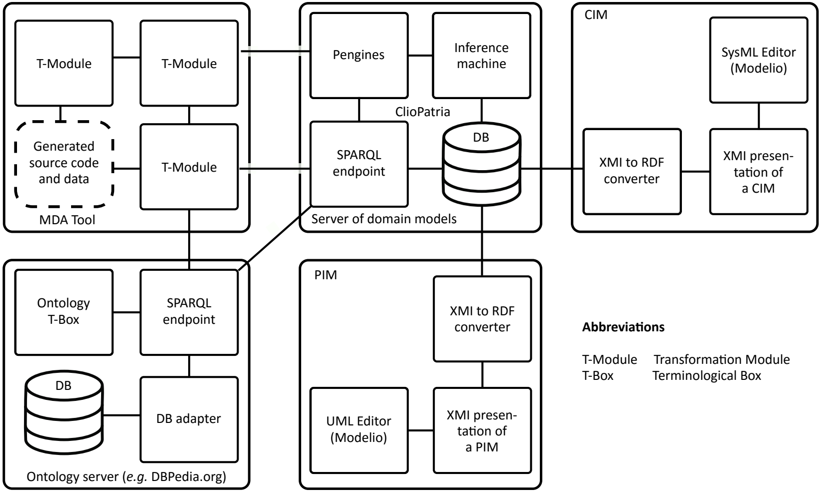
1. Information is published in Internet with open access license;
2. It is represented in a machine-readable form, e.g., Excel table instead of a bitmap picture;
3. An open format used, e.g., CSV instead of Excel;
4. The format is based on W3C recommended standards, allowing RDF and SPARQL reference;
5. Published data refer to objects, forming context.

Thus, applications publish data as relations of objects (entities).

Model Driven Architecture and Linked Open Data



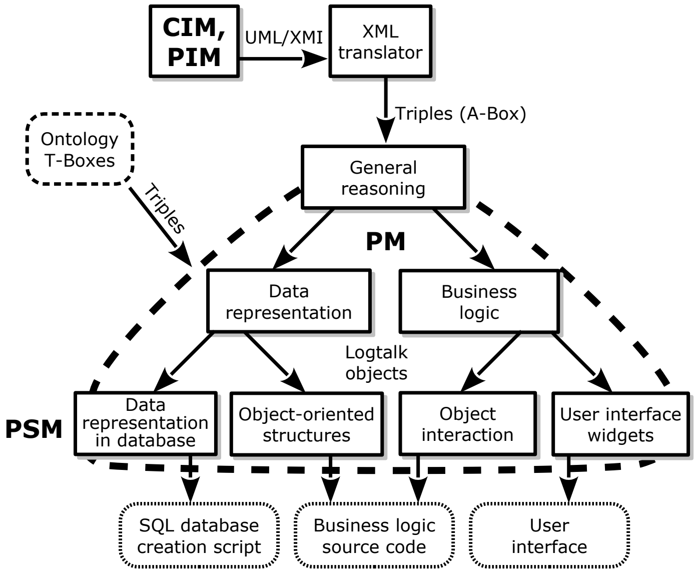
MDA infrastructure



Abbreviations

T-Module	Transformation Module
T-Box	Terminological Box

Architecture of transformation modules

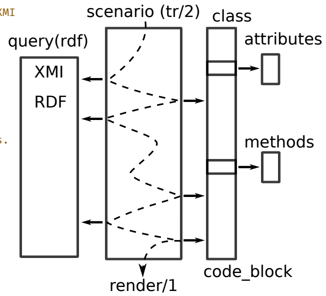


PSM: Scenario of a Class synthesis

```
:- 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.
```

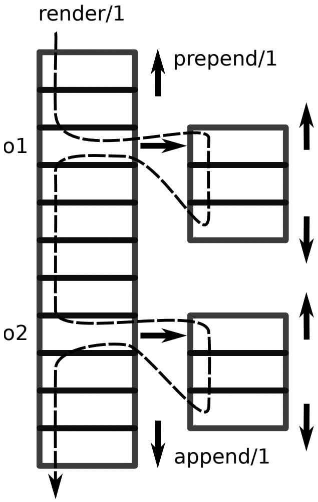


Implementation of Query object

```
:- 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.
```

Code Block (idea is taken from `llvmlite`*)

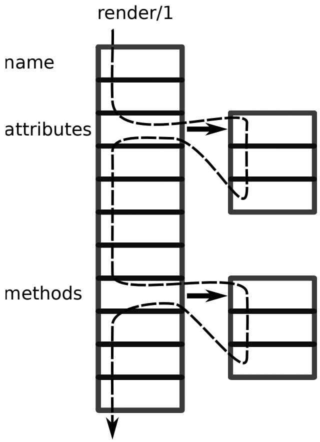
```
:- object(code_block, specializes(root)).
% Public interface of the object
:- public([append/1, prepend/1, clear/0,
   render/1, render_to/1, remove/1,
   item/1, items/1]).
% Code block items
:- dynamic([item_/1]).
:- private([item_/1]).
% Methods specialized during inheritance
:- protected([renderitem/2, 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 of a Python Class as a specialization of 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.
```



Logtalk Categories

A category of named entities

```
:- 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.
```

Category of named and typed entities

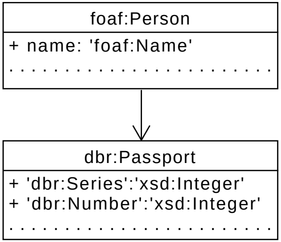
```
:- category(namedtyped, extends(named)).
:- public([type/1,render/2, separator_option/2,list_separator/1]).
:- protected([renderitem/2]).
type(Type):- ::append(type(Type)).
renderitem(Item, String):- ^^renderitem(Item, String),!.
renderitem(type(Type),String):-!, ::list_separator(Separator),
    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.
```

Access LOD data

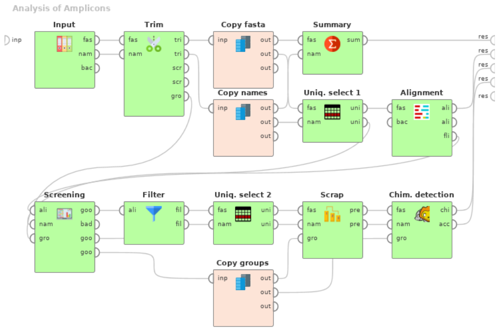
```
:- category(sparql).
:- public(query/2).
query(Pattern,Parameters,Row):-
    prepare(Pattern,Parameters,Query),
    server(Host,Port,Path),
    sparql_query(Query, Row,
        [host(Host),port(Port),path(Path)]).
:- protected(server/3).  % must be implemented
                         % by a subclass.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

:- object(dbpedia, 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).
```



Applications: Dataflow representation of NGS analysis of amplicons



Term	Description
NGS	New Generation Sequencing
Amplicon	A DNA or RNA part copied many times
Mothur	A software toolset for NGS research
Rapidminer	A visual tool for data mining modeling and execution

Green blocks are Mothur modules.
Others are Rapidminer modules.

Discussion

Interesting positive impressions obtained:

- Logtalk and RDF are flexible, sufficiently universal and convenient implementation infrastructures for MDA;
- The best implemenation means is Prolog predicate wrapping and Logtalk object encapsulation of rules;
- Not all Logtalk properties are investigated: there might be more sophisticated programming techniques developed, *e.g.*, on the base of message watchers.

Technical problems making the approach somewhat problematic:

- Very simple tasks take too much efforts, *e.g.*, text processing: convert an identifier into the CamelCase;
- It takes too long to surf Internet in order to find a vocabulary for a domain, but it is more productive than development;
- Prolog is not a popular language in MDA, neither Logtalk.

Document authoring and storage

In most cases documents are created as a result of

- creative activity of a person with a text processors (authoring);
- printing a digital copy or a data record in a database;
- aggregation operation over database records (report).

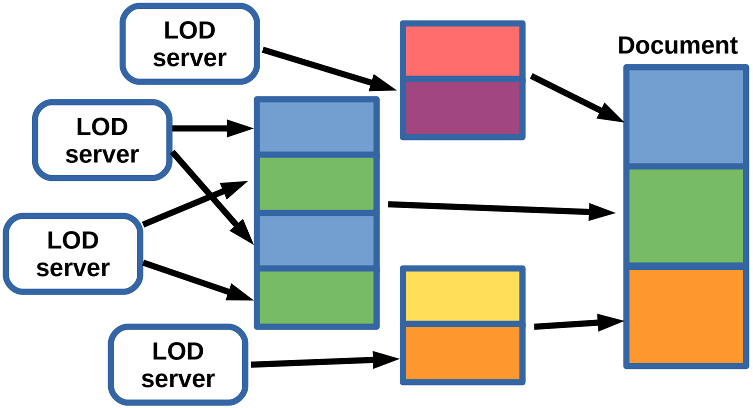
Then it is stored either as a physical paper and/or a digital document (PDF, DOCX, HTML).

Since 2000-th, Semantic Web and Linked Open Data (LOD) is being developed, allowing

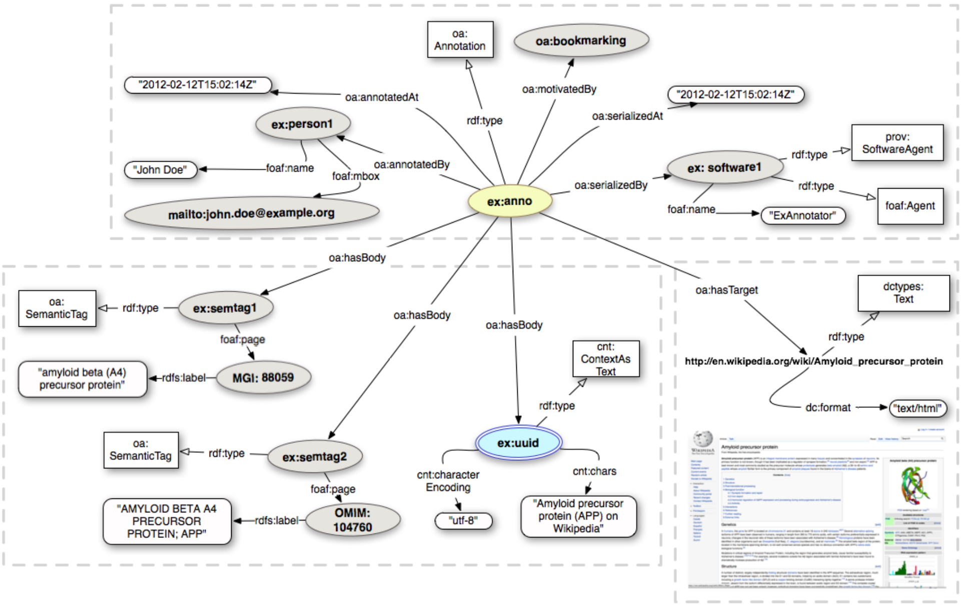
- structural storage of data within published documents;
- processing stored data computationally;
- integration of data structures and data objects globally.

The **aim of this research** is to develop technologies, software and services allowing construction of digital archives supporting document data inclusion and inference from existing documents.

Structure of a document



Open Annotaiton (oa)

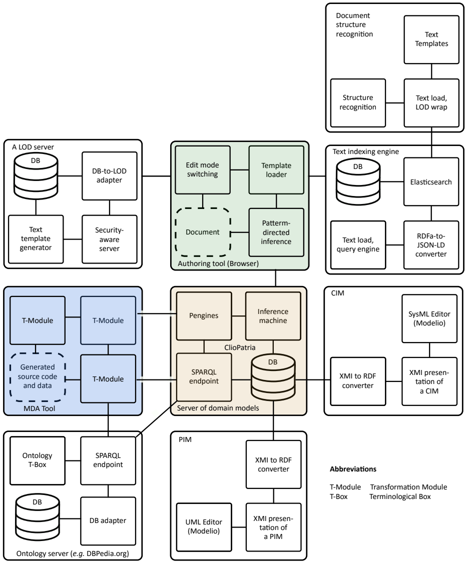


Representation

```
<html lang="ru" xmlns=http://www.w3.org/1999/xhtml
 xmlns:taa =http://irnok.net/engine/rdfa-manipulation
xml:lang="ru" metal:define-macro="page">
<head> . . . . </head>
<body prefix="rdf: http://www.w3.org/1999/...-ns# foaf: http://xmlns.com/foaf/...
imei: imei.html# course: https://irnok.net/college/plan/01..16-...\
%D0\%BA_PB-SM.plm.xml.xlsx-....2.3.1.html#"  resource="#post"
typeof="schema:CreativeWork sioc:Post prov:Entity">
<!– The application control panel –>

<main lang="ru" resource="#annotation" typeof="oa:Annotation" id="main-doc-cnt">
<div property="oa:hasTarget" resource="#course-work-prog"></div>
<article property="oa:hasBody" typeof="foaf:Document curr:WorkingProgram"
         resource="#course-work-program" id="main-document">
  <div  taa:content ="imei:title-page"></div>
  <div  taa:content ="imei:neg-UMK"></div>
  <section id="TOC" class="break-after"> <h2>Table of Contents</h2>
    <div id="tableOfContents"></div>
  </section>
  <section id="course-description" resource="#description"
          property="schema:hasPart" typeof="schema:CreativeWork">
    <div property="schema:hasPart" resource="#purpose"
         typeof="dc:Text cnt:ContentAsText" >
      <div property="cnt:chars" datatype="xsd:string">
        <h2 property="dc:title" datatype="xsd:string">
          Aims and objectives of the discipline (module)</h2>
        <p>The aim of teaching the discipline ...</p>
      </div>
   </div>
  . . . . . . . .
```

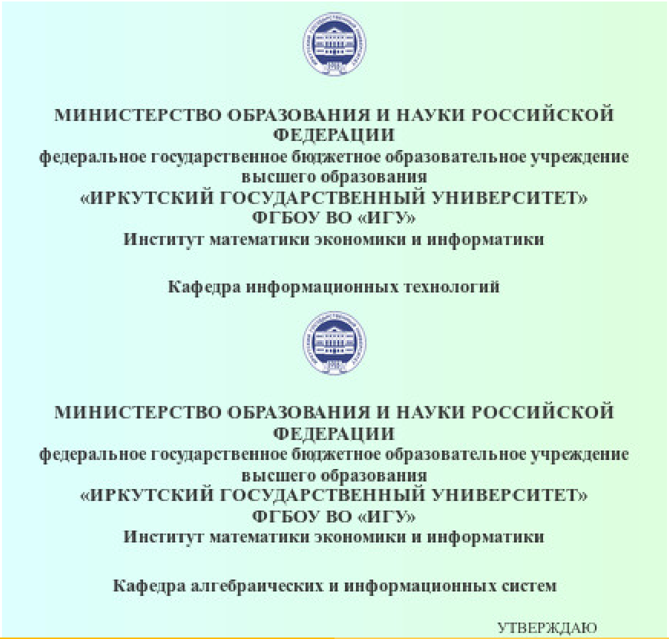
Architecture



Abbreviations

T-Module	Transformation Module
T-Box	Terminological Box

Generated list of title page preambles



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФГБОУ ВО «ИГУ»
Институт математики экономики и информатики

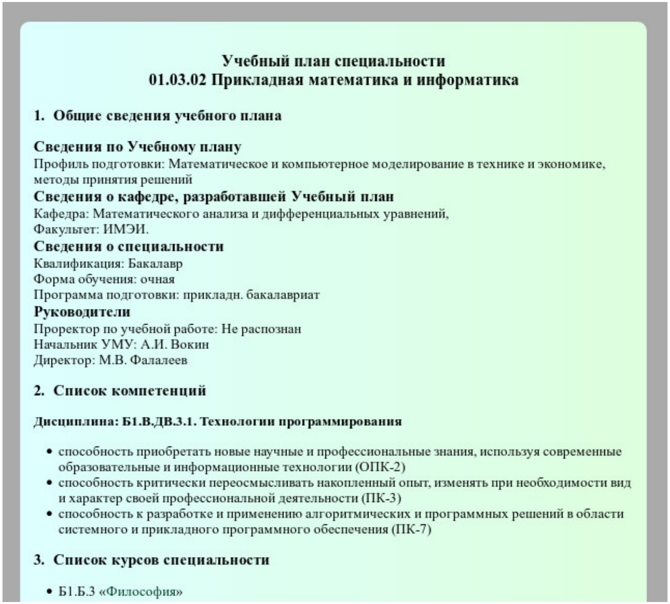
Кафедра информационных технологий

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФГБОУ ВО «ИГУ»
Институт математики экономики и информатики

Кафедра алгебраических и информационных систем

УТВЕРЖДАЮ

Generated part of a study program



Учебный план специальности
01.03.02 Прикладная математика и информатика

1. Общие сведения учебного плана

Сведения по Учебному плану
Профиль подготовки: Математическое и компьютерное моделирование в технике и экономике, методы принятия решений
Сведения о кафедре, разработавшей Учебный план
Кафедра: Математического анализа и дифференциальных уравнений,
Факультет: ИМЭИ.
Сведения о специальности
Квалификация: Бакалавр
Форма обучения: очная
Программа подготовки: прикладн. бакалавриат
Руководители
Проректор по учебной работе: Не распознан
Начальник УМУ: А.И. Вокин
Директор: М.В. Фалалеев

2. Список компетенций

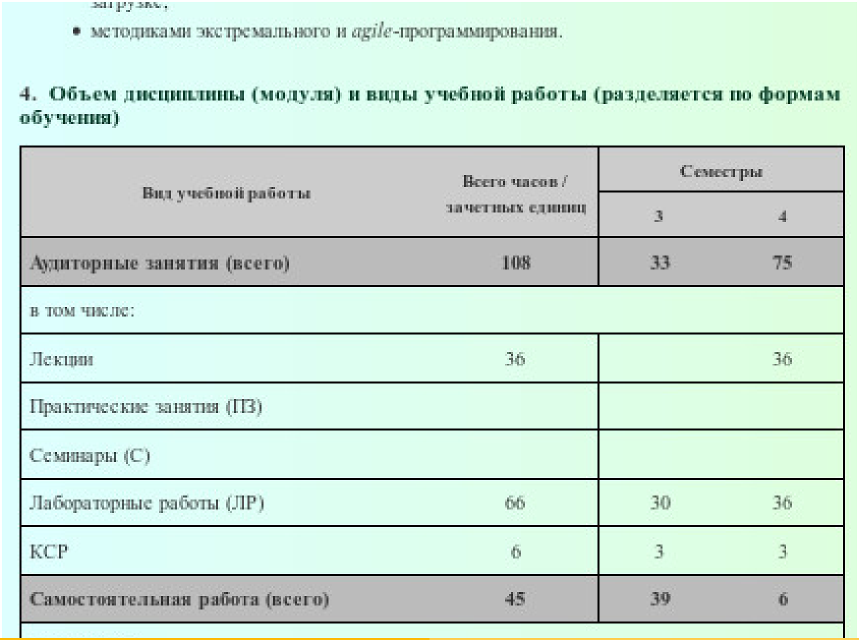
Дисциплина: Б1.В.ДВ.3.1. Технологии программирования

- способность приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии (ОПК-2)
- способность критически переосмысливать накопленный опыт, изменять при необходимости вид и характер своей профессиональной деятельности (ПК-3)
- способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения (ПК-7)

3. Список курсов специальности

- Б1.Б.3 «Философия»

Imported time distribution for lecture, seminary, ...



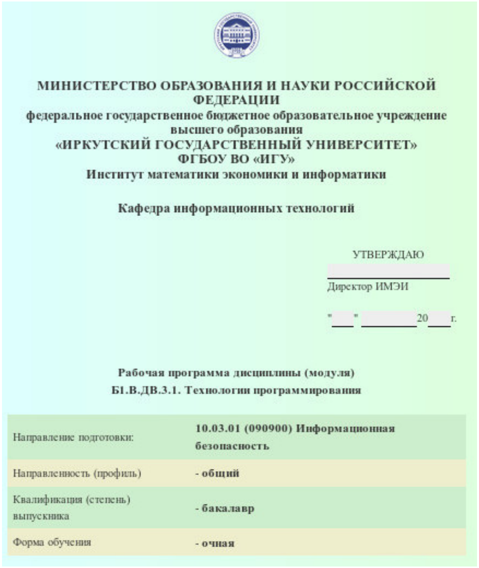
загрузке;

- методиками экстремального и *agile*-программирования.

4. Объем дисциплины (модуля) и виды учебной работы (разделяется по формам обучения)

Вид учебной работы	Всего часов / зачетных единиц	Семестры	
		3	4
Аудиторные занятия (всего)	**108**	**33**	**75**
в том числе:			
Лекции	36		36
Практические занятия (ПЗ)			
Семинары (С)			
Лабораторные работы (ЛР)	66	30	36
КСР	6	3	3
Самостоятельная работа (всего)	**45**	**39**	**6**

Complete document



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФГБОУ ВО «ИГУ»
Институт математики экономики и информатики

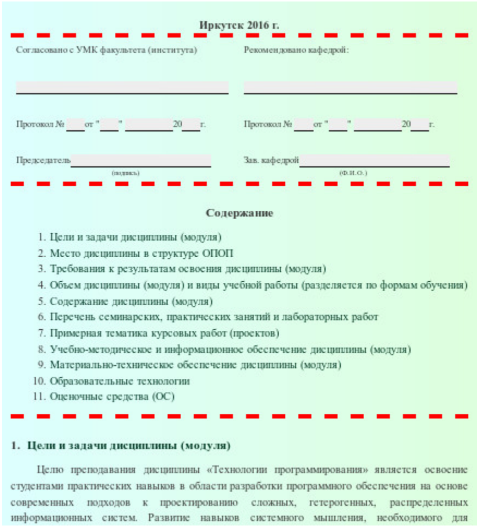
Кафедра информационных технологий

УТВЕРЖДАЮ
____
Директор ИМЭИ

"___" ________ 20___ г.

Рабочая программа дисциплины (модуля)
Б1.В.ДВ.3.1. Технологии программирования

Направление подготовки:	10.03.01 (090900) Информационная безопасность
Направленность (профиль)	- общий
Квалификация (степень) выпускника	- бакалавр
Форма обучения	- очная



Иркутск 2016 г.

Согласовано с УМК факультета (института)

____

Протокол № ___ от "___" ________ 20___ г.

Председатель ____________
(подпись)

Рекомендовано кафедрой:

____

Протокол № ___ от "___" ________ 20___ г.

Зав. кафедрой ____________
(Ф.И.О.)

Содержание

1. **Цели и задачи дисциплины (модуля)**

Целю преподавания дисциплины «Технологии программирования» является освоение студентами практических навыков в области разработки программного обеспечения на основе современных подходов к проектированию сложных, гетерогенных, распределенных информационных систем. Развитие навыков системного мышления, необходимого для

Used ontologies

- Friend-of-a-friend (**foaf**) - agent information: individuals, legal entities, program agents.
- Provenance (**prov**) - references between documents.
- Dublin Core (**dc**) - edited annotation mark up.
- DBPedia resource (**dbr**) – references to instant objects and classes.
- Schema.org (**schema**) - Google, Yandex, Yahoo, *etc.* searchable objects, structural elements.
- The Bibliographic Ontology (**bibo**) - literature reference mark up.

Conclusion

A tools (components) for digital archive implementation, which allows to device information systems and document processing services with the following features:

- load LOD marked up document, extract, store in a graph and index RDF data;
- retrieve RDF data as triples or as a result of full-text search query;
- combine existing LOD data and its content in new documents dynamically with browser based context inference machine;
- use server-site inference machine (Prolog) to process RDF data upon request from browser's part of the system;
- convert created RDFa marked up HTML5 documents into Excel and Word formats.

Applications

- Document authoring automation;
- Context-depended editing;
- Self-organizing global document flows;
- Documents as data sources for information systems.

Conclusion

The following results have been obtained as for today:

- A technique for model representation has been developed and tested.
- A programming technique using object-oriented logical language Logtalk is deviced.
- Ptototypes of various transformation procedures are implemented.
- Transformation tools are tested in application areas and no significant technical problems were mentioned.

Further development directions are as follows:

- A technique for document automatic markup with vocabulary entities.
- A transformation implemenation techniques, minimizing usage of dynamic objects, targeting on macro properties of Logtalk.
- Form a toolset out of existing prototypes obeing nowadays software development requirements.

The source codes are available at
`https://github.com/isu-enterprise/icc.xmitransform`,
`https://github.com/eugeneai/icc.mothurpim`.

Thanks for Your interest to our project!

Rapidminer module

```
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);
  }
}
```

RDF (TTL) representation and ad its query object

```
@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).
% . . . . .
:- end_object.
```