A COMPUTATIONAL INDEPENDENT MODEL FOR A MEDICAL QUALITY MANAGEMENT INFORMATION SYSTEM

Evgeny Cherkashin[1,2,4], Ljubica Kazi[3], Alexey Shigarov[1,2], and Viacheslav Paramonov[2]

[1] Irkutsk Scientific Center of SB RAS, 134 Lermontov Street, Irkutsk, 664033, Russia
[2] Matrosov Institute for System Dynamics and Control Theory of SB RAS, 134 Lermontov Street, Irkutsk, 664033, Russia
[3] University of Novi Sad, Technical faculty "Mihajlo Pupin", Đure Đakovića bb, Zrenjanin, 23000, Serbia
[4] National Research Irkutsk State Technical University, 83 Lermontov Street, Irkutsk, 664074, Russia
{eugeneai,shig,slv}@icc.ru, ljubica.kazi@gmail.com

Abstract. We consider problem of Quality Management System (QMS) development for Irkutsk Regional Oncological Dispancery. The system is intended to organize the process management of medical treatment according to the standard ISO 9001:2015. QMS software subsystems are synthesized as a result of a logical inference of a set of subgoals (a scenario) with a hierarchy of modules represented in the LogTalk programming language within Model Driven Architecture (MDA) paradigm. The source Computational Independent Model is represented as a set of BPMN2.0, SysML and CMMN diagrams. The models are imported and stored as ontology A-boxes on an ontology server. The models are transformed into Platform Specific Models with following source code and initial data population. Additional data for the transformation are provided from Linked Open Data compliant sources. Usage of such kind of MDA allows us to develop QMS on the abstract model level in most cases and involve domain specialist in the formal part of the development. Rengineering business processes of the medical institution should result in patient treatment quality improvement, as well as formalizing the processes, giving ideas for automation, data accumulation and processing.

Keywords: quality management systems, model driven architecture, logical inference, linked open data

1 Introduction

International certification is one of the activities of an organization in the improvement of the internal business processes and the collaboration with other counterparties. The correspondence to the ISO 9001:2015 standards for a medical institution is an evidence of its management conformity to the highest international degree of quality (recognized in 157 countries), which is related to the used technical equipment. Standard ISO 9001:2015 [1] specifies requirements for a quality management system when an organization

a) needs to demonstrate its ability to consistently provide products and services that meet customer and applicable statutory and regulatory requirements,
b) aims to enhance customer satisfaction through the effective application of the system, including processes for improvement of the system and the assurance of conformity to customer and applicable statutory and regulatory requirements.

All the requirements of ISO 9001:2015 are generic and are intended to be applicable to any organization regardless of its type or size, the products and services it provides. The introduction of the standard into a medical institution management ensures an improvement of its performance with economic costs remain minimized. The quality of the medical service become much higher, as well as working, financial and psychological conditions. According to William Edwards Deming [2], 96% of the problems caused by faulty management system and only 4% is staff errors. Also, the main profits of international certification for the institutions is the higher degree of credit of counterparties, namely, patients. The certification results in

– formal quality recognition in Russia and other countries;
– better positions in tenders;
– better positions within competitors;
– preferences by foreigners working in Russia to cooperate with international level organizations;
– being a part of the club of cooperating institutions with preferences;
– a formal statement of conformation to the sectoral standards;
– analysis of the structure of the institution, better understanding its objectives and process.

Internal business processes of certified institution conforms to a general scheme depicted in Fig. 1.

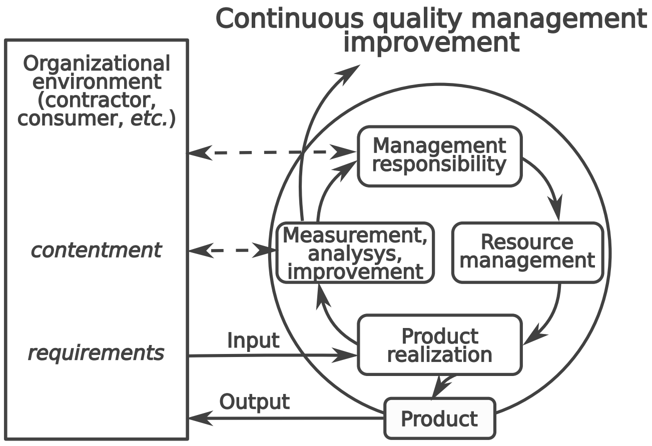
Continuous quality management improvement
Organizational environment (contractor, consumer, *etc.*)
contentment
requirements
Input
Output
Management responsibility
Measurement, analysys, improvement
Resource management
Product realization
Product

Fig. 1. A general view on the ISO 9001:2015

Standard ISO 9001 considers each business process (BP) in a set of aspects. A BP *realizes* a *product* according to *requirements* stated by counterparties or *organizational environment*. The result of the production and customers' contentment *are measured and analyzed* forming *improvement* recommendations for *management staff*. Management staff works out decisions on *resource management* and BP improvement, accounting customers' recommendations.

ISO 9001 QMS implements practical approach of multistep process structure reconfiguration, locally and sequentially improving it in a subsystem. The basis of a decision forming is analysis of *customer needs* solved by a product and its *quality criteria*. These are named as *counterparties expectations*. The management stuff determines *unities of purpose* and direction of the organization functioning and raises the *engagement of people*, aligning people's and organization's needs. Since a model of the production is represented as a *system of process*, it allows managers in changing condition to maintain the functionality and improve the system locally and in several independent parts simultaneously to generate various new opportunities. The improvement is based on *decisions*, which, in turn, are based on *evidences*, *e.g.* measurements. The outer aspect of the organization sustainable functioning is expressed with *relationship management* within *resource approach*. These terms are the reference points for information system supporting the rengineering of the organizational structure.

As of today the organizing structure of Irkutsk Regional Oncological Dispancery (IROD) is described as a set of documents presenting the business processes as hierarchy structures of the BP and, in parallel, a block diagrams of algorithmic schemes of the process element relations in time. This approach corresponds the present modeling technologies taught to students of management occupations in Irkutsk universities. The approach ought to be improved with more expressive but complex notions of SysML (System Markup Language), BPMN–2.0 (Business Process Modeling Notation) and CMMN (Case Management Modeling Notation). SysML used to describe organizational structures in higher degree of abstraction with respect to UML2 and in the same time to be more detailed. The usage of these modeling notation will allow to describe the organizational structure of IROD more formally and precisely as a Computational Independent Model of MDA paradigm.

Heaving described BPs structure and flows of IROD in the formal detailed multilevel form in SysML, we could construct an environment of a permanent Informational System (IS) research and development. At the first stage, a most simple case is being realized: the description could be converted into check lists controlling conditions of business process starting and ending points, *e.g.*, check whether all the manipulation and medicines were applied to a patient before surgery. The second stage of implementation is the automation of the patient medical care planning after the surgery. The third stage is to organize a CRM subsystem for patients, *e.g.* arranging appointments at medical office and proceed with a medical care. In the long run a refined SysML description and the IS would include integration with Laboratory IS installed at the hospital. All the stages are accompanied with a lot of useful information to be accumulated, forming the quality measurement and analysis basis [3].

The *object* of the research is to develop generative technology of expression of process structure of a medical institution in abovementioned notations (SysML, BPMN-2.0, CMMN) and conversions of the description into subsystems of a complex information system supporting the BP reingeneering processes. The *subject* we consider in this paper is to describe tools and techniques of their applications in the technologies as being represented within MDA.

2 Architecture of MDA tools

The architecture of the software development tools is based on the provision of uniformity of the transformation rules language and predicate input data representation. That's why we require to store all the input model data in Semantic Web graphs of ontologies. The ontologies could be served as RDF–files of triples or with servers having SPARQL endpoints. A general architecture of the development tools is presented in Fig. 2.

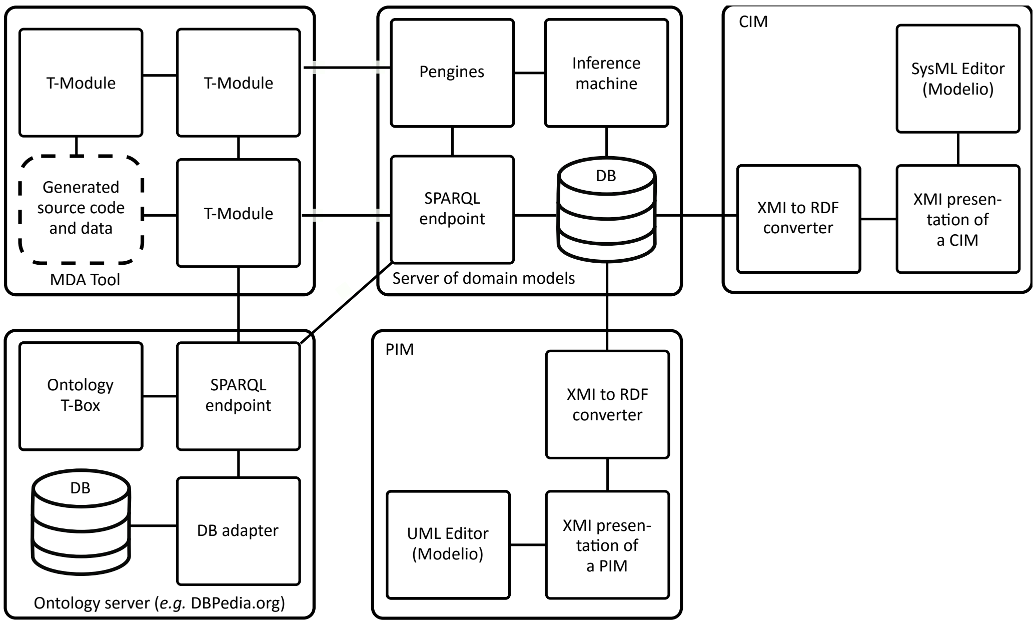

Fig. 2. Architecture of MDA tools

The kernel of the system is MDA Tool, which is a set of transformational modules (T–Modules). Each of T–Module in a general case is a parameterized LogTalk object. The object queries the Server of domain models, which is built on the base of Cliopatria [4], other objects and other Ontology servers for structural patterns. These patterns are analyzed and the results are stored as the objects states, *i.e.* cached. The states of the object represent Platform Specific Model (PSM), which is translated into source code and initial data. Input structures of CIM and PIM models are converted into graphs with converter adapters. For example in Fig. 2 CIM drawn in Modelio [5] is converted to triples with a special module “XMI to RDF converter”. Similar converter is realized for PIM represented in UML. This approach makes the input stage modular, and, consequently, easier extensible with new models.

Another important source of information for the synthesizer is ontologies describing various abstract and concrete special domains. The Semantic Web technologies, namely Linked Open Data, allow us to infer the properties of generated structural elements. User interface visual element attributes (a part of PSM) can be partially figured out by means of SPARQL queries to `DBPedia.org`, which contains naming of entities in various languages, as well as other useful relationships. For example, we can use DBPedia for constructing `title` and `placeholder` attributes of an input widget, together with its label in user’s locale and a help description.

3 Related work: SysML usage in system design and synthesis

SysML is the dedicated system-level UML-based notation proposed by the OMG. In research [6] system design productivity is addressed as one of the main challenges. Some suggested approaches are increasing the level of abstraction and automation, as well as producing executable specifications. In [7], model driven development is used for the construction of complex embedded systems integrating software and firmware. A SysML system model is devised according to the platform-based design paradigm, in which a functional model of the system is paired to a model of the execution platform. Subsystems are refined as Simulink models or hand-coded in C++.

The SysML standard is attracting more attention of hardware designers as UML and SysML have been used to automatically generate an HDL code written in SystemC, Verilog and VHDL. In [8], contrarily to the existing works, authors propose the new reverse engineering approach to generate SysML definition of block and internal block diagrams from VHDL code. Code generation is done on the basis of a set of well-defined mapping rules between SysML and VHDL concepts.

UML profiles like SysML and MARTE have been a major research topic in electronic system design, but are mainly applied for specification and analysis in early design phases. Research [9] addresses the problem of the High-Level Synthesis (HLS), *i.e.* physical implementation aspect of electronic systems, which need diversity of design models and levels of abstraction. To overcome the conflict between a higher degree of abstraction and necessary details for further synthesis, modular interfaces are introduced as object-oriented synthesizable technique. In [9], SysML is used as an adequate modeling language for modular interfaces and C/C++/SystemC-based HLS. Authors extended SysML with annotations for synthesizable SystemC and high-level synthesis constraints and implemented a code generation scheme to achieve design flow automation. They used SysML editor Artisan Studio with industrial case study to demonstrate the applicability of SysML as a front-end for HLS.

In [10] UML/SysML is used to model avionics. The implementation language is C# with using libXML for XMI import and processing. Generated objects represent each syntactic element (`struct`, `enum`, function, *etc.*), which represents the target source code. Authors implemented only State Machine diagram transformation.

Some experience of representing document flows are obtained by commercial software vendors developing Directum, Documentum systems automatizing Russian municipal institutions and state corporations. Directum uses SysML state–like block diagrams to define flow of documents and the ISBL object-oriented language to express sophisticated behavior.

In our research we focus on representation of ISO 9001 entities in SysML and other DSLs translatable to RDF, and the MDA transformation logical inference representable as LogTalk objects.

4 SysML representation of QMS

QMS is represented in SysML with nine diagrams [11]. QMS requirements is represented with new (with respect to UML) *Requirement Diagram*, describing stated structural and logical constraints. Structural aspects are represented with following UML2 diagrams:

– *Block Definition Diagram* decomposes structure of activities, attributes and objects on elements;
– *Internal Block Diagram* models interaction of the decomposed elements and their data flow; its descendant new *Parametric Diagram* describes model of block attribute relationships;
– *Package Diagram* organizes models in packages in various ways.

Behavioral aspect of a QMS is expressed with UML2

– *Use Case Diagram*, which expresses functions of the system and their mutual relationships;
– *State Machine Diagram* defines the states of the blocks and transitions between the states;
– *Sequence Diagram* describes the order of messages between blocks and actors;
– *Activity Diagram* represents computational process of converting data enclosed in blocks.

BPMN2.0 diagram is used to describe imperative procedures, it is very expressive diagram representing functions and involving agents, together with their relationships. CMMN represents declarative aspects of the QMS especially data structures, processing stages, events and check points (milestones). Adding BPMN2.0 and CMMN diagrams to SysML diagram set creates a redundancy, but in the same time it allows one to use more rich tool set for QMS modeling.

5 Defining rules of transformation

Transformation process is organized as a scenario of connected transformational objects (in terms of LogTalk) [12]. Scenario is programmed as a special object defining transformation by means of calling `tr/N` rules. Rules recognize structures in the input graph elements and constructs code blocks.

```
:- object(direct(_Package,_LocalProf,_CodeProf)).
:- public([tr/4,tr/3]).
:- protected([package/1, profiles/2, profile/1]).
package(Package):- parameter(1, Package).
profile(Profile):- parameter(2, Profile).
profile(Profile):- parameter(3, Profile).
profiles(L):-
    findall(Profile, ::profile(Profile), L).
tr(class, Class, ClassID):- ::package(Package),
    query(Package)::class(Name, ClassID),
    create_object(Class,
        [instantiates(class)],[],[]),
    create_object(Attributes,
        [instantiates(params)],[],[]),
    create_object(Methods,
        [instantiates(methodlist)],[],[]),
    Class::name(Name),
    forall(
        ::tr(attribute,Attribute,ClassID,_AttrID),
        Attributes::append(Attribute) ),
    forall(
        ::tr(method, Method, ClassID, _MethodID),
        Methods::append(Method) ),
    Class::attributes(Attributes),
    Class::methods(Methods).
tr(attribute, Attribute, ClassID, AttributeID):-
    ::package(Package),
    query(Package)::attribute(Name,ClassID,AttrID),
    create_object(Attribute,
        [instantiates(param)],[],[]),
    Attribute::name(Name).
tr(method, Method, ClassID, MethodID):-
    ::package(Package),
    query(Package)::method(Name,ClassID,MethodID),
    create_object(Method,
    [instantiates(method)],[],[]),
    Method::name(Name).
:- end_object.
```

The previous listing uses static parameterized object `query/1`, whose argument is a graph representing model under transformation. This is a powerful LogTalk abstraction instrument, which reduces necessity of adapter object creation in dynamic memory. The SPARQL is encapsulated in methods.

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

Procedure of source code generation is build using `code_block`–objects (the idea was taken from `llvmlite` library), which provide PSM representation:

```
:- object(code_block, specializes(root)).
:- public([append/1, prepend/1, clear/0,
   render/1, render_to/1, remove/1, item/1,
   items/1 ]).
:- dynamic([item_/1]). % elements of
:- private([item_/1]). % the code block
:- protected([renderitem/2, render_to/2]).

item(Item)   :- ::item_(Item).
items(Items) :- bagof(I, ::item(I), Items).
append(Item) :- ::assertz(item_(Item)).
prepend(Item):- ::asserta(item_(Item)).
remove(Item) :- ::retract(item_(Item)).
clear        :- ::retractall(item_(_)).
render(_)    :- writef::writef("ERROR: \
Implement render/1 by a subclass!\n"), fail.
```

```
render_to(Stream):- ::render(List),
    ::render_to(List, Stream).
render_to(List, Stream):-
    lists::is_list(List),!,
    forall(lists::member(X,List),
       ::render_to(X, Stream)).
render_to(X,_) :- write(X),nl.
renderitem(Object, String):-
    current_object(Object), !,
    Object::render(String).
renderitem(literal(Item), String):-!,
    atom_string(Item, String).
renderitem(Item, String):-
    root::iswritef(String, '%q', [Item]).
:- end_object.
```

The code_block–objects consist of source code string lines and other objects, including other code blocks. Each item is denoted with item/1 and its private database element item_/1. The type of the element is defined by outer functor of the argument, *e.g.*, we define with attributes(L) a list L of attributes of a class. The way of rendering the sources is implemented by subclassing the object and realizing render/1 and renderitem/2. The first argument of renderitem/2 is the item/1–structure to be rendered, and the second one is the list of strings representing generated source code. Elements of the code block can be appended, prepended and removeed. We specially disallow to insert items into the database, as it will make object and its database more complex. Instead, programmer can add an item being another code block, inserting elements of the sources in it.

The following example shows rendering class for Python classes. It contains public methods for defining class elements.

```
:- object(class, specializes(code_block),
   imports([named])). % A category of named entities
:- public([classlist/1, methods/1, attributes/1]).
classlist(ClassList):- % List of base classes
    ::prepend(classlist(ClassList)).
attributes(Attributes):- % List of attributes
    ::prepend(attributes(Attributes)).
methods(MethodList):- % List of methods
    ::append(methods(MethodList)).

renderitem(Item, Result):- % Default
    ^^renderitem(Item, Result).
render(Result):- % Render the class
    ^^render(Name),
    ( ::item(classlist(List)) ->
      List::render(ClassList),
      root::iswritef(Signature,'class %w(%w):',
        [Name, ClassList]);
      root::iswritef(Signature,'class %w:',
        [Name]) ),
    root::indent, % add an indent
    ( ::item(attributes(Attributes))->
      Attributes::render(DefAttrList),
      root::iswritef(ConstructorDef,
```

```
        'def __init__(self, %w):',
        [DefAttrList]),
      root::indent,  % more indent
      Attributes::items(InstanceAttrs),
      findall(S, ( % initialize attributes
         lists::member(Attr, InstanceAttrs),
         Attr::item(name(AttrName)),
         root::iswritef(S, "self.%w=%w",
           [AttrName, AttrName])
         ), AttrAssigns),
        root::unindent,
        AttrList=[ConstructorDef|AttrAssigns];
        root::iswritef(ConstructorDef,
          'def __init__(self): ', []),
        root::indent,
        root::iswritef(Pass,'pass', []),
        root::unindent,
        AttrList=[ConstructorDef, Pass] ),
    ( ::item(methods(Methods))-> % if any ...
      Methods::render(MethodList);
      MethodList=[] ),
    lists::append(AttrList,MethodList,StringList),
    root::unindent, Result=[Signature|StringList].
:- end_object.
```

In the same way, we can construct procedures for creating structures of databases and populating them with initial data.

6 Conclusion

The idea of formal description of the business process during transition of a medical institution for compliance to ISO 9001:2015 standard is considered in the paper. Diagrams SysML, BPMN2.0, CMMN are proposed to be used for this purpose. The description provides Computational Independent Model (CIM) of Model Driven Architecture (MDA) software development approach, and it is to be transformed in subsystems of Informational System (IS).

CIM is being designed with visual editors, *e.g.* Modelio [5], translated from their XMI to RDF and stored as ontology graphs as files or network resources. MDA-transformation is represented and executed within Prolog programming environment LogTalk [12], which provides us knowledge structuring, rich set of libraries and uniform well-known programming language for defining transformation rules. The general scheme of transformation definition is presented.

IS development tools is being tested in representation of bioinformatic tool Mothur as Rapidminer dataflow diagrams [13]. The source data for transformation is the module specifications and UML Class Diagram, which organizes submodules in virtual class hierarchies.

7 Acknowledgments

The results are obtained with the partial support of the various projects: Irkutsk scientific center of SB RAS No 4.2; Russian Foundation for Basic Research, grants 17-07-01341, 18-07-00758 and 17-47-380007. The results obtained with the use of the network infrastructure of Telecommunication center of collective use “Integrated information-computational network of Irkutsk scientific-educational complex” (`http://net.icc.ru`).

References

1. ISO 9001:2015 standard. URL:`(https://www.iso.org/standard/62085.html)`
2. *Lynn,M.*, *Osborn,D.* Deming’s quality principles: a health care application. Hosp Health Serv Adm. 1991 Spring 36(1), p. 111-20. URL:`https://www.ncbi.nlm.nih.gov/pubmed/10108969`
3. *Levet,J.M.* Implementing an ISO 9001 Quality Management System in a Multispecialty Clinic. 2005. 5 p. `http://rube.asq.org/health/implementing-an-iso-9001-quality-management-system-in-a-multispecialty-clinic.pdf`
4. *Wielemaker,J.*, *Beek,W.*, *Hildebrand,M.*, *Ossenbruggen,J.* ClioPatria: A SWI-Prolog Infrastructure for the Semantic Web, Semantic Web. vol. 7, no. 5, 2016, pp. 529-541. DOI:`10.3233/SW-150191`
5. Modelio Open Source – UML and BPMN free modeling tool. URL:`https://www.modelio.org/`.
6. *Raslan, W.*, *Sameh, A.* Accelerating High-Level SysML and SystemC SoC Designs, URL:`https://www.design-reuse.com/articles/17562/high-level-sysml-systemc-soc-designs.html`
7. *Natale,M.*, *Perillo,D.*, *Chirico,F.*, *Sindico,A.*, *Sangiovanni-Vincentelli,A.* A Model-based approach for the synthesis of software to firmware adapters for use with automatically generated components, Software & Systems Modeling, February 2018, Volume 17, Issue 1, pp. 11-–33
8. *Boutekkouk,F.*, *Fartas,O.* Automatic generation of SysML diagrams from VHDL code, In Proceedings of the Symposium on Complex Systems and Intelligent Computing (CompSIC), Souk Ahras, Algeria, September 2015.
9. *Mischkalla,F.*, *He,D.*, *Mueller,W.*, *Azcarate,F.*, *Carballeda,M.* A Retargetable SysML-based Front-End for High-Level Synthesis, In: Proceedings of 2nd Workshop on Model Based Engineering for Embedded Systems Design (M-BED), Mrz. 2011
10. *Hauswald,T.* Automatic Code Synthesis of UML/SysML State Machines for airborne Applications. Bachelor Thesis. Hamburg Univeristy of Technology. August 15. 2016. 80 p.
11. *Friedenthal,S.*, *Moore,A.*, *Steiner,R.* OMG Systems Modeling Language (OMG SysML™) Tutorial. 2009. 132 p. `http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf`
12. *Cherkashin,E.*, *Larionov,A. et al.* Logical programming and data mining as engine for MDA model transformation implementation. Procs of 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). May 20–24, 2013, Opatija, Croatia, pp. 1029–1036.
13. *Johnston,W.M.*, *Hanna,J.R.P.*, *Millar,R.J.* Advances in Dataflow Programming Languages. ACM Computing Surveys. – 2004. – Vol. 36. – pp. 1–34. 2004. – Vol. 36. – P. 1–34.