Journal of Computer Science

Original Research Paper

Prototyping Rule-Based Expert Systems with the Aid of Model Transformations

[1,2]Alexander Yurievich Yurin, [1]Nikita Olegovich Dorodnykh, [1]Olga Anatolievna Nikolaychuk and [1]Maksim Andreevich Grishenko

[1]*Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS), Irkutsk, Russia*
[2]*Irkutsk National Research Technical University (IrNRTU), Irkutsk, Russia*

Article history
Received: 16-10-2017
Revised: 02-02-2018
Accepted: 2-04-2018

Corresponding Author:
Alexander Yurievich Yurin
Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS), Irkutsk, Russia
Email: iskander@icc.ru

Abstract: The problem of improving efficiency of intelligence systems engineering remains a relevant topic of scientific research. One of the trends in this area is the use of the principles of cognitive (visual) modelling and design as well as approaches based on generative programming and model transformations. This paper aims to describe the implementation and application of model transformations for prototyping rule-based knowledge bases and expert systems. The implementation proposed uses the main principles of the Model Driven Architecture (MDA) (e.g., model types and creation stages) and considers the features of developing intelligent systems. Therefore, the current research employs the following tools: Ontologies for the representation of the computation-independent model; the author's original notation, namely, the Rule Visual Modelling Language (RVML) to create the platform-independent and platform-specific models; the C Language Integrated Production System (CLIPS) and the Drools Rule Language (DRL) as the programming languages (as the platforms). The approach proposed targets non-programmers (domain experts and analytics) and makes the design process of rule-based expert systems and knowledge bases more efficient. The paper also presents a detailed description of the main elements of the approach including models, transformations and a specialised software (Personal Knowledge Base Designer).

Keywords: Model-Driven Engineering, Expert System, Rules, Ontology, Prototyping, Model Transformations

Introduction

The problem of improving efficiency of Knowledge Bases (KB) and expert systems engineering remains a challenging topic of scientific research and it can be addressed in different ways: By improving the existing approaches or creating a specialized software for automation of the development process (Jackson, 1998; Giarratano and Riley, 2004; Liebowitz, 1998; Luger, 2008; Sahin *et al*., 2012).

At the same time there exist several main trends to improve the efficiency.

Using the software for ontological and cognitive modeling, CASE-tools (Protégé, OntoStudio, IHMC CmapTools, XMind, FreeMind, TheBrain, IBM Rational Rose, StarUML and etc.), which create graphic models that correspond to the key software abstractions. However, most of these systems do not cover all the creation stages of KBs and ESs and do not provide the completeness of the development process: From the subject domain model to the program codes. In some cases, they can only help obtain graphic images of KB structures. Perhaps, only Protégé is capable of generating a limited set of KB elements, in particular, for CLIPS/COOL.

Using KBs editors and ESs shells (Expert System Designer, Expert System Creator, ARITY Expert Development Package, CxPERT, Exsys Developer, DDTRES and etc.), which are programmer-oriented and allow implementation of a formalized description of the domain concepts and KB structures in a certain Programming Language (PL), but have a low integration capacity with visual modeling systems and knowledge interpretation modules, in most cases supporting one specific PL.

© 2018 Alexander Yurievich Yurin, Nikita Olegovich Dorodnykh, Olga Anatolievna Nikolaychuk and Maksim Andreevich Grishenko. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

Using integrated frameworks and unified approaches that provide the coverage of all phases of the life cycle of knowledge-based systems and the integration of the first two trends.

It should be noted that this area offers such solutions as AT-TECHNOLOGY (Rybina *et al.*, 2016) and such special methodologies as HeKatE (Nalepa and Ligęza, 2010) and CommonKADS (Schreiber *et al.*, 2000), however, there is a general tendency to target non-programmers (Nofal and Fouad, 2015; Ruiz-Mezcua *et al.*, 2011) and employ conceptual models, including ontologies and semantic nets (Baumeister and Striffler, 2015; Corsar and Sleeman, 2008; Nofal and Fouad, 2014; Rajput *et al.*, 2014; Shue *et al.*, 2009; Zagorulko and Zagorulko, 2013), when creating KBs.

At the same time, it remains relevant to expand the set of conceptual models used and further minimize the participation of the programmer in the creation of ESs and KBs.

One of the trend in these areas is the use of the principles of cognitive (visual) modelling and design as well as approaches based on generative programming (Czarnecki and Eisenecker, 2000; Czarnecki and Helsen, 2006), in particular, the Model-Driven Engineering (MDE) or the Model Driven Software Development (MDD) and its variants (modifications).

The MDE/MDD is a software design approach that uses the information models as the major artifacts, which, in turn, can be used for obtaining other models and generating programming codes (Sami *et al.*, 2005). This approach enables programmers and non-programmers (depending on the implementation) to create software on the basis of conceptual models.

Thus, the core ideas of the model-driven approach are:

- A model is a key artifact during the development process of software (a formal specification of the function, structure and behavior of a system within a given context)
- The software development process is a sequence (a chain) of transformations of models (from more abstract to less abstract)

To date, the best-known MDE initiatives are the following:

- The Model-Driven Architecture (MDA), which is a registered trademark of the Object Management Group (OMG) (Sami *et al.*, 2005; Djurić *et al.*, 2005; Frankel, 2003; Kleppe *et al.*, 2003; MDA, 2017; Schmidt, 2006). The main idea of the approach is to build an abstract meta-model for the management and exchange of metadata (models) and set the ways of their transformation into a software-supported technology (Java, CORBA, XML, etc.). MDA specifies three default viewpoints on software: Computation independent, platform independent and platform specific. The viewpoint is an abstraction technique for focusing on a particular set of concerns within a system while suppressing all irrelevant details. The viewpoint can be represented via one or more models
- The Eclipse Modeling Framework (EMF) is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model (EMF, 2017). The EMF provides the foundation for interoperability with other EMF-based tools and applications. The heart of EMF is Ecore. Ecore is a special language for description of meta-models (implementation of OMG's Essential Meta-Object Facility, EMOF). The basic tools to work with meta-models and skeletal code generation of software (programming skeletons) are EMF.Core, EMF.Edit, EMF.Codegen
- The Model-Integrated Computing (MIC) has been developed for over two decades at ISIS, Vanderbilt University, for building a wide range of software systems. MIC focuses on the formal representation, composition, analysis and manipulation of models during the design process. It places some models in the center of the entire system life-cycle, including specification, design, development, verification, integration and maintenance (MIC, 2017). MIC provides three core elements: The technology for the specification and use of the Domain-Specific Modeling Languages (DSML); the fully integrated metaprogrammable MIC tool suite and an open integration framework to support formal analysis tools, verification techniques and model transformations in the development process; the three-level representation of the system development process (Application Level, Model-Integrated Program Synthesis Level, Meta-Level)

In the context of the development of rule-based ESs we choose the MDA as the primary approach. This is the most standardized version (initiative) of the MDE, which uses the UML, one of the most common modeling and software design languages.

There can be found examples of successful use of the MDE approach in the development of database applications (e.g., ECO, for Enterprise Core Objects), agent-oriented monitoring applications (Gascueña *et al.*, 2012; 2014), decision support systems (Baumeister and Striffler, 2015; Shue *et al.*, 2009; Neto *et al.*, 2017), embedded systems (software components) for the Internet (Canadas *et al.*, 2009; Cabello *et al.*, 2009; Distante *et al.*, 2007), including rule-based ESs (Nofal and Fouad, 2014; Shue *et al.*, 2009; Nofal and Fouad, 2015; Ruiz-Mezcua *et al.*, 2011; Canadas *et al.*, 2009).

This paper aims to describe the implementation and application of the MDA/MDE approach and model transformations in prototyping rule-based KBs and ESs. The implementation proposed uses the main principles of the MDA/MDE (e.g., model types and creation stages) and considers the features of developing intelligent systems, in particular, rule-based ESs and KBs through specialization and redefinition of certain models and stages.

In particular, we suggest using the following tools:

- Ontologies and conceptual models in the form of UML class diagrams (Star UML and IBM Rational Rose formats) or mind maps (IHMC CmapTools format) to represent a computation-independent model (CIM), that distinguishes this work from similar ones, in particular (Nofal and Fouad, 2014; Shue *et al*., 2009; Nofal and Fouad, 2015; Ruiz-Mezcua *et al*., 2011; Canadas *et al*., 2009)
- The original author's notation - a Rule Visual Modelling Language (RVML) to improve the visibility of representations of cause-effect relations for designing platform-independent and platform-specific models, that allows us to take into account the specifics of the logical rules formalism, in contrast to (Canadas *et al*., 2009; Cabello *et al*., 2009)
- C Language Integration Production System (CLIPS) as a platform model

We also define the rules of model transformation in accordance with the principles of the MDA/MDE in the context of designing KBs and ESs. The closest works (Canadas *et al*., 2009; Cabello *et al*., 2009) use the classical MDE-based scheme for developing applications without taking into account the features of the development of knowledge-based systems, in particular, the need for a conceptualization stage or the selection of certain formalism for the knowledge representation.

The approach proposed is implemented in the form of a prototype of the specialized software: Personal Knowledge Base Designer (PKBD, 2017). This software supports the main stages of the development process and it was used in the ESs design to define the causes of damage and destruction of construction materials. This software was also used in the educational process at the Irkutsk National Research Technical University (IrNRTU).

The approach targets non-programmers (domain experts and analytics) and improves the efficiency of the design process of rule-based KBs and ESs.

This paper is organized as follows. The next section presents analysis of the related works; Section 3 contains the problem statement. Section 4 describes the modification and application of the MDA/MDE approach to the automated creation of rule-based KBs and ESs. Sections 5 and 6 demonstrate the implementation and applicability of the approach using an example. Discussion is presented in Section 7 and the concluding remarks are stated in Conclusion.

Related Works

The following several works have been already developed in the area of engineering rule-based KBs and ESs and exploit the principles of the MDA/MDE to a greater or lesser extent. Analysis of those works showed that they can be divided into two groups:

1. The first group includes the works that do not explicitly indicate the use of the MDE methodology and principles, however, they actually use conceptual models to describe the subject domain and some transformations of these models to interpret or generate software codes. In this case, the specialized transformation languages are not used and the main results are presented either in the form of single domain specific applications or problem-oriented shells.

In particular, Dunstan (2008) presents a method to automatically generate web-based ESs from XML descriptions of the knowledge domain. The case study is university course rules. An XML data definition file is developed featuring common rules and restrictions regarding courses. In this work the generator conception is used and the method is programmer-oriented.

Nofal and Fouad (2014) describe a tool for developing web-based ESs. The tool utilizes the Semantic Web technology which enables the knowledge engineers and domain experts to define knowledge without having to know anything about programming languages and Artificial Intelligence (AI). The facts of the knowledge can be annotated using the semantic concepts and relations found in WordNet ontology and interpreted in the tool.

Shue *et al*. (2009) use the ontology to model the domain knowledge and decision rules to represent operational knowledge. The system described integrates Protege, as a domain KB and the Java Expert System Shell (JESS), as an operational KB, into one complete ES. The case study is corporate financial rating. In this work the generator conception is implemented, so the main results are the JESS and Java program codes. The special transformation languages are not used.

Ruiz-Mezcua *et al*. (2011) created an ES development tool for non AI experts. This tool proposed allows development of the ESs on the basis of knowledge representation models. The models are described in the form of trees and interpreted in the tool. The special transformation languages are not used.

An approach for the ESs construction on the basis of UML is described in (Touzi and Messaoud, 2009). The approach proposed uses the extension of the CLIPS, called VCLIPS_UML, which is developed in Java and allows one to automatically generate the corresponding scripts in accordance with the CLIPS language. This approach is oriented to non-programmers.

Kadhim *et al.* (2013) introduced a tool for constructing rule-based ESs called Diagnosis Domain Tool for Rule-based Expert System (DDTRES), which tool provides a variety of functions to facilitate the development of ESs for practical problems in different diagnosis domains. This system is developed and implemented using the visual PROLOG programming language. Since the tool proposed is a problem-oriented shell, the structure of the domain model is already defined and is filled with the use of data mining methods.

2. The second group contains the works that explicitly use the MDE principles in the context of the EMF or MDA initiatives.

For instance, Canadas *et al.* (2009) present an MDE for the development of rule-based applications for the Web. Their approach uses ontology to describe the subject domain. In this case, the Conceptual Modeling Language (CML) (instead of UML) is used to describe the ontology (as the Computation-Independent (CIM) and Platform Independent Models (PIM)) as a rule modeling formalism. The implementation was made as a part of the Eclipse Modeling Project (using the Eclipse Modeling Framework); accordingly, model transformations (model-to-model and model-to-code) are described with the aid of the ATLAS Transformation Language (ATL).

The CIM and PIM are not separated and the selection of a possible formalism for knowledge representation is not offered. Ontology and rules are transformed into JESS, which supports the development and deployment of rule-based systems tightly coupled to Java applications. Furthermore, a Web-based architecture is generated from the CML model, enhanced by the interaction and presentation features.

The Web application code is based on the MVC architectural pattern and the Java Server Faces (JSF) framework, producing a set of JavaBeans classes and Java Server Pages (JSP). Further use of the results assumes that the user has programming skills.

Chaur (2004) offers an approach to create rule-based systems based on the concept of the EMF. In particular, the ECore meta-metamodel is used to describe a Rule Meta-Model which defines a conceptual model for representation of domain expert's knowledge in the form of JESS rules, but a clear description of CIM, PIM and a Platform-Specific Model (PSM) is lacking. In general, the principle of the generator for the JESS platform (oriented to programmers) is implemented.

Cabello *et al.* (2009) suggest an implementation of the MDA for the PRISMA platform. This tool allows generation of diagnostic ESs (as PRISMA architectural models) on the basis of conceptual models describing various aspects of software: The feature model, the decision tree, the domain conceptual model, the application domain conceptual model and etc. In this case, the first two models are considered as CIM and the rest as PIM. No specialized languages are used to implement the model transformations. The main results are the generated program codes for C# and .NET.

The model transformation is one of the main principles of the MDE/MDE approach and can be considered from different points of view. In particular, (Kleppe *et al.*, 2003; Czarnecki and Helsen, 2006) identified two types of transformations:

- Model-to-Model (M2M)
- Model-to-Text (M2T) and Text-to-Model (T2M)

At the same time, the M2T transformation corresponds to the concept of 'pretty printing' in the program transformation and the Model-to-Code (M2C) can be considered as a special case of M2T.

Two types of transformations are identified in (Mens and Gorp, 2006) in accordance with the modeling languages used to describe the source and target models:

- The endogenous transformation is a transformation between models that uses one modeling language
- The exogenous transformation is a transformation between models that uses different modeling languages

The model transformations can also be classified by the transformation direction (Mens and Gorp, 2006):

- A vertical transformation is a transformation where the source and target models reside at different abstraction levels
- A horizontal transformation is a transformation where the source and target models reside at the same abstraction level

The transformations should satisfy the following main requirements (Czarnecki and Helsen, 2006; Gardner *et al.*, 2003; Sendall and Kozaczynski, 2003):

- Completeness: It should allow one to represent any necessary transformation in accordance with the defined models
- Formality: It should allow automatic execution
- Flexibility: It should not depend on a specific subject domain

At present, there exist several research areas related to the implementation of model transformations:

- Using graph grammars (graph rewriting) (Rozenberg, 1999) (e.g., VIsual Automated model TRAnsformations (VIATRA2) (Varro and Balogh, 2007), Graph REwriting And Transformation (GReAT) (Balasubramanian *et al.*, 2007), Henshin (Arendt *et al.*, 2010), etc.)

- Using special languages and standards of model transformation (e.g., Query/View/Transformation (QVT) (QVT, 2017), ATL (Jouault *et al*., 2008), Epsilon (2017), etc.)
- Using declarative and procedural programming languages (Berman *et al*., 2010)
- Using languages for transforming XML documents (e.g., eXtensible Stylesheet Language Transformations (XSLT) (XSLT, 2017), etc.)

In the context of the MDA approach, it is recommended to use the standard for the M2M transformation called the QVT (Operational, Relational and Core languages). However, a significant drawback of QVT (like all model transformation languages) is the high qualification requirements for a user (developer). In particular, the user should:

- Know the syntax of the specific model transformation language
- Be able to describe transformation rules with the aid of the transformation language
- Know the meta-modelling languages (e.g., MOF, Ecore, KM3, etc.) to define the source and target languages (to support the model transformation process)
- Know and be able to use other languages in addition to the main model transformation languages, for example, the Object Constraint Language (OCL)

Therefore, we decided to make an 'ad-hoc' solution and use a direct-manipulation approach (Czarnecki and Helsen, 2006) for description of transformations and a general-purpose programming language (Object Pascal) to provide the internal representation and transformations of the models in the prototype (framework) of the software tool that implements the approach proposed.

The Problem Statement

Analysis of the existing technologies for the development of ESs and KBs (Jackson, 1998; Giarratano and Riley, 2004; Liebowitz, 1998; Luger, 2008; Djurić *et al*., 2005; Canadas *et al*., 2009) revealed that they primarily target users with knowledge of software engineering.

Therefore, to apply these technologies, the developer should have programming skills and know at least one programming language for KBs. Similarly, if a programmer develops KBs, then he/she has to study the subject domain model and formalize basic concepts and relations. It is a rare occasion for the same person to have necessary programming skills and be an expert in the subject area.

As a result, we propose to adapt (modify) and apply the MDA approach to the automated creation of rule-based KBs and ESs, including automatic generation of program codes and specifications on the basis of the subject domain models represented in the form of graphics primitives. This type of automation would help minimize the participation of the programmer in the software development process.

We formalize an MDA as follows:

$$MDA = \left\langle \begin{array}{l} L, CIM, PIM, PSM, PDM, \\ F_{CIM-to-PIM}, F_{PIM-to-PSM}, \\ F_{PSM-to-CODE} \end{array} \right\rangle$$

where, L are visual modeling languages, $L=\{UML\}$; CIM, PIM, PSM, PDM are corresponding models; $F_{CIM\text{-}to\text{-}PIM}: CIM \rightarrow PIM$, $F_{PIM\text{-}to\text{-}PSM}: PIM \rightarrow PSM$, $F_{PSM\text{-}to\text{-}CODE}: PSM \rightarrow CODE$ are model transformation rules.

Thus, the main purpose of this paper is to adapt (to specialize) the MDA methodology in the context of the development of rule-based expert systems and knowledge bases, i.e. to define an MDE^{RB_ES}:

$$MDA^{RB\_ES} = \left\langle \begin{array}{l} L^{RB\_ES}, CIM^{RB\_ES}, PIM^{RB\_ES}, \\ PSM^{RB\_ES}, PDM^{RB\_ES}, \\ F^{RB\_ES}_{CIM-to-PIM}, F^{RB\_ES}_{PIM-to-PSM}, \\ F^{RB\_ES}_{PSM-to-CODE} \end{array} \right\rangle$$

Thus, it is necessary to define the elements of the MDA^{RB_ES}, including models and rules for the transformation of models in the context of designing rule-based ESs and KBs and implement them in the form of a special tool.

Prototyping Rule-Based Expert Systems with the Aid of Model Transformations

According to MDA (Sami *et al*., 2005; Frankel, 2003; Kleppe *et al*., 2003; MDA, 2017), the designed software, which includes a description of the basic concepts, the relations between them and methods for processing them, is represented in the form of information models defining the composition, structure and behaviour. Therefore, the process of software development is a gradual transition from abstract information models (i.e., models that do not contain the details of the implementation on a specific technological platform; such models are called logical) to specific information models (i.e., models that contain the details of the implementation on a specific technological platform; such models are called physical) with the subsequent generation (synthesis) of the program codes of a KB and an ES.

Main Stages

The development process of KBs and ESs is presented by a sequence of stages providing the creation and transformation of information models.

Stage 1: Building a Model of a Subject Domain

The main concepts and relations. At this stage, the user creates a Computation-Independent Model (CIM). This model can be implemented in the form of ontology or an UML-model (in particular, as a class diagram). In addition to the concepts and relations of 'is-part-of' and 'is-a', the relation 'depends-on' is introduced; this relation provides a description of cause-and-effect relations. In the case of UML class diagrams, these types of relations are defined by the mechanism of stereotypes. A description of the main architectural elements of the ES (such as the 'input form'; the 'output form' etc., which are derived from the 'border class'; the 'inference engine', which is derived from the 'control'; the 'knowledge-base') is also produced at this stage.

The efficiency of this stage can be improved by reusing the existing conceptual models created using various ontological and cognitive editors, such as CASE-tools (e.g., Protégé, CmapTools, IBM Rational Rose Enterprise) (Dorodnykh and Yurin, 2015). Most of the software that supports the MDA approach (e.g., Bold for Delphi) does not realize this stage and only enables development of the software starting at the next stage. In this case, the conceptual model of a subject domain (even presented in the form of ontology (Djurić *et al*., 2005)) is considered as a Platform-Independent Model (PIM) that describes the main concepts and business logic (that is acceptable for databases). In the case of developing intelligent systems, this stage is necessary and corresponds to the stage of the conceptualization of knowledge. This stage allows transition from a general conceptual model of a subject domain to a knowledge representation model (with logical rules).

Stage 2: Building Platform-Independent Models (PIMs)

The models of this stage describe logical rules that stem from the automated transformation of a CIM with the subsequent specification of the results of the automated transformation. In the process of the CIM transformation, the concepts are transformed into the fact templates and rule elements (such as the conditions and actions) and the cause-and-effect relations are transformed into logical rules. In fact, the automated formalization of a subject domain model is carried out.

Visual modelling is one of the main aspects of the MDA approach. MDA traditionally uses a Unified Modelling Language (UML) for building models. It should be noted that applying the MDA approach to develop specific software requires the use of UML extensions (Miguel *et al*., 2002) that allow one to take into consideration some features of a subject domain (e.g., telecommunication or health.), architectures (e.g., real-time access and reliability) and programming languages and formalisms (e.g., Prolog). Because a UML is not intended for illustrative and unambiguous representation of cause-and-effect relations (logical rules), we use the author's original notation of the 'Rule Visual Modelling Language' (RVML) to represent logical rules) and furthermore, to serve as a UML.

Designing an ES during the development of a KB involves the design of the ES's structure and the user interface. Thus, elements of the approach known as 'Ontology Driven Architecture' (ODA, the section within MDA) are used (Djurić *et al*., 2005; Gašević *et al*., 2009). This approach is intended to develop the theory and tools for building software based on the ontological transformation. Hence, after the creation of rules the user is prompted to choose the 'initial' rule, which helps construct the chains of the logical inference. The analysis of the obtained chains allows one to design the architecture of an ES (i.e., a set of software components that provide the input, output and processing of information). The architecture is presented in the form of a UML class diagram.

Stage 3: Building Platform-Specific Models (PSMs)

That take into account the features of a certain knowledge representation language (e.g., CLIPS), such as priorities of rules and 'by default' values of slots.

Stage 4: Generating the Code of a KB and an ES

At this stage, the interpretation of the UML-class diagram (that describes the software architecture) and RVML diagrams is performed. The main results of the interpretation are the program codes and specifications for an interpreter. In the process of interpretation and code generation, the Platform Description Model (PDM) and rules for the transformation of models are used. In this case, a PDM describes the syntax and semantics of the programming languages for which program code is generated.

Stage 5: Testing

At this stage, the program codes obtained are tested in a special software (in the interpreter).

It should be noted that the end user (an expert or a system analytic) only designs a CIM, a PIM and part of a PSM. All the transformations of the models and the generation of program codes (with the possibility of modifications) are implemented with a specialized software that includes a PDM.

The described sequence of stages almost coincides with a 'standard' MDA approach, but the stage content is redefined on the basis of designs of the rule-based ESs and KBs.

Next, we focus on the models under study.

Models

The description of the models and their transformations are important for the MDA/MDE approach.

The Computation-Independent Model

The computation-independent model (CIM) can be presented in the form of ontology (Fig. 1) that includes the subject domain ontology (e.g., the reliability of technical systems) and the ontology of rule-based ESs, which includes the description of the main architectural elements that are necessary for the implementation of the approach proposed. In turn, the subject domain ontology includes the concepts (i.e., the classes and instances) and the relations between them including the basic data types; the classes and properties.

The Platform-Independent and Platform-Specific Models.

A PIM is described with two models and can be represented as follows:

$$PIM^{RB\_ES} = \left\langle UML^{RB\_KB}, UML^{RB\_ES} \right\rangle$$

where, UML^{RB_KB} is a model of a KB and UML^{RB_ES} is a model of an ES architecture.

Representation and Modelling of the ES Architecture

The UML class diagrams with additional classes (e.g., 'Input From Class', 'Output From Class') that extend the standard classes of 'Border Class', 'Entity' and 'Control' are used for the representation and modelling of the ES architecture.

The following equations give the main concepts of UML^{RB_ES}:

<UML_RB_ES> = <Class>+

<Class> = <Border Class> | <Entity> | <Control>

<Border Class> = <Input Form Class> | <Output Form Class>

<Input Form Class> = Facts input form

<Output Form Class> = Results form | Explanation form

<Control> = DROOLS interpreter| CLIPS interpreter.

Representation and Modelling of Logical Rules

The RVML-notation (RVML, 2017) (which is based on the UML) is used for the platform-independent modelling of logical rules (Fig. 2). This notation allows description of the cause-and-effect relations and abstraction from the features of programming languages for rule-based KBs.

In addition, the specification of certain elements of the notation (such as the Priority (P) and the Certainty Factor (CF)) provides the means to create a PSM, especially for a CLIPS.

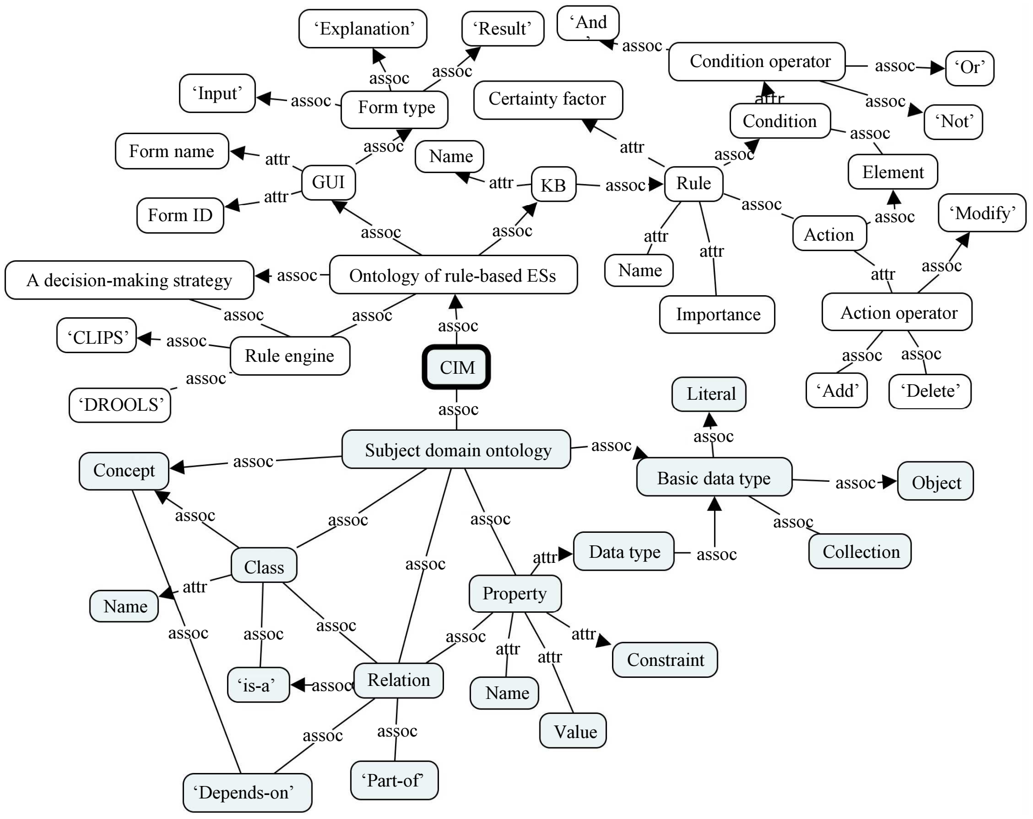


Fig. 1: The structure of a computation-independent model for prototyping rule-based expert systems

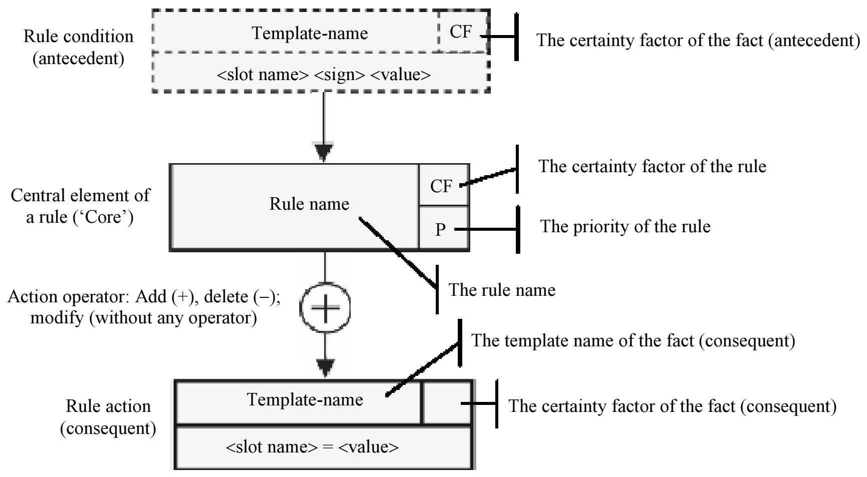

Rule condition (antecedent)
Template-name
CF
<slot name> <sign> <value>
The certainty factor of the fact (antecedent)
Central element of a rule ('Core')
Rule name
CF
P
The certainty factor of the rule
The priority of the rule
The rule name
Action operator: Add (+), delete (−); modify (without any operator)
The template name of the fact (consequent)
Rule action (consequent)
Template-name
<slot name> = <value>
The certainty factor of the fact (consequent)

Fig. 2: An example of the basic elements of the RVML

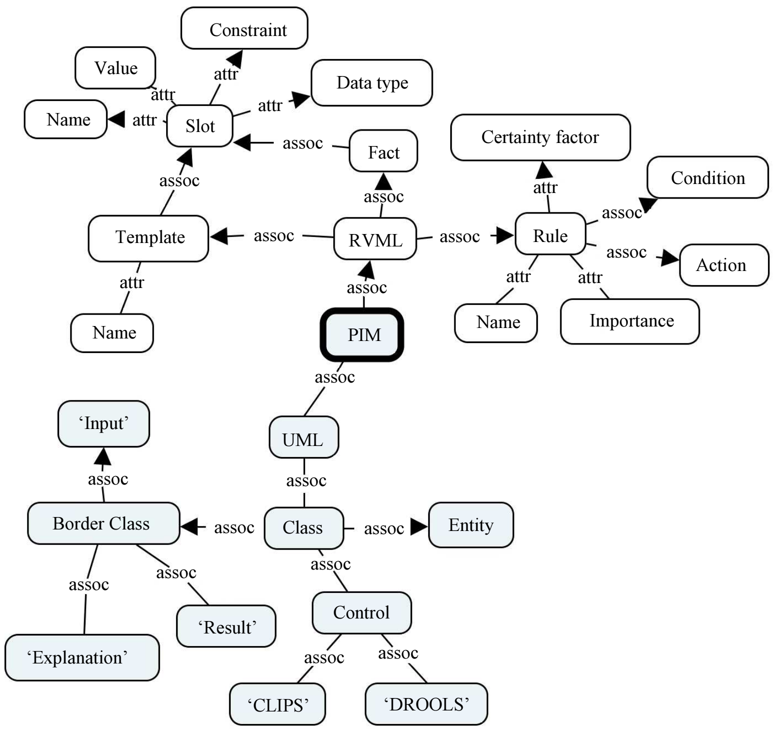


Fig. 3: The structure of a platform-independent model for prototyping rule-based expert systems

The structure of a PIM in the form of a concept card is presented in Fig. 3.

The Platform Model

The C Language Integrated Production System (CLIPS) was selected as a platform model. The following equations give the main elements of the CLIPS in the EBNF):

<CLIPS> = <template>*, <fact>*, <rule>*.

<template> = (deftemplate <template-name> [<optional-comments>] [<slot-definition>*]).

<slot-definition> = <simple-slot-definition> | <composite-slot-definition>.

<simple-slot-definition> = (slot <slot-name> <template-attributes>).

<template-attributes> = <attribute-default-value> | <restriction-attribute>

<restriction-attribute> = <type-attribute>

<type-attribute> = (type <type-specification>)

<type-specification> = float | integer | symbol | string | external-address | fact-address | instance-name | instance-address

<fact> = (deffacts <facts-list-name> [<optional-comments>] [<fact>*])

<rule> = (defrule <rule-name> <comment>] [<rule-property- definition>]

<antecedent>; rule LHS

=>

<consequent>; rule RHS).

Transformation of the Models

Thus, it is necessary to implement a sequence of exogenous vertical transformations:

- The M2M-transformation for $F^{RB\_ES}_{CIM-to-PIM}$
- The M2M-transformation for $F^{RB\_ES}_{PIM-to-PSM}$
- The M2C-transformation for $F^{RB\_ES}_{PSM-to-CODE}$

The following is a fragment of the transformation rules in the EBNF:

<Transformation> = <Transformation rule> {<Transformation rule>}.

<Transformation rule> = Rule <name> {<Source model element>, <Result>}.

<Source model element> = <ONT_D element> | <ONT_RB_ES element> | <UML_RB_ES element> | <UML_RB_KB element>.

<Result> = <UML_RB_ES element>|<UML_RB_KB element>|<CLIPS element>|<GUI element>.

where <ONT_D element>, <ONT_RB_ES element>, <UML_RB_ES element>, <UML_RB_KB element>, <CLIPS element>, <GUI element> are the elements of the domain ontology, the ontology of rule-based ESs, the UML-models of rule-based ESs, the rules (RVML) (Miguel *et al*., 2002), the CLIPS-models and the models of a Graphic User Interface (GUI), respectively.

Then, we use the following transformation rule templates:

$F^{RB\_ES}_{CIM-to-PIM}$ = {**Rule** ONT_D-TO-UML_RB_KB; **Rule** ONT_RB_ES-TO-UML_RB_ES},

$F^{RB\_ES}_{PIM-to-PSM}$ = {**Rule** UML_RB_KB-TO-UML_RB_KB*}, $F^{RB\_ES}_{PSM-to-CODE}$ = {**Rule** UML_RB_KB*-TO-CLIPS; **Rule** UML_RB_ES-TO-GUI}:

Rule ONT_D-TO-UML_RB_KB{<ONT_D element>, <UML_RB_KB element>}.

Rule ONT_RB_ES-TO-UML_RB_ES {<ONT_RB_ES element>, <UML_RB_ES element>}.

Rule UML_RB_KB*-TO-CLIPS {<UML_RB_KB* element>, <CLIPS element>}.

Rule UML_RB_ES-TO-GUI{<UML_RB_ES element>, <GUI element>}.

These are the transformation rules we employ:

- ONT_D-TO-UML_RB_KB:

Rule Class-Template {<Class>, <Template_UML_RB_KB>}.

Rule Object-Fact {<Object>, <Fact_UML_RB_KB>}.

Rule is-a-Slot {<Relation>:<Relation type>:<is-a>, <Slot>}.

Rule Class-Property-Slot {<Relation>:<Relation type>:<is-part-of>, <Slot>}.

Rule Cause-Rule {<Relation>:<Relation type>:<depends-on>, <Rule_UML_RB_KB>}.

Rule Property-Slot {<Property>, <Slot>}.

Rule Value {<Property value>, <Slot value>}.

- ONT_RB_ES-TO-UML_RB_ES:

Rule Form-UML {<Form>, <Border Class>}.

Rule Knowledge_base-UML_RB_ES{<Knowledge base>, <Entity>}.

Rule Interpreter-UML_RB_ES{<Interpreter>, <Control>}.

- UML_RB_KB*-TO-CLIPS:

Rule Template-CLIPS {<Template_UML_RB_KB*>, <template>}.

Rule Fact-CLIPS {<Fact_UML_RB_KB*>, <fact>}.

Rule Rule-CLIPS {<Rule_UML_RB_KB*>, <rule>}.

Rule Slot-CLIPS {<Slot_UML_RB_KB*>, <slot-value>}.

Rule Value-CLIPS {<Slot_value_UML_RB_KB*>}.

- UML_RB_ES-TO-GUI:

Rule Border_Class-GUI {<Border Class>, <GUI Border Class>}.

Rule Entity-GUI {<Entity>, <GUI Entity>}.

Rule Entity-GUI {<Control>, <GUI Control>},

where <GUI Border Class>, <GUI Entity>, <GUI Control> are the elements of the user interface for classes with the corresponding stereotypes.

The elements of the models can be represented in a (Table 1).

The model transformation rules are implemented with an imperative programming language.

Table 1: A fragment of a table of mapping of models' elements (CIM to PIM and PIM to CLIPS and DRL)

CIM elements	PIM elements	CLIPS elements	DRL elements
Class (*name*, *description*)	TemplateFact (*name*, *description*)	(**deftemplate** *name* "*description*" …)	**declare** name … **end**
Property	Slot (*description*, *value*)	(**slot** *name*)	*name*
Property value	Slot value	(**default** *value*)	
Property type	Slot type	(**type** *datatype*)	: datatype
Relationship	Rule (nodal element)	(**defrule** *name* (…) => (…))	**rule** "name" salience 0 **when** … **then** … **end**

Implementation

The approach proposed was implemented in the form of a research prototype of software known as the Personal Knowledge Base Designer (PKBD, 2017).

This software includes main modules with the following purposes:

- Designing and modelling; these modules are intended to create an application model and include a model editor, a model checker and modules for importing and exporting models
- Code generation; this module generates the program codes, including specifications for the interpreter and CLIPS codes
- Interpretation and control; these modules provide execution, including the interpretation and access to model elements and the interaction between the data level and the graphical user interface

The software provides full support for stages 1 and 2 of the proposed approach and partial support for stages 3-5.

Consider an example of application of the proposed approach for the development of a KB and an ES for the definition of the causes of damage and destruction of construction materials in petrochemistry.

The main cause of damage and destruction of a construction is degradation processes. The processes of degradation (Berman and Nikolaichuk, 2007) are the objective physical-chemical processes conditioned by both different technological processes and structural, manufacturing and maintaining irregularities which cause damages and destruction of materials and parts, failures of mechanical and petrochemical systems (or apparatus) and emergencies. Each process of degradation is characterized by a mechanism and kinetics. A mechanism of degradation is a set of properties of the technological object and effecting factors. A kinetics is a set of micro-and (or) macroscopic phenomena resulting from accumulating elementary movement acts. A description of kinetics includes: Events; event parameters; functional relations (if it is possible) for the definition of event parameters at a specified moment of time. In the field of safety the degradation processes are called hazardous processes.

According to the approach proposed, in order to develop a KB and an ES for definition of the causes of damage and destruction of construction materials, it is necessary to design conceptual models of the basic concepts of the subject domain that will be a CIM:

Stage 1

Building a CIM using a model of the dynamics of technical states (Berman and Nikolaichuk, 2007) by the experts. The main result of this stage is the subject domain concepts and relations.

In particular, a fragment of a CIM (Fig. 4) describes a mechanism of a degradation process (exist-mech), events (exist-event), a construction material (material), a technological environment, etc. At the same time, the «association» relation denotes the existence of relations between concepts that can be interpreted as cause-and-effect relationships.

Stages 2 and 3

Building a PIM includes the definition of cause-and-effect relationships in the form of logical rules. In addition, the ES architecture is created on the basis of the transformation of a CIM. Depending on the element type of the CIM, the concepts are transformed to templates for facts and rules (Fig. 5) for further modifications by the user with consequent destining specific rules (Fig. 6).

In addition to the PIM for a KB, a PIM for the ES is formed. The PIM for the ES includes a description of the main GUI forms and this model is created on the basis of a rule chain analysis. In particular, the following rules (forming the PIM for a KB) were obtained:

1. **IF** the water is under pressure of 3-4 MPa **AND** the temperature is approximately 300°C **AND** there are chlorine ions **AND** there is dissolved oxygen **THEN** the technological environment is active
2. **IF** we have low-alloy steel **THEN** the construction material is sensitive
3. **IF** the construction material is sensitive **AND** the technological environment is active with properties

alternation **AND** incident object is a pipe into pipe **AND** a constant mechanical stress with high cycle frequency exists **THEN** the event is corrosion **AND** the mechanism is local corrosion (Cf = 0,6) **AND** the mechanism is corrosion cracking (Cf = 0,9)

4. … .

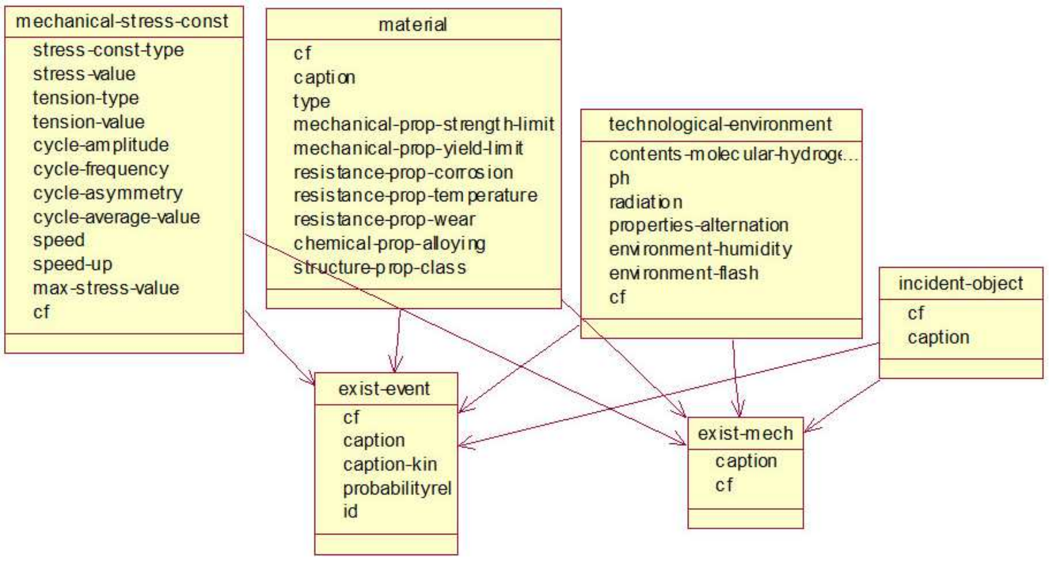


Fig. 4: A fragment of a CIM in the form of a UML class diagram (IBM Rational Rose)

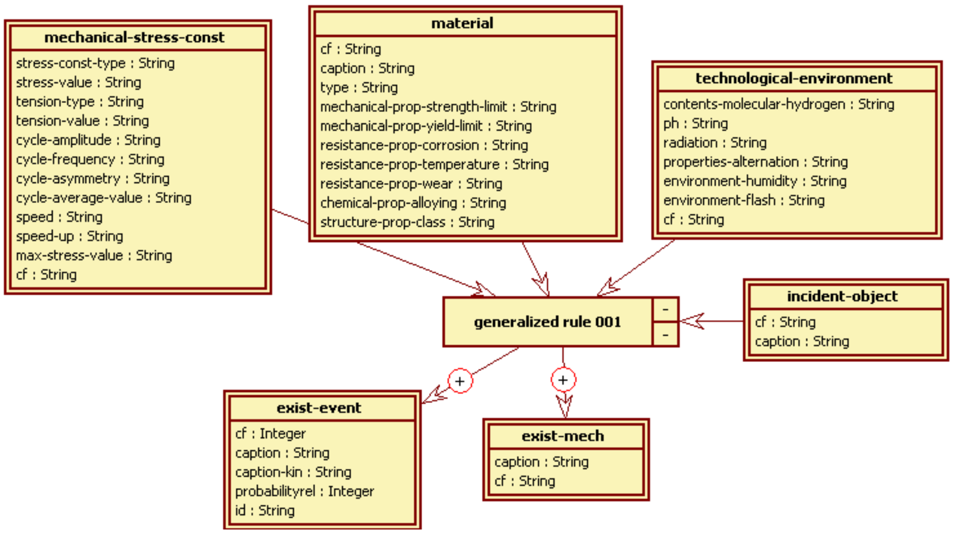


Fig. 5: An example of a template for a rule (RVML) (Dorodnykh and Yurin, 2015)

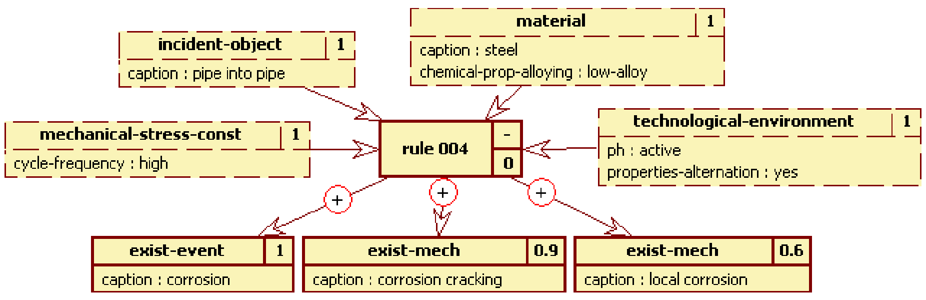


Fig. 6: An example of a specific rule (RVML) (Dorodnykh and Yurin, 2015)

These rules form a chain of logical inference that does not require any additional information (i.e., this chain uses the contents of the working memory and the initial data entered by the user). The PIM for the ES (the architecture) corresponding to this chain includes the following software components:

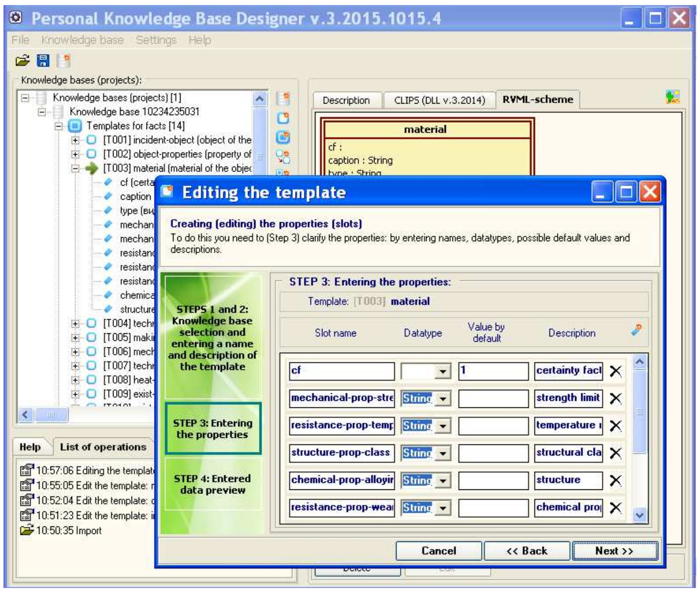
- The initial data input form that provides the input information on the technological environment, the stresses, the investigated object and the material
- The inference machine (this element is a part of the ES and is hidden from the user)
- The output form that provides the publication of the results of the logical inference
- The output form that provides an interpretation of the results obtained (it shows activated rules and modified, added and deleted facts)

Stage 4

Generating a code and specifications, including:

- The CLIPS code

Specifications of the ES for the interpreter, which provides the generation of the user interface for the creation, reading, updating and deleting (CRUD) of the KB elements (Fig. 7) and the interaction between software components.


Personal Knowledge Base Designer v.3.2015.1015.4

File Knowledge base Settings Help

Knowledge bases (projects):

- Knowledge bases (projects) [1]
  - Knowledge base 10234235031
    - Templates for facts [14]
      - [T001] incident-object (object of the
      - [T002] object-properties (property of
      - [T003] material (material of the objec
        - cf (certa
        - caption
        - type (ви
        - mechan
        - mechan
        - resistan
        - resistan
        - resistan
        - chemica
        - structure
      - [T004] techn
      - [T005] maki
      - [T006] mech
      - [T007] techn
      - [T008] heat-
      - [T009] exist-

Help | List of operations

- 10:57:06 Editing the templat
- 10:55:05 Edit the template:
- 10:52:04 Edit the template:
- 10:51:23 Edit the template:
- 10:50:35 Import

Description | CLIPS (DLL v.3.2014) | RVML-scheme

material
cf :
caption : String
type : String

Editing the template

Creating (editing) the properties (slots)
To do this you need to (Step 3) clarify the properties: by entering names, datatypes, possible default values and descriptions.

STEPS 1 and 2: Knowledge base selection and entering a name and description of the template

STEP 3: Entering the properties

STEP 4: Entered data preview

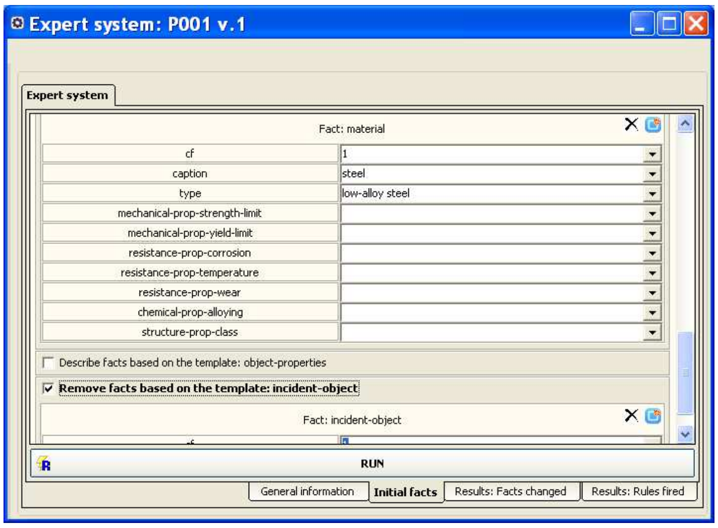
STEP 3: Entering the properties:

Template: [T003] material

Slot name	Datatype	Value by default	Description
cf		1	certainty fact
mechanical-prop-stre	String		strength limit
resistance-prop-temp	String		temperature i
structure-prop-class	String		structural cla
chemical-prop-alloyir	String		structure
resistance-prop-wear	String		chemical pro

Cancel | << Back | Next >>

Fig. 7: An example of a PKBD GUI (Dorodnykh and Yurin, 2015)


Expert system: P001 v.1

Expert system

Fact: material

cf	1
caption	steel
type	low-alloy steel
mechanical-prop-strength-limit	
mechanical-prop-yield-limit	
resistance-prop-corrosion	
resistance-prop-temperature	
resistance-prop-wear	
chemical-prop-alloying	
structure-prop-class	

☐ Describe facts based on the template: object-properties

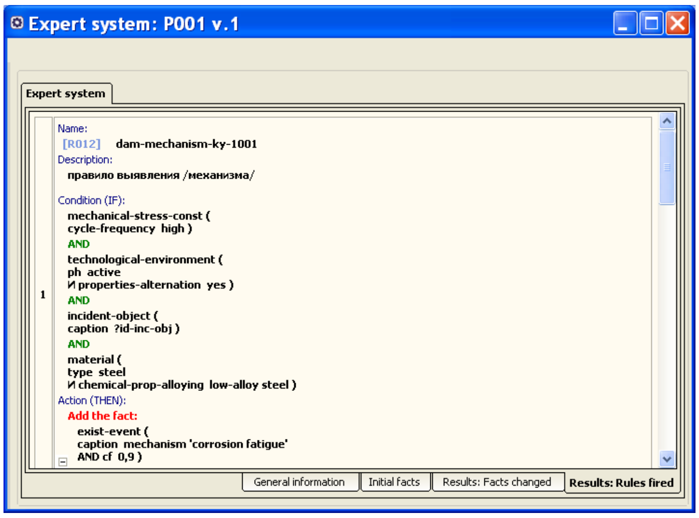
☑ **Remove facts based on the template: incident-object**

Fact: incident-object

RUN

General information | **Initial facts** | Results: Facts changed | Results: Rules fired

Fig. 8: An example of a GUI: Initial facts preview



```
1  Name:
    [R012]   dam-mechanism-ky-1001
   Description:
     правило выявления /механизма/

   Condition (IF):
     mechanical-stress-const (
     cycle-frequency  high )
     AND
     technological-environment (
     ph  active
     И properties-alternation  yes )
     AND
     incident-object (
     caption  ?id-inc-obj )
     AND
     material (
     type  steel
     И chemical-prop-alloying  low-alloy steel )
   Action (THEN):
     Add the fact:
       exist-event (
       caption  mechanism 'corrosion fatigue'
       AND cf  0,9 )
```

Fig. 9: An example of a GUI: Rules activated preview

Stage 5

Testing a KB and an ES is carried out by an expert by means of logical inferences (Fig. 8 and 9). It is possible to return to one of the previous stages in accordance with the results of the testing.

Efficacy Evaluation

The results obtained (the approach and the software) were verified on the basis of the Irkutsk National Research Technical University (IrNRTU). The case study involved 60 students, who graduated from the Popov Cybernetics Institute, where they completed training in CASE-tools, Means of Information Technologies and Programming Technologies. As a result, the students know the basic concepts of software design, UML, knowledge management and expert systems.

The main objectives of the case study were: (1) to assess the complexity of the development of knowledge bases of expert systems using our approach and the software developed (UML-modeling + PKBD). Denote this approach as A1; (2) to compare A1 with the complexity of the KBs development under different conditions:

- Without UML-modeling, but with the use of the software for knowledge base design, in particular, ClipsWin (2017) – a free and simple CLIPS editor for Windows (this is just a pure programmer's approach, denote this approach as A2)
- UML-modeling + other software for knowledge base design, in particular, ClipsWin (denote this approach as A3)
- IBM Rational Rose (2017) is chosen as a UML-modeling tool which is widely used when creating non-specialized software

There are 20 variants (tasks) for the design of static expert systems for solving problems of diagnosing or prognosis in different subject areas. Some constraints were imposed on the characteristics of subject area models and knowledge bases (on the tasks), in particular:

- The number of subject area entities: 5-10
- The number of properties of subject area entities: 3
- The number of connections between subject area entities: 5-10
- The number of cause-effect relations (generalized rules): 3-4
- The number of instances of cause-effect relations (possible concrete rules): 10-15

Using the constraints provides multiple repetitions of the tasks and their time compactness.

The time criterion is used (the time required to perform certain stages of development of expert systems) to assess the complexity.

The assessment was carried out in the following stages (Jackson, 1998):

- Conceptualization (including building the conceptual model)
- Formalization (including the transformation of the key concepts and relations to some formal knowledge representation language)
- Programming (including the transformation of formalized knowledge into a working program)

The main results of the conceptualization and formalization stages are the conceptual models of the subject areas presented in the form of UML class diagrams. The main results of the implementation stage are syntactically corrected program codes of the knowledge bases, checked to ensure their adequacy and consistency.

For each variant (task) three results describing the time used were obtained, the average of their values is presented in Table 2 and Fig. 10.

Table 2: The results of evaluation of the time used

Variant (Task)	UML-modeling with the aid of IBM Rationa Rose, min.	Designing knowledge base with the aid of PKBD, min.	A1: IBM rational rose + PKBD, min.	A2: ClipsWin, min.	A3: IBM rational rose + ClipsWin мин.	Relative difference, % A1 vs. A2	A1 vs. A3
1	10,89	7,2	18,09	41,29	30,4	40,49	56,19
2	8,36	7,1	15,46	32,86	24,5	36,89	52,95
3	8,58	8,3	16,88	36,46	27,88	39,45	53,70
4	9,36	5,83	15,19	26,82	17,46	13,00	43,36
5	11,25	5,52	16,77	64,41	53,16	68,45	**73,96**
6	10,78	4,6	15,38	43,8	33,02	53,42	64,89
7	6,6	15,82	22,42	68	61,4	63,48	67,03
8	10,95	7,56	18,51	57,23	46,28	60,00	67,66
9	7,37	7,2	14,57	54,71	47,34	**69,22**	73,37
10	12,58	6,6	19,18	42,7	30,12	36,32	55,08
11	8,69	5,5	14,19	38,01	29,32	51,60	62,67
12	8,36	6	14,36	45,22	36,86	61,04	68,24
13	10,64	7,42	18,06	44,31	33,67	46,36	59,24
14	10,66	10,23	20,89	50,57	39,91	47,66	58,69
15	10,01	7,56	17,57	55,5	45,49	61,38	68,34
16	10,92	8,96	19,88	49,42	38,5	48,36	59,77
17	8,14	8,36	16,5	45,59	37,45	55,94	63,81
18	11,55	18	29,55	47,43	35,88	17,64	37,70
19	11,85	5,2	17,05	43,28	31,43	45,75	60,61
20	9,12	7,44	16,56	40,95	31,83	47,97	59,56
The resulting average value of the relative difference:						**48,2**	**60,3**

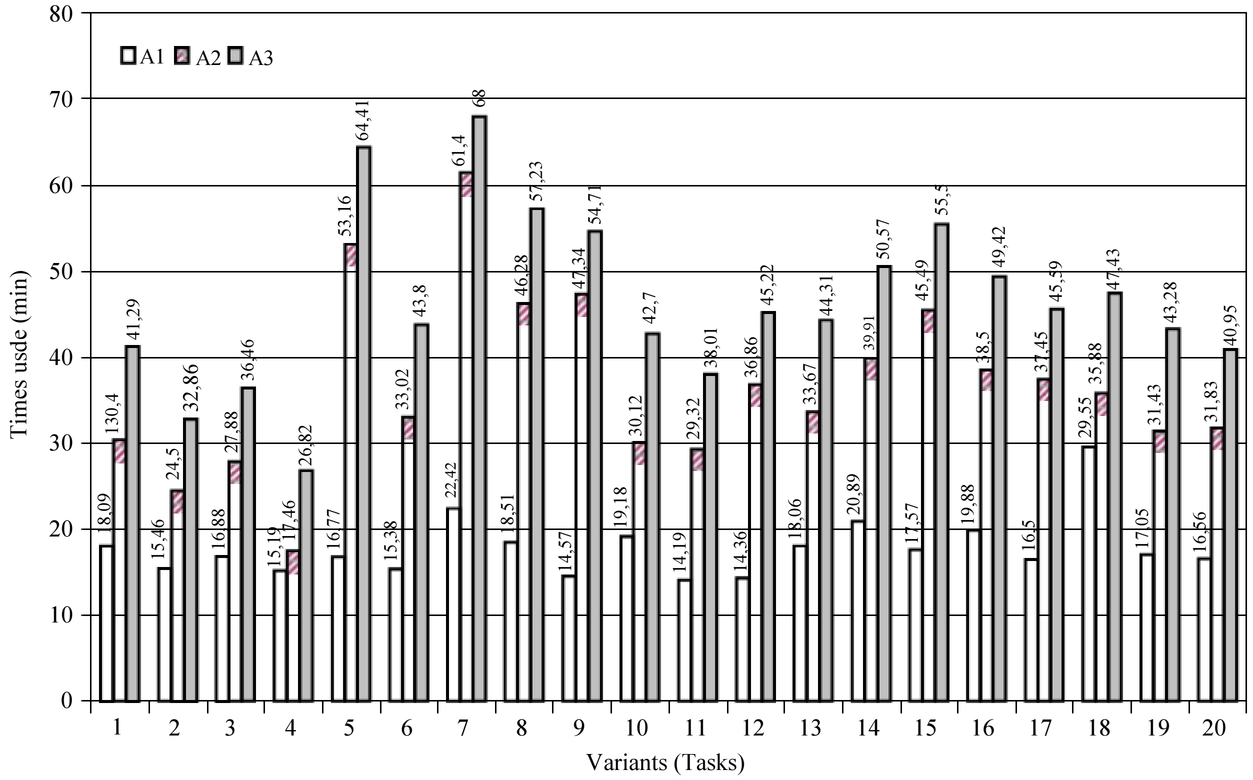


Fig. 10: The results of evaluation of time used

Let us highlight the features of performing the work at various stages for different approaches:

A2: The ClipsWin has functional limitations when it comes to manual editing of codes. This obstacle stipulated application of the additional text editor, Programmer's Notepad, at the implementation stage.

In particular, first, the description of the code in an external text editor (using the copying and pasting of individual blocks of a code) is carried out and then the resulting code is imported into ClipsWin, which carries out the syntax check. In practice, using this scheme, the creation of knowledge bases took one and a half time less.

A3: This approach provides the greatest time performance and uses IBM Rational Rose Enterprise for conceptual modeling of the subject area. The greatest time performance is caused by the manual transfer of the obtained conceptual models into (due to the absence of the function of automatic code generation for the knowledge base on the basis of conceptual models).

The analysis of the effectiveness of the approach proposed by the time criteria showed that the effectiveness of the development of knowledge bases by the A1 can be increased by 60.3% Vs. A3 and by 48.2% Vs. A2 on average due to the automatic code generation based on conceptual models, which in turn allows:

- Increase of the effectiveness of using the results of the conceptualization and formalization stages in the form of UML class diagrams, considering them not as static images, but as a basis for the automatic formation of the program codes in accordance with the ideology of a model-driven approach
- Reduction of the risk of design errors by enabling rapid prototyping knowledge bases and getting their program codes
- Elimination of programming errors (hand coding errors) by automatically transferring the elements of the conceptual models into CLIPS language constructs

Discussion

In accordance with the problem statement, we proposed an approach for developing ESs and KBs based on the MDA/MDE principles and redefined its main elements, in particular: Models (CIM, PIM, PSM, PDM) and the main stages. Methodologically, this approach is a combination of:

- The classical methodology for the development of ESs and KBs (Jackson, 1998; Giarratano and Riley, 2004; Liebowitz, 1998; Luger, 2008), that forms a chain of steps: Identification, conceptualization, formalization, implementation and testing
- The MDE-based methodology (Sami *et al.*, 2005; Djurić *et al.*, 2005; Frankel, 2003; Kleppe *et al.*, 2003; MDA, 2017; Schmidt, 2006), that forms a chain of steps: The creation of CIM, PIM, PSM and code generation (or model interpretation)

This combination is a qualitative difference between this work and similar ones (Table 3) and it allowed us to transfer the MDE principles in the field of knowledge engineering and will provide the use of the methodology proposed when creating other types of ESs, for example, case-based (by means of redefining PIM) or to extend the list of supported programming languages (by adding new PSMs).

In turn, the software (PKBD), which is implemented in the form of a shell, can be used by both programmers and non-programmers to create ESs and the generated codes of KBs can be used in other applications.

The main area of application of the proposed approach and software is the rapid prototyping of rule-based ESs and KBs on the basis of conceptual models.

The approach presented has some limitations: it helps create rule-based ESs and KBs only and does not enable the creation of ESs and KBs in the form of embedded components for intelligent systems. In future, we plan to eliminate these limitations:

- To provide the generation of program codes of embedded ESs for applications
- To extend support for programming languages and formats of conceptual models
- To specify the RVML in order to improve its expressiveness (for example, to define fuzziness, etc.) and ability to describe other knowledge representation formalisms
- To develop the web-oriented version of software (shell) to support the distributed user interaction through Internet when creating ESs and KBs

The results of testing of the approach and software proposed showed their high efficiency caused by the automatic code generation and the simplicity of the examples (tasks) with limitations. We expected that the superiority might be lost when solving more complex problems.

The main differences between the present work and those considered above are the following (Table 3):

- An explicit use of the MDE approach principles, including the identification of conceptual models of varying degrees of abstraction (CIM, PIM, PSM) and the consequent transformation of these models (in relation to the first group of works)
- A combination of conceptions of a generator (for the synthesis of KB codes) and an interpreter (for the synthesis of PSM specifications)
- A combination of the MDE-based and ES-based methods for the development of ESs, that allows further expansion of the set of supported knowledge formalisms (for example, for case-based ESs)
- Usage of RVML to describe PIM and PSM
- Usage of CLIPS and the Drools Rule Language (DRL) as platforms

Table 3: A brief comparison of some works that describe the creation of KBs and ESs on the basis of a MDE approach (SD – Subject Domain, DCM - Domain Conceptual Model)

Criterion/Work	Dunstan (2008)	Nofal and Fouad (2014; 2015)	Shue *et al.* (2009)	Ruiz-Mezcua *et al.* (2011)	Touzi and Messaoud (2009)	Kadhim *et al.* (2013)	Canadas *et al.* (2009)	Chaur (2004)	Cabello *et al.* (2009)	Our
Conception of the implementation: G – Generator I - Interpreter	G	I	G	I	G(for KB), I	G	G	G	G	G(for KB), I
Conceptual models used	XML	WorldNet ontology XML	Ontology (OWL, Protege)	Tree	UML	–	Ontology	Rule models	Conceptual models	Ontology (OWL), UML, Mind Maps
CIM	–	–	–	–	–	–	Conceptual Modeling Language	–	Feature model, Decision Tree	OWL, UML, Mind Maps
PIM	–	–	–	–	–	–	(CML)	–	DCM, Appl. DCM	UML, RVML
PSM	–	–	–	–	–	–	Java, JSF Web, JESS	–	PRISMA	RVML
Platform	HTML, Perl, Prolog	Own	JESS, Java	Own	CLIPS	Prolog	JESS, Java	JESS, Java	PRISMA, C#, .NET	CLIPS, DRL, DSL
Methodology	Own	Own	Own	Own	Own	Own	MDD-based, 2 transfor-Mations	MDD-based, 1 transfor-mation	MDD-based, multi transfor-mations	MDD-based + ES-based, 3 transfor-mations
Initiative	–	–	–	–	–	–	EMF	EMF	MDA	MDA
Universality	No, SD: university courses	Yes, Shell	No, SD: corporate financial rating	Yes, Shell	Yes, Shell	No, SD Shell, diag-nostics	Yes	Yes	No, SD: diagnostics	Yes, Shell
Special language and standard used	– (Ad-hoc)	– (Ad-hoc)	– (Ad-hoc)	– (Ad-hoc)	– (Ad-hoc)	– (Ad-hoc)	+ (ATL)	– (ECore)	– (Ad-hoc)	– (Ad-hoc)
Non-programmers	–	+	–	+	+	+	–	–	+	+

Conclusion

The paper describes the specialization and implementation of the MDA/MDE approach for the prototyping rule-based ESs and KBs. The specialization include: The use of ontology as the CIM, the use of the Rule Visual Modelling Language (RVML) notation to create the PIM and the PSM and the use of CLIPS and DRL as the PDM. The problem statement, basic elements of the modified approach and formalized descriptions of the models and transformations are considered.

The approach proposed is designed for non-programmers: Experts and system analytics who can only develop two information models: A CIM (ontology) and PIMs (models of a rule-based KB and ES). In this case, it is possible to automate the PIMs' creation with automated analyses of conceptual models (UML class diagrams) (Dorodnykh and Yurin, 2015). According to the MDA/MDE approach, other models are either integrated into the software that implements the approach or they are created automatically up to the testing stage. At the testing stage, the user can check the developed KB for completeness and validity.

The benefits of the approach proposed in comparison with the standard method of ES development (Jackson, 1998; Giarratano and Riley, 2004; Nofal and Fouad, 2014) are as follows:

- A significant reduction of time for the implementation stage and the elimination of programming errors through automatic code generation
- A reduction of time for the identification, conceptualization and formalization stages due to the use of an ontology and cognitive graphics

The approach proposed is implemented in the form of a research prototype of software that is intended for the rapid development of prototypes of rule-based KBs and ESs. The main advantages of the personal knowledge base designer are listed below:

- Built-in editor of models
- Integration with IBM Rational Rose (in terms of imports of UML-models)
- Generation of CLIPS and DRL code and specifications for the interpreter
- Usage of models at runtime

The approach and software proposed were used for the development of the ES for defining the causes of damage and destruction of construction materials (Berman *et al.*, 2015) and they were also used in the educational process at the Irkutsk National Research Technical University (IrNRTU).

Acknowledgement

The reported study was partially supported by RFBR (research projects No. 15-07-03088, 15-07-05641, 16-37-00041).

Author's Contributions

Alexander Yurievich Yurin: He devised the main conceptual ideas, designed the research plan, made an implementation and experiments.

Nikita Olegovich Dorodnykh: He made model transformations and experiments.

Olga Anatolievna Nikolaychuk: She built models and made a formalization.

Maksim Andreevich Grishenko: He made an implementation.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Arendt, T., E. Biermann, S. Jurack, C. Krause and G. Taentzer, 2010. Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. In: Model Driven Engineering Languages and Systems, Petriu, D.C., N. Rouquette and O. Haugen (Eds.), Springer, Berlin, Heidelberg, SBN-10: 978-3-642-16145-2, pp: 121-135.

Balasubramanian, D., A. Narayanan, C. Buskirk and G. Karsai, 2007. The graph rewriting and transformation language: GreAT. Electronic Commun. EASST, 1: 1-8.

Baumeister, J. and A. Striffler, 2015. Knowledge-driven systems for episodic decision support. Knowledge-Based Syst., 88: 45-56. DOI: 10.1016/j.knosys.2015.08.008

Berman, A.F. and O.A. Nikolaichuk, 2007. Technical state space of unique mechanical systems. J. Machinery Manufacture Reliability, 36: 10-16. DOI: 10.3103/S1052618807010025

Berman, A.F., O.A. Nikolaichuk, A.Y. Yurin and K.A. Kuznetsov, 2015. Support of decision-making based on a production approach in the performance of an industrial safety review. Chem. Petrol. Eng., 50: 730-738. DOI: 10.1007/s10556-015-9970-x

Berman, A.F., O.A. Nikolaychuk and A.Y. Yurin, 2010. Intelligent planner for control of failures analysis of unique mechanical systems. Expert Syst. Applic., 37: 7101-7107. DOI: 10.1016/j.eswa.2010.03.005

Cabello, M.E., I. Ramos, A. Gomez and R. Limon, 2009. Baseline-oriented modeling: An MDA approach based on software product lines for the expert systems development. Proceedings of the 1st Asian Conference on Intelligent Information and Database Systems, Apr. 1-3, IEEE Xplore Press, Dong Hoi, Vietnam, pp: 208-213. DOI: 10.1109/ACIIDS.2009.15

Canadas, J., J. Palma and S. Tunez, 2009. InSCo-Gen: A MDD Tool for Web Rule-Based Applications. Web Eng., 5648: 523-526. DOI: 10.1007/978-3-642-02818-2_53

Chaur, G.W., 2004. Modeling rule-based systems with EMF. Eclipse Corner articles.

ClipsWin, 2017. ClipsWin: CLIPS Rule Based Programming Language.

Corsar, D. and D.H. Sleeman, 2008. Developing knowledge-based systems using the semantic web. Proceedings of the International Conference on Visions of Computer Science: BCS International Academic Conference, Sept. 22-24, BCS Learning and Development Ltd. Swindon, UK, pp: 29-40.

Czarnecki, K. and S. Helsen, 2006. Feature-based survey of model transformation approaches. IBM Syst. J., 45: 621-645. DOI: 10.1147/sj.453.0621

Czarnecki, K. and U. Eisenecker, 2000. Generative Programming: Methods, Tools and Applications. 1st Edn., Addison-Wesley Professional, ISBN-10: 0201309777, pp: 864.

Distante, D., P. Pedone, G. Rossi and G. Canfora, 2007. Model-driven development of web applications with UWA, MVC and JavaServer Faces. In: Web Engineering, Baresi, L., P. Fraternali and G.J. Houben (Eds.), Springer, Berlin, Heidelberg, ISBN-10: 978-3-540-73597-7, pp: 457-472.

Djurić, D., D. Gašević and V. Devedžić, 2005. Ontology modeling and MDA. J. Object Technol., 4: 109-128. DOI: 10.5381/jot.2005.4.1.a3

Dorodnykh, N.O. and A.Y. Yurin, 2015. Using UML class diagrams for design of rule-bases. Software Eng., 4: 3-9.

Dunstan, N., 2008. Generating domain-specific web-based expert systems. Expert Syst. Applic., 35: 686-690. DOI: 10.1016/j.eswa.2007.07.048

EMF, 2017. Eclipse modeling framework.

Epsilon, 2017. http://www.eclipse.org/epsilon

Frankel, D., 2003. Model Driven Architecture: Applying MDA to Enterprise Computing. 1st Edn., Wiley, New York, ISBN-10: 0471319201, pp: 352.

Gardner, T., C. Griffin, J. Koehler and R. Hauser, 2003. A review of OMG MOF 2.0 query/views/transformations submissions and recommendations towards the final standard. Proceedings of the MetaModelling for MDA Workshop, (MMW' 03), Object Management Group, pp: 1-20.

Gascueña, J.M., E. Navarro, A. Fernández-Caballero and R. Martínez-Tomás, 2014. Model-to-model and model-to-text: looking for the automation of VigilAgent. Expert Syst., 31: 199-212. DOI: 10.1111/exsy.12023

Gascueña, J.M., E. Navarro and A. Fernández-Caballero, 2012. Model-driven engineering techniques for the development of multi-agent systems. Eng. Applic. Artificial Intell., 25: 159-173. DOI: 10.1016/j.engappai.2011.08.008

Gašević, D., D. Djurić and V. Devedžić, 2009. Model Driven Engineering and Ontology Development. 2nd Edn., Springer-Verlag, New York, ISBN-10: 978-3-642-00281-6, pp: 378.

Giarratano, J.C. and G. Riley, 2004. Expert Systems: Principles and Programming. 4th Edn., Course Technology, ISBN-10: 0534384471, pp: 288.

IBM Rational Rose, 2017. IBM Rational Rose Enterprise.

Jackson, P., 1998. Introduction to Expert Systems. 3rd Edn., Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, ISBN-10: 0201876868, pp: 542.

Jouault, F., F. Allilaire, J. Bézivin and I. Kurtev, 2008. ATL: A model transformation tool. Sci. Comput. Programm., 72: 31-39. DOI: 10.1016/j.scico.2007.08.002

Kadhim, M.A., M.A. Alam and H. Kaur, 2013. Design and implementation of intelligent agent and diagnosis domain tool for rule-based expert system. Proceedings of the International Conference on Machine Intelligence Research and Advancement, Dec. 21-23, IEEE Xplore Press, Katra, India, pp: 619-622. DOI: 10.1109/ICMIRA.2013.129

Kleppe, A., J. Warmer and W. Bast, 2003. MDA Explained: The Model Driven Architecture: Practice and Promise. 1st Edn., Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, ISBN-10: 032119442X, pp: 170.

Liebowitz, J., 1998. The Handbook of Applied Expert Systems. 1st Edn., CRC Press, Boca Raton, FL., ISBN-10: 9780849331060, pp: 736.

Luger, G.F., 2008. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. 6th Edn., Addison-Wesley, ISBN-10: 0321545893, pp: 784.

MDA, 2017. OMG's model driven architecture.

Mens, T. and P.V. Gorp, 2006. A taxonomy of model transformations. Electronic Notes Theor. Comput. Sci., 152: 125-142. DOI: 10.1016/j.entcs.2005.10.021

MIC, 2017. Model integrated computing.

Miguel, M., J. Jourdan and S. Salicki, 2002. Practical Experiences in the Application of MDA. In: The Unified Modeling Language, Jézéquel J.M., H. Hussmann and S. Cook (Eds.), Springer, Berlin, Heidelberg, SBN-10: 978-3-540-45800-5 pp: 128-139.

Nalepa, G.J. and A. Ligęza, 2010. The HeKatE methodology, hybrid engineering of intelligent systems. Int. J. Applied Math. Comput. Sci., 20: 35-53. DOI: 10.2478/v10006-010-0003-9

Neto, R., P.J. Adeodato and A.C. Salgado, 2017. A framework for data transformation in credit behavioral scoring applications based on model driven development. Expert Syst. Applic., 72: 293-305. DOI: 10.1016/j.eswa.2016.10.059

Nofal, M. and K.M. Fouad, 2014. Developing web-based semantic expert systems. Int. J. Comput. Sci. Issues, 11: 103-110.

Nofal, M.A. and K.M. Fouad, 2015. Developing web-based Semantic and fuzzy expert systems using proposed tool. Int. J. Comput. Applic., 112: 38-45. DOI: 10.5120/19682-1414

QVT, 2017. Meta Object Facility (MOF) 2.0 query/view/transformation, V 1.3. Object Management Group.

PKBD, 2017. Personal knowledge base designer.

Rajput, Q., N.S. Khan, A. Larik and S. Haider, 2014. Ontology based expert-system for suspicious transactions detection. Comput. Inform. Sci., 7: 103-114. DOI: 10.5539/cis.v7n1p103

Rozenberg, G., 1999. Handbook of Graph Grammars and Computing by Graph Transformations. 1st Edn., World Scientific Publishing Company, ISBN-10: 978-981-02-4020-2, pp: 720.

Ruiz-Mezcua, B., A. Garcia-Crespo, J. Lopez-Cuadrado and I. Gonzalez-Carrasco, 2011. An expert system development tool for non AI experts. Expert Syst. Applic., 38: 597-609. DOI: 10.1016/j.eswa.2010.07.009

RVML, 2017. Rule Visual Modeling Language (RVML).

Rybina, G.V., V.M. Rybin, Y.M. Blokhin and S.S. Parondzhanov, 2016. Intelligent programm support for dynamic integrated expert systems construction. Proc. Comput. Sci., 88: 205-210. DOI: 10.1016/j.procs.2016.07.426

Sahin, S., M.R. Tolun and R. Hassanpour, 2012. Hybrid expert systems: A survey of current approaches and applications. Expert Syst. Applic., 39: 4609-4617. DOI: 10.1016/j.eswa.2011.08.130

Sami, B., M. Book and V. Gruhn, 2005. Model-Driven Software Development. 1st Edn., Springer-Verlag Berlin Heidelberg, ISBN-10: 978-3-540-25613-7, pp: 464.

Schmidt, D.C., 2006. Guest editor's introduction: Model-driven engineering. Computer, 39: 25-31. DOI: 10.1109/MC.2006.58

Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog and N. Shadbolt *et al.*, 2000. Knowledge Engineering and Management: The Common KADS Methodology. 1st Edn., The MIT Press, Cambridge, ISBN-10: 0-262-19300-0, pp: 455.

Sendall, S. and W. Kozaczynski, 2003. Model transformation – the heart and soul of model-driven software development. IEEE Software, 20: 42-45. DOI: 10.1109/MS.2003.1231150

Shue, L., C. Chen and W. Shiue, 2009. The development of an ontology-based expert system for corporate financial rating. Expert Syst. Applic., 36: 2130-2142. DOI: 10.1016/j.eswa.2007.12.044

Touzi, A. and M.B. Messaoud, 2009. New approach for conception and implementation of object oriented expert system using UML. Int. Arab J. Inform. Technol., 6: 99-106.

Varro, D. and A. Balogh, 2007. The model transformation language of the VIATRA2 framework. Sci. Comput. Programm., 63: 214-234. DOI: 10.1016/j.scico.2007.05.004

XSLT, 2017. XSL Transformations (XSLT), version 2.0.

Zagorulko, Y. and G. Zagorulko, 2013. Ontology-based program shell for building and editing multilingual thesauri of subject domains. Proceedings of the 12th International Conference on Intelligent Software Methodologies, Tools and Techniques, Sept. 22-24, IEEE Xplore Press, Budapest, Hungary, pp: 99-106. DOI: 10.1109/SoMeT.2013.6645680