Web-based Software for Automating Development of Knowledge Bases on the Basis of Transformation of Conceptual Models

Dorodnykh N.O.
Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences (IDSTU SB RAS),
Irkutsk, Russia
Email: tualatin32@mail.ru

***Abstract*—The paper describes the web-oriented software designed for the development of intelligent system software packages (modules) intended for the code generation of knowledge bases based on the transformation of conceptual models. Software can be positioned as a framework for the development of knowledge bases also. It provides network access to a shared pool of knowledge bases projects and means for the visual design of knowledge bases.**

Keywords*—web-based software, knowledge acquisition, knowledge bases, conceptual model, model transformation, code generation.

I. Introduction

Currently the complexity of development process of intelligent systems is caused, mainly, by the features of the development phase of the knowledge bases (KB) which traditionally has been considered as a "bottleneck" [1]. The efficiency of this phase can be improved through the use of specialized software.

This paper describes the prototype of web-based software called the Knowledge Base Development System (KBDS). The KBDS implements the approach to the development of software components (modules) for the program code generation for KBs based on conceptual models presented in the XML format (the most common format for the exchange and storage of the different conceptual models) [2]. The C Language Integrated Production System (CLIPS) [3] and the Web Ontology Language (OWL 2) [4] were selected as the targeted knowledge base programming languages. This software is also an environment (framework) for the KBs development and it provides network access to a shared pool of KB projects and means for the visual design of KBs.

The KBDS prototype was implemented in PHP using the Yii2 Framework and JQuery, jsPlumb libraries. PostgreSQL was used as the object-relational database management system.

II. Knowledge Base Development System

The main purposes of the developed software are:

- the support of the development of software components for the program code generation for KBs based on different conceptual models;
- the support of the development of KBs (with the use of the developed software components).

Further we present the main functions and the architecture of KBDS in detail.

A. Functions

The main functions of KBDS in the context of the development of the software components are the following:

- the creation (import) of a meta-model of the source conceptual model on the basis on the XML schema (XSD) of this model, or by analyzing the model (Reverse Engineering);
- the visual representation and modification of the obtained meta-models;
- the visual design of a transformation model (in the form of a set of transformation rules) and the automatic TMRL (Transformation Model Representation Language) [2] code generation. In this case, a transformation rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language;
- the generation (assembling) of the software component based on the developed transformation model and the analyzer and generator units selected (depends on the type of the software component).

The KBDS provides the following main functions for the design of the KBs:

- the code generation for KBs on the targeted knowledge base programming language (CLIPS or OWL) using the software components developed;
- the automated synthesis of an ontological and a rule-based model (the internal representation of knowledge in the KBDS) based on the analysis of conceptual models;
- the storage and representation of obtained knowledge with the use of these models;
- the use of the special graphic notation - Rule Visual Modeling Language (RVML) [5] for the representation and modeling the logical rules;
- the visual representation and modeling knowledge in the form of ontological model.

B. Architecture

The KBDS has the client-server architecture (Figure 1) which allows to implement the main factions.

The client part of the KBDS includes the following main modules:

- the RVML editor that provides a visual representation and editing the logical rules with the aid of RVML;
- the ontology editor that provides a visual representation and editing knowledge in the form of a graph (ontological model);
- the meta-model editor that provides a visual representation and editing the meta-model elements;
- the transformation model editor that provides a visual representation and editing the transformation rules.

The server part of the KBDS includes the following main modules:

- the administration module that provides an user interaction with the KBDS (the limitation of user rights, the collection and analysis of various statistical information, etc.);
- the knowledge bases management module that provides a creation and managing KB projects;
- the meta-level management module that provides an internal representation of knowledge in the KBDS in the form of the rule-based and ontological models. These models allow us to abstract form the features in elements descriptions of various knowledge representation languages which are used in the implementation of KBs (for example, CLIPS, Jess, Drools, RuleML, OWL, SWRL, etc.) and to store knowledge in own independent format;
- the software components development module that provides the creation and managing software component projects, as well as code generation of the software components based on the developed transformation model and the analyzer and generator units selected;
- software components that provide a synthesis of the KB model (ontological or rule-based models) based on the analysis of conceptual models and a program code generation for a KB (CLIPS or OWL) based on the analysis of the KB model or the source conceptual model.

The architecture of a typical software component and a conceptual KBDS architecture are discussed in [2], [6] in detail.

C. Types of the software components

The KBDS allows to develop the following types of software components depending of transformations:

- the integrated components for the analysis that provide a formation of the rule-based or the ontological models based on the transformation of a conceptual model;

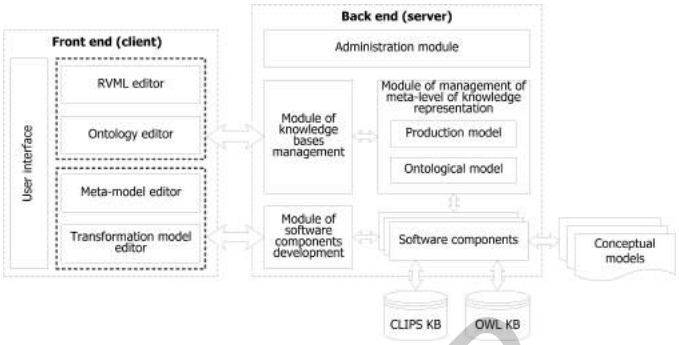

Figure 1. The client-server architecture of the KBDS

- the autonomous components for the program code generation that provide a program code generation for a KB on CLIPS or OWL based on the transformation of a conceptual model.

At the same time, the previously developed integrated software components for the code generation for CLIPS and OWL are provided to the user by default.

Description of the components is discussed in [2] in detail.

D. User roles

The developed prototype of web-based software (KBDS) is focused on the non-programmers (experts, knowledge engineers, analysts, etc.).

The KBDS supports the following user roles:

1) "guest" is a unregistered user, which can develop the KBs only (has limited access rights);
2) "developer" is a registered user, which can develop the KBs and software components, create user groups (has extended access rights);
3) "administrator" is a registered user who have access to all functions of the KBDS (has full set of access rights).

E. User interface

The user interface of web-based software (KBDS) is composed of five main elements (Figure 2):

1) the main menu that contains the main KBDS sections ("Account", "My projects", "Contacts" and "Administration");
2) the navigation chain (breadcrumbs) that is a path through the KBDS pages from the root to the current worksheet;
3) the right additional submenu contains all the possible actions that are available to the user within the section selected;
4) the workspace that is an basic element of the user interface that contains the semantic content of the selected page (section);
5) the bottom block (footer) that contains contact information and copyright.

Figure 3 shows the meta-model editor form (the meta-model for the XTM CmapTools concept maps is opened).

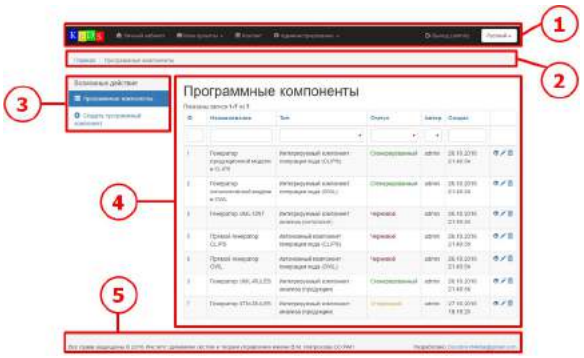

Figure 2. The main element of the KBDS user interface

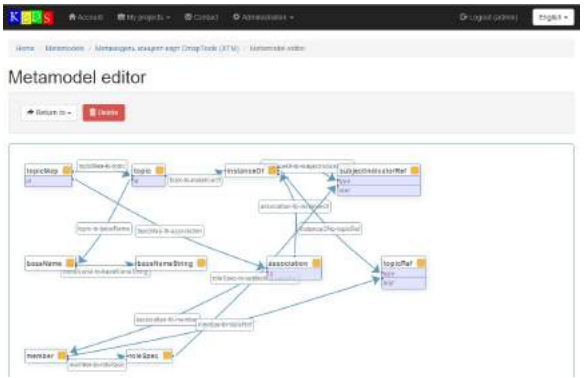

Figure 3. The meta-model editor form

Figure 4 shows the transformation model editor form (the scenario for transformation of the XMI UML class diagrams to rule-based model is opened).

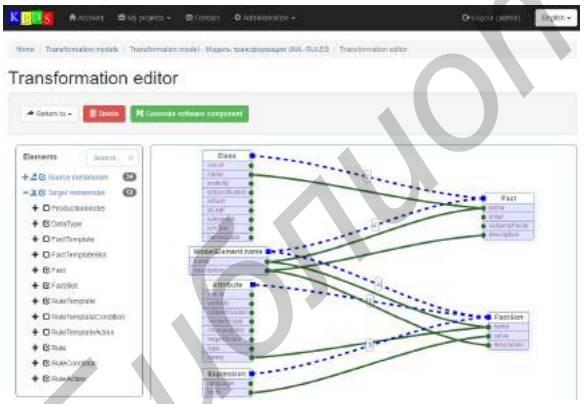

Figure 4. The transformation model editor form

Figure 5 shows the form of of generated transformation model code.

Figure 6 shows the RVML editor.

III. An illustrative example

Let's consider an illustrative example of a knowledge base design fragment using the proposed approach and software. IBM Rational Rose Enterprise was used as the UML-model source. The example analysed UML class diagram is presented in Figure 7.

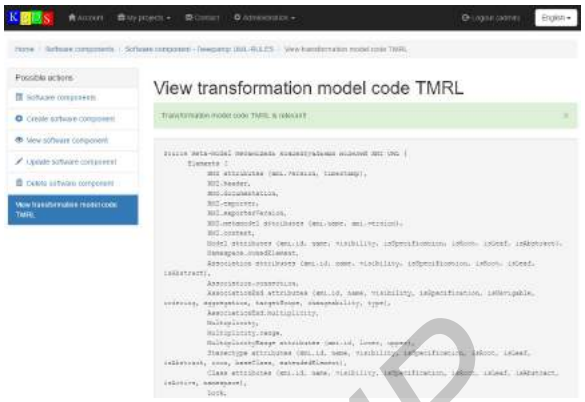

Figure 5. The form of generated transformation model code

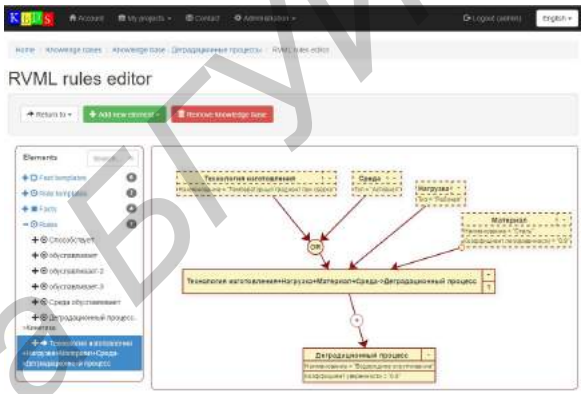

Figure 6. The RVML editor

The following XML document fragment corresponds to the UML-model is obtained:

```
...
<UML:Stereotype xmi.id = 'S.339.1116.39.0'
    name = 'assert' visibility = 'public'
    isSpecification = 'false' isRoot = '
        false'
    isLeaf = 'false' isAbstract = 'false'
    icon = '' baseClass = 'Association'
    extendedElement = 'G.13' />
<UML:Class xmi.id = 'S.339.1116.38.1'
    name = 'Sole' visibility = 'public'
    isSpecification = 'false' isRoot = '
        true'
    isLeaf = 'true' isAbstract = 'false'
    isActive = 'false' namespace = 'G.0' >
  <UML:Classifier.feature>
    <UML:Attribute xmi.id = 'S
        .339.1116.38.2'
        name = 'Material' visibility = '
            private'
        isSpecification = 'false'
            ownerScope = 'instance'
        changeability = 'changeable'
            targetScope = 'instance'
        type = 'G.16' >
        <UML:StructuralFeature.
            multiplicity>
```

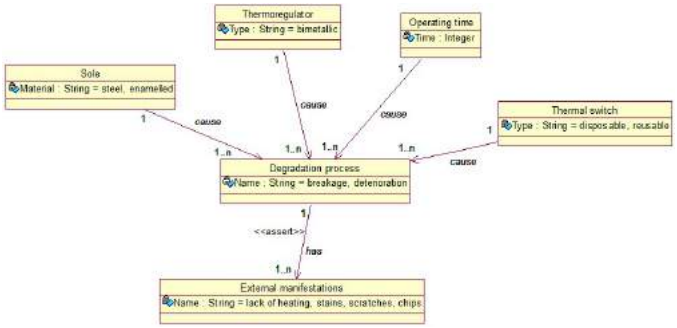

Figure 7. The analysed UML class diagram

```
            <UML:Multiplicity >
              <UML:Multiplicity.range>
                <UML:MultiplicityRange xmi
                    .id = 'id.3400316.11'
                  lower = '1' upper = '1'
                    />
              </UML:Multiplicity.range>
            </UML:Multiplicity>
          </UML:StructuralFeature.
              multiplicity>
          <UML:Attribute.initialValue>
              <UML:Expression language = ''
                  body = 'steel, enamelled'
                    />
          </UML:Attribute.initialValue>
      </UML:Attribute>
    </UML:Classifier.feature>
</UML:Class>
<UML:DataType xmi.id = 'G.16'
      name = 'String' visibility = 'public
          '
      isSpecification = 'false' isRoot = '
          false'
      isLeaf = 'false' isAbstract = 'false
          ' />
...
```

Furthermore, the user creates a new KB project and imports conceptual model of a subject domain (in our case, the industrial safety examination of petrochemical facilities).

The concepts and relations extracted from this XML document and mapped to the production model can be presented using the RVML notation for further modification and validation (Figure 8).

The following generated CLIPS KB code fragment corresponds to the RVML model is obtained:

```
...
(defrule Sole+Thermoregulator+Operating
    time+Thermal
  switch->Degradation process
    (declare (salience 2))
    (Sole (Material "steel, enamelled") )
    (Thermoregulator (Type "bimetallic") )
    (Operating time (Time "") )
    (Thermal switch (Type "disposable,
        reusable") )
```

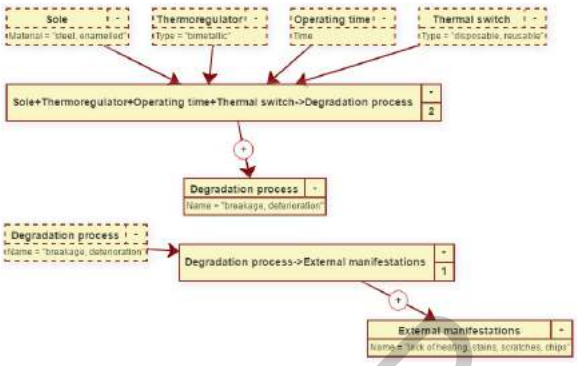

Figure 8. The obtained RVML model

```
    =>
    (assert
        (Degradation process (Name "
            breakage, deterioration") )
    )
)
(defrule Degradation process->External
    manifestations
    (declare (salience 1))
    (Degradation process
        (Name "breakage, deterioration")
    )
    =>
    (assert
        (External manifestations
            (Name "lack of heating, stains
                , scratches, chips")
        )
    )
)
```

IV. Discussion

The efficacy evaluation of the use of the developed software (KBDS) is carried out within a case study. The main objective of the case study was to assess the complexity of the development of KBs for expert systems with the use of the proposed approach and software (a UML-modeling tool + the KBDS, let's denote this approach as A1) and compare it with the complexity of other approaches:

- a UML-modeling tool [7] + the other software for the KB design, in particular, ClipsWin [9] (let's denote this approach as A2);
- without any a UML-modeling tool, but with the use of software for the KB design, in particular, ClipsWin (just a pure programmer's approach, let's denote this approach as A3).

IBM Rational Rose is chosen as a UML-modeling tool [8].

There are 20 variants (tasks) for the design of static expert systems for solving problems of diagnosing or prognosis in different subject areas (Table 1). Some constraints were imposed on the characteristics of subject areas models and KBs (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 the possibility of multiple repetitions of the tasks and their time compactness.

Table I. VARIANTS OF TEST TASKS

Variant	Domain entities	Connections	Cause-effect relations	Rules
1	6	5	3	10
2	5	6	3	10
3	8	5	3	10
4	5	8	4	11
5	8	7	3	12
6	9	5	3	10
7	5	6	3	14
8	8	7	4	14
9	6	5	3	15
10	7	10	3	12
11	5	6	3	11
12	5	6	3	12
13	7	7	3	14
14	8	5	3	11
15	7	6	3	18
16	6	8	3	14
17	6	5	3	11
18	8	7	3	12
19	7	8	3	10
20	5	7	3	12

The time criterion 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 [1], [10]: conceptualization, formalization, realization.

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 a syntactically corrected program codes of the KBs, checked on the adequacy and consistency.

For each variant (task) 3 results describing the time used are obtained, the average of their values are presented in Figure 9.

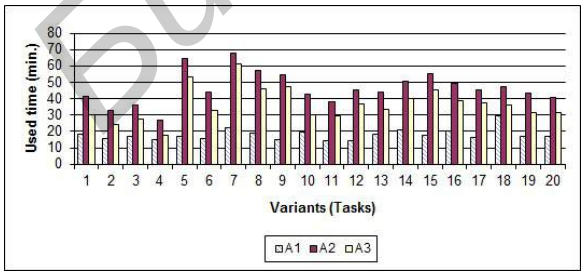

Figure 9. The results of the evaluation

Let's note features of performing the work at various stages for different approaches:

- A2: 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 hand transfer of the conceptual models obtained to ClipsWin (due to the absence the function of automatic code generation for the KB on the basis of conceptual models);
- A3: Functional limitations of the ClipsWin in the part of editing typing codes caused the application of the additional text editor (Programmer's Notepad) at the stage of realization. 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 KBs took 1.5 time less.

Analysis of the effectiveness of the proposed approach and software (KBDS) by the time criteria showed that the effectiveness of the development of KBs by the A1 can be increased 60.3% vs. A2 and 48.2% vs. A3 in an average due to automatic code generation based on conceptual models, which in turn allows:

- to increase 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 [11]. A Model-Driven Approach (Model-Driven Engineering or Model-Driven Architecture) is a software design approach which uses the conceptual (information) models as the major artifacts for software development [12], [13];
- to reduce the risk of design errors by enabling rapid prototyping KBs and getting their program codes;
- to eliminate programming errors (hand coding errors) by automatically transferring the elements of the conceptual models in CLIPS language constructs.

In addition to test cases designed to demonstrate the principal possibility of the application of the approach and its advantages, this approach was used to design the KB of the decision-support system for the industrial safety examination of petrochemical facilities [14], in the part of the definition the degradation processes and recommendations to reduce their rate.

V. CONCLUSION

The development of KBs for intelligent systems can be improved by acquiring subject domain knowledge presented in the form of conceptual models. This paper describes the prototype of web-based software called the Knowledge Base Development System (KBDS). This software implements the approach proposed by the authors [2].

The description of main functions, the client-server architecture, the user interface, user roles and types of software components are presented. The efficacy evaluation of the use

of the developed prototype of web-based software (KBDS) is also given.

There are some features of the KBDS:

- the program code generation for CLIPS or OWL based on the transformation of conceptual models presented in the XML format. This feature significantly reduces the creation time for KBs. The proposed approach does not eliminate errors due to inaccurately or incompletely analysed conceptual models, however, the automatic model-based program code generation uses the principles of rapid prototyping to implement KBs;
- the modularity: the ability to expand the KBDS by adding new software components;
- the visual construction of the transformation rules and the automatic TMRL code generation;
- the use of the rule-based and ontological models for generalized representation and storage of acquired knowledge that, in turn, allows to support the code generation for various knowledge representation languages (eg, CLIPS, OWL, etc.);
- the non-programmers orientation. This feature is implemented by a set of editors that provide the description of the logical rules in the RVML and ontological (conceptual) knowledge in the form of a graph. The implementation of this feature expands the community of the KBDS users by experts, knowledge engineers, analysts;
- the possibility to support the collective, distributed work of specialists in the process of creating the KBs.

At present, the prototype of software is used in the learning process in Irkutsk National Research Technical University (IrNRTU) with in "CASE-tools", "Means of information technologies" and "Programming technologies" courses.

Future work will focus on improving the software support for the OWL2 as another target knowledge representation language and the algorithms support for the analysis of fuzzy conceptual models.

ACKNOWLEDGEMENT

The reported study was partially supported by RFBR (research projects No. 15-07-05641, 16-37-00122).

REFERENCES

[1] Gavrilova T.A., Kudryavtsev D.V., Muromtsev D.I. *Knowledge Engineering. Models and methods.* SPb.: Lan, 2016. 324 p. (In Russ.)

[2] Bychkov I.V., Dorodnykh N.O., Yurin A.Yu. *Approach to the development of software components for generation of knowledge bases based on conceptual models* // Computational Technologies. 2016. Vol.21, No.4. P.16-36. (In Russ.)

[3] *CLIPS: A Tool for Building Expert Systems.* Available at: http://clipsrules.sourceforge.net/ (accessed 25.11.2016).

[4] Grau B.C., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U. *OWL 2: The next step for OWL* // Web Semantics: Science, Services and Agents on the World Wide Web. 2008. Vol.6, No.4. P.309-322.

[5] *Rule Visual Modeling Language.* Available at: http://www.knowledge-core.ru/index.php?p=rvml (accessed 25.11.2016).

[6] Dorodnykh N.O., Yurin A.Yu. *An approach for design of knowledge bases on the basis of computer-aided transformation of conceptual models* // Proceedings of the VI International Scientific and Technical Conference - Open Semantic Technologies for Intelligent Systems (OSTIS-2016). – Minsk: BSUIR, 2016. P.209-212. (In Russ.)

[7] *Unified Modeling Language (UML) Version 2.5* // OMG Document formal/15-03-01. Available at: http://www.omg.org/spec/UML/2.5/ (accessed 25.11.2016).

[8] *IBM Rational Rose Enterprise.* Available at: http://www-03.ibm.com/software/products/ru/enterprise/ (accessed 25.11.2016).

[9] *ClipsWin: CLIPS Rule Based Programming Language.* Available at: https://sourceforge.net/p/clipsrules/news/2008/01/clipswin-6241/ (accessed 11.06.2016).

[10] Jackson P. *Introduction To Expert Systems.*, 3rd ed. Addison-Wesley, 1998. 560 p.

[11] Stahl T., Voelter M., Czarnecki K. *Model-Driven Software Development: Technology, Engineering, Management.*, 1rd ed. John Wiley & Sons, 2006, 446 p.

[12] Schmidt D.C. *Model-driven engineering* // IEEE Computer. 2006. Vol.39, No.2. 25 p.

[13] Kleppe A., Warmer J., Bast W. *MDA Explained: The Model Driven Architecture: Practice and Promise*, 1rd ed. New York: Addison-Wesley Professional, 2003. 192 p.

[14] Berman A.F., Nikolaichuk O.A., Yurin A.Yu., Kuznetsov K.A. *Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review.* // Chemical and Petroleum Engineering. Vol.50, No.1-2. 2015. P.730-738.

ВЕБ-ОРИЕНТИРОВАННАЯ ПРОГРАММНАЯ СИСТЕМА АВТОМАТИЗАЦИИ РАЗРАБОТКИ БАЗ ЗНАНИЙ НА ОСНОВЕ ТРАНСФОРМАЦИИ КОНЦЕПТУАЛЬНЫХ МОДЕЛЕЙ

Дородных Н.О.

Рассмотрена веб-ориентированная программная система (Knowledge Base Development System, KBDS) автоматизации разработки программных компонентов (модулей) интеллектуальных систем. Создаваемые с помощью системы компоненты предназначены для генерации кода баз знаний (представленных в формате CLIPS и OWL) на основе трансформации концептуальных моделей (представленных в формате XML). Система позволяет не только создавать программные компоненты, но она также является средой для разработки баз знаний, обеспечивая сетевой доступ к общему пулу проектов баз знаний и предоставляя средства для визуального проектирования баз знаний. Разработанная система обладает клиент-серверной архитектурой и ориентирована на непрограммирующего специалиста. Приведено ее описание и оценка эффективности.