Development of Software Decision-Making Modules Based on a Model-Driven Approach

Aleksandr Yu. Yurin[a], Nikita O. Dorodnykh[a]

[a]*Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences, Irkutsk, Russia*

Abstract

The computer-aided engineering software modules for decision-making intelligent systems requires the development of specialized methods, algorithms and software. The use of a model-driven approach that implements the principles of generative and visual programming as well as model transformations, is promising. In this paper, we propose an approach for the development of rule-based intelligent system software components in the form of decision-making modules by specializing and using main principles of a model-driven development. The proposed specialization includes using a step-by-step development scheme (chain of model transformations) from information models to source codes and specifications; a method for the automated creation of computation-independent models based on the transformation of spreadsheets; domain-specific tools for formalization, visualization and generation of codes. The developed approach was applied for creating decision-making modules for rule-based intelligent systems.

1. Introduction

The development of intelligent systems and their components (software modules) continues to be a complex and time-consuming task. One of the ways to increase the efficiency of creating such systems is to use the principles of generative and visual programming, as well as the concept of model transformations for conceptualization, formalization and automatic codification.

These principles are implemented in various techniques and tools. However, in most cases, existing solutions are focused on certain software platforms and have high qualification requirements for developers. The integrated use of these principles is also implemented within approaches based on model transformations. Currently they are known as Model-Driven Engineering (MDE) or Model-Driven Development (MDD)[Sil15].

In this paper, we propose a modified standardised MDE approach for computer-aided engineering of software components in the form of decision-making modules for intelligent systems. Our approach includes:

- A step-by-step development scheme in the form of a chain of model transformations (more abstract models are transformed to less abstract models, and source codes and specifications are generated at the end of this chain).

Russian Advances in Artificial Intelligence: selected contributions to the Russian Conference on Artificial intelligence (RCAI 2020), October 10-16, 2020, Moscow, Russia

iskander@icc.ru (A.Yu. Yurin); tualatin32@mail.ru (N.O. Dorodnykh)

- A method for the automated creation of computation-independent models in the form of domain models, this method based on the transformation of spreadsheets with arbitrary layouts and styles.
- Specialized domain-specific tools for formalization, visualization and generation of codes.

Our approach is focused on non-programmers (end-users) and provides algorithms, languages and software for creating rule-based knowledge bases and decision-making modules of intelligent systems. We tested the approach when created the following software prototypes of:

- A module for definition causes of damages and destruction of technical system elements (a part of the "IS Expertise" decision support system[Ber15]).
- A "Detector" module[1] for detection of banned messages and clients who com-promise a SMS mailing service (a part of the "SMS Organizer" platform[2]).
- A module for facial features interpretation and detection of emotions (a part of the "HR Robot" project[3]).

Our approach was also used in educational process at the Institute of Information Technologies and Data Analysis of the Irkutsk National Research Technical University (IrNITU).

2. State-of-Art

2.1. Model-Driven Engineering and Model Transformations

Model transformations, generative and visual programming are widely used practices in software engineering especially in model-driven approaches (MDE, as well as Model-Driven Development (MDD))[Sil15]. The main idea of such approaches is using the transformation and interpretation of information models, in particular, conceptual models, for software development.

Of course, researches related to the model transformations and meta-programming have been conducted since the origin of programming theory and expert systems, including by Russian and Belarusian researchers. In particular, we can highlight the AT-Technology and Open Semantic Technologies for Intelligent Systems (OSTIS) projects. However, they are weakly integrated with international software development standards (for example, UML, XMI, etc.), while some MDE implementations are based on them.

There are some implementations of MDE: Eclipse Modelling Framework (EMF), Model-Integrated Computing (MIC) and Model-Driven Architecture (MDA). MDA[4] is the most standardized ones and suggests the integrated use of OMG (Object Management Group) standards, including:

[1] Yurin, A.Yu.: Detector. Certificate of software registration, № 2020614257, 27.03.2020
[2] SMS-Organizer Home: http://centrasib.ru/index.php?p=smso
[3] Personnel Evaluation Home: http://www.ocenkakadrov.ru
[4] MDA: https://www.omg.org/mda/

- Unified Modeling Language (UML) as a main language for formalizing and visualizing models.
- Meta Object Facility (MOF) as a language for describing metamodels.
- Query/View/Transformation (QVT) as a set of model transformation languages.
- XML Metadata Interchange (XMI) as the standard for textual representation of models.

The MDA approach generates three viewpoints on the software, which are present-ed in the form of corresponding models: a Computation-Independent Model (CIM), a Platform-Independent Model (PIM) and a Platform-Specific Model (PSM). The software development process represents a sequential transformation of these models from the most abstract CIM to a specific PSM. Program codes or specifications are generated at the last step. A model transformation is the automatic generation of a target model from a source model, according to a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. The MDA approach implements a four-level metamodel architecture to support this process[Spr10]. According to this architecture there are following types of model transformations[Men06]: horizontal and vertical, endogenous and exogenous, unidirectional and bidirectional. Metamodels are the part of this architecture and the way to define abstract syntax of languages used in the development process.

2.1.1. Intelligent System and Knowledge Base Engineering Based on Model Transformations

There are several recent studies dealing with knowledge base and intelligent system engineering based on principles of model-driven approaches. Moreover, they can be classified according to the following criteria:

- Used models and transformation chains: approaches using their own methodological principles without a clear classification of models and transformation chains[Dun08, Nof15, Rui11, Shu09, Kad13]; approaches that implement the model transformation within standards[Yur18, Cab09, Can09].
- Target platforms: approaches using general-purpose languages such as Perl, C#, .NET[Dun08, Cab09]; approaches using specialized knowledge representation languages, for example, Jess, Drools, CLIPS and others[Shu09, Yur18, Can09, Cha04]; approaches using own original languages and means, for example, PRISMA[Cab09].
- Sources of domain information: ontology-based approaches[Nof15, Shu09, Can09]; UML-based approaches[Yur18]; semantic trees-based approaches[Rui11] and approaches based on XML-like structures[Dun08].
- End users: programmers-oriented approaches[Dun08, Shu09, Can09, Cha04] or non-programmers-oriented approaches[Nof15, Kad13, Yur18, Cab09].

The analysis of these studies showed that they are focused primarily on users with high programming and metamodeling skills. So, as a rule, existing solutions do not clearly correspond to the MDA methodology, for example, skipping the CIM construction stage, going directly to the PIM development stage. However, CIM plays an important role, it bridging the gap that usually exists between domain experts and information technology specialists. This stage is very important when developing intelligent systems; it results to a conceptual domain model and knowledge bases represented by certain formalisms, for example, rules.

2.2. Background

In our previous work[Yur18] we have proposed an approach for the rule-based expert systems and knowledge bases engineering based on MDA principles and model trans-formations. This approach partly eliminated the disadvantages described above and used the Transformation Model Representation Language (TMRL)[Dor18] to describe transformation rules.

The following main stages of model transformations were defined:

Stage 1. Creating domain models. Domain modeling and description of main architectural elements of expert systems are carried out at this stage. The resulting models are considered as a CIM and can be represented in the form of OWL ontology, UML models (e.g. class diagrams), concept maps, etc.

Stage 2. Forming rule-based models in the RVML notation. Domain models created at the previous stage are automatically transformed into rule-based models. In this case, concepts from the domain model are matched with fact templates and elements of logical rules (conditions and actions). Causal relationships are transformed into rules. In fact, an automated formalization of the domain model is carried out. The resulting rule-based models are considered as a PIM and can be represented using a special graphic notation called Rule Visual Modeling Language (RVML)[Yur18].

Stage 3. Modifying rule-based models in accordance with the knowledge representation language. At this stage, the resulting rule-based models are modified in RVML notation taking into account the characteristics of the selected target platforms (knowledge representation or programming languages). For example, when using CLIPS (C Language Integrated Production System), rule priorities and "by default" values of slots are specified. The resulting revised rules (modified RVML models) are considered as a PSM.

Stage 4. Generating program codes and specifications by specialized tools, in particular:

- Web-oriented software called Knowledge Base development System (KBDS)[Dor17] that provides development of converters for transformation domain models to rule-based models.
- Desktop application called Personal Knowledge Base Designer (PKBD)[Yur20] that provides generation of source codes and specifications based on the received rule-based models in the RVML notation.

Stage 5. Testing obtained program codes both in special software (in the PKBD interpreter), and in the application.

Generally, this approach allows experts to prototype rule-based knowledge bases and expert systems on the basis of existing conceptual models (for example, UML class diagrams). Despite a significant reduction of time of the implementation stage and the elimination of programming errors through automatic code generation, the proposed approach has some disadvantages, in particular:

- The long time of the stage of creating domain models by experts: at this stage additional information sources that can reduce the development time and improve the quality of resulted models are not used.
- A limited set of means when creating rule-based models: the RVML notation only used, which is not effective when designing large knowledge bases (more than 50 rules).
- A limited set of supported platforms: CLIPS only supported, which has low integration ability with existing software (including those based on web technologies) in terms of expanding their functionality.

Thus, in this paper, we propose a modified approach to the development of rule-based intelligent systems in the form of embedded web-based decision-making mod-ules, and this modification taking into account the above mentioned factors. So, the main our contributions are the followings:

- A new method for the automated formation of a conceptual domain model (a CIM) based on data extracted from spreadsheets with arbitrary layouts and styles.
- The combined use of a tabular (in the form of a decision table) and a graphical approaches (in the form of RVML schemes) to formalize and visualize the trans-formed models as a PIM.
- Extension of a set of supported platforms by Drools and PHP (Hypertext Preprocessor).

3. Modified Approach

We modify the first four stages of our previous approach. Thus, our modified ap-proach includes the following chain of model transformations:

$$T = \langle T_{IS-CIM}, T_{CIM-PIM}, T_{PIN-PSM}, T_{PSM-CODE} \rangle, \tag{1}$$

when T_{IS-CIM} is a set of transformation rules of information source (e.g., databases, spreadsheets, texts, etc.) into a conceptual domain model; $T_{CIM-PIM}$ is a set of transformation rules of a conceptual domain model into a rule-based model; $T_{PIM-PSM}$ is a set of transformation rules of a rule-based model in general form into a modified rule-based model corresponding to a specific target knowledge representation language; $T_{PSM-CODE}$ is a set of transformation rues of a modified rule-based model into a program code on the target knowledge representation language.

Next, we consider main stages in detail:

Stage 1: Creating domain models. The effectiveness of this stage can be improved by reusing existing information sources. In this paper, we propose to use spreadsheets in the Excel and CSV format. Spreadsheets are a fairly popular means of structuring and storing various types of information. Spreadsheets are commonly used in various domains and can contain a large number of facts. We have developed a special method for the automated construction of a conceptual domain model by transforming data extracted from arbitrary spreadsheets. The method is based on the use of a specific canonicalized (relational) form for the source spreadsheet and rules for its transformation. The method includes the following main steps:

1. Recognizing spreadsheets with arbitrary layouts and styles, and its transfor-mation to a canonicalized (relational) form:

$$CS = \{D, RH, CH\}, \tag{2}$$

when D is a data block; RH is a set of row labels of the category; CH is a set of column labels of the category. The values in cells for heading blocks can be separated by the "|" symbol to divide categories into subcategories. Thus, the canonical table denotes hierarchical relationships between categories (headings). This step results in tables in the unified (canonicalized) form prepared for their further automated processing. A detailed description of this step is given in[Shi17].

2. Generating conceptual model fragments based on the transformation of canonical spreadsheets. In this case, UML class diagrams or concept maps can be used to represent model fragments. According to the MDA approach, transformation is carried out at the metamodel level (abstract level "M2" of a four-level metamodel architecture[Spr10]). For this reason, we have defined metamodels describing the source canonical spreadsheets (Fig. 1), the target UML class diagrams and concept maps[Dor19]. At the same time, transformation rules describing the correspondence between the elements of these metamodels were created on TMRL.

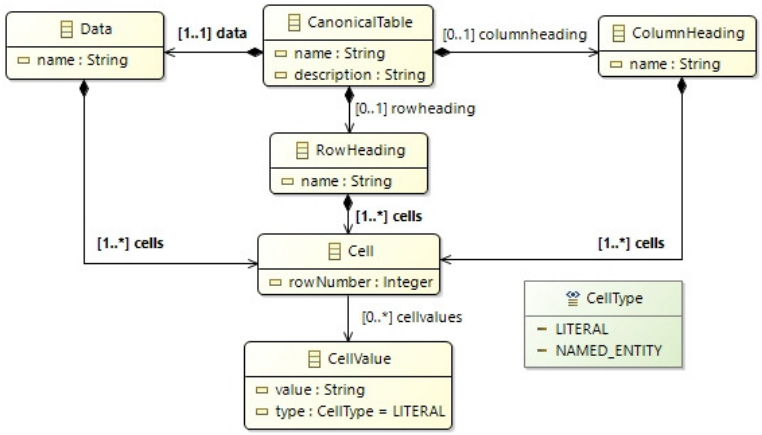

Figure 1: A metamodel of canonical spreadsheets

3. Aggregating conceptual model fragments into one general domain model. This step includes operations for clarifying the names of classes, their attributes and relationships (e.g.,

associations), and also their possible merging and separation. The following main rules used for aggregation of conceptual models fragments:

- Classes with equal names are merged (a common list of attributes is formed).
- When the names of a class and attribute are equal, the attribute of the same name is removed; the corresponding relationship between classes is created.

Manual merging and separation operations are performed by the user using PKBD.
Thus, let's clarify the operator T_{IS-CIM} from (1):

$$T_{IS-CIM} : CS \longrightarrow CM, \tag{3}$$

when CS is a spreadsheet in a canonicalized (relational) form; CM is a conceptual domain model. $CM = \left\langle CM^{UML}, CM^{XTM} \right\rangle$, when CM^{UML} is a conceptual model in the form of a UML class diagram; CM^{XTM} is a conceptual model in the form of a concept map (XML Topic Maps).

Stage 2: Forming decision tables as platform-independent models. At this stage, we pro-pose using the formalism of decision tables containing all properties of concepts of a conceptual domain model (a CIM) with the ability to represent them in RVML for visualization and subsequent modification.

Decision tables are a popular way to describe logical dependencies[Ere16]. Moreover, both specialized tools and general-purpose office suites, in particular, Microsoft Excel, can be used to create them. The main advantage of decision tables is its visibility, especially in the case of large knowledge bases, when the use of graphic schemes leads to clutter or fragmentation of the representation, and the use of programming languages leads to excess information in the form of service operators and symbols.

Thus, let's clarify the operator $T_{CIM-PIM}$ from (1):

$$T_{CIM-PIM} : CM \longrightarrow RM, \tag{4}$$

when CM is a conceptual domain model; RM is a rule-based model which is a universal high-level abstraction of knowledge representation in the form of logical rules that is independent of a target knowledge representation language (e.g., CLIPS, Jess, Drools, RuleML, etc.). Wherein: $RM = \left\langle RM^{DT}, RM^{RVML} \right\rangle$, when RM^{DT} is a rule-based model in the form of a decision table; RM^{RVML} is a rule-based model in the form of RVML diagram.

Stage 3: Modifying rule-based models as platform-dependent models in accordance with the selected knowledge representation language. Automatic conversion of decision tables to RVML models is carried out at this stage. In accordance with our approach, the developer should clarify the resulting RVML models taking into account the characteristics of the target software platform. In addition to CLIPS, we proposed to use:

- Drools, it is a business rule management system that uses an enhanced implementation of the Rete algorithm and supports standard of API Java Rules Engine. The obtained rules can be integrated into existing Java codes.

- PHP (Hypertext Preprocessor), it is a general-purpose scripting language widely used in web applications. PHP modules (units) are easily to integrate, while they do not require the use of additional rule engine or any extensions.

The use of this software platforms (languages) use allows one to integrate the de-veloped decision-making modules (components) into existing web-based intelligent systems.

Thus, let's clarify the operator $T_{PIM-PSM}$ from (1):

$$T_{PIM-PSM} : RM \longrightarrow RM^*, \tag{5}$$

when RM is a rule-based model (rules); R^* is a modified rule-based model for a specific platform (a knowledge representation language).

Stage 4: Generating program codes and specifications using the extended PKBD toolkit that supports new platforms (Drools and PHP) and implements a method for the automated formation of conceptual domain models based on the transformation of spreadsheets.

Thus, let's clarify the operator $T_{PSM-CODE}$ from (1):

$$T_{PSM-CODE} : RM^* \longrightarrow KB, \tag{6}$$

when R^* is a modified rule-based model for a specific platform; KB is source codes of a knowledge base on a target language (e.g., CLIPS, Drools or PHP).

Note that the integration process of the obtained modules requires programming skills.

4. Case Studies

Our modified approach was used to solve practical problems. In our case studies we use various ways for representing domain concepts and its relationships, and also different metamodels and platforms. In particular, we developed prototypes of decision-making modules for the following intelligent systems:

- "IS Expertise"[Ber15] designed to define causes of damages and destructions of technical system elements.
- "SMS Organizer" designed to provide detection of banned messages and clients who compromise a SMS mailing service ("Detector" module).
- "HR Robot" designed to interpret facial features for detection of emotions.

Further, we briefly consider these case studies.

4.1. IS Expertise Software Module

"IS Expertise" (Industrial Safety Expertise) decision support system[Ber15] is intended to automate the process of an industrial safety inspection of chemical and petrochemical equipment. This system automates collecting, storing and processing information for preparing and conducting the expertise and forming conclusions and reports. "IS Expertise" is a client-server

application, which includes the subsystems: data storage (contains information about the inspected objects, experts and other information for the inspection); previewing and entering information; analytical processing that provides decision support on the basis of case-based and rule-based expert system modules.

Let's consider the application of the proposed approach for developing a prototype of rule-based expert system:

Stage 1: Creating domain models. The creation of domain models was carried out based on a model of the dynamics of technical states[Ber07]. This model describes the main concepts in the field of equipment degradation in the petrochemical industry, and includes a description and relationships of the following main concepts: mechanical stresses (properties: magnitude, cycle, amplitude, etc.); materials (properties: name, type, resistance temperature, etc.); contact medium (properties: form, pressure, temperature, chlorine ions, dissolved oxygen, etc.); degradation process (properties: mechanism name, kinetics, etc.). All models designed in the form of concept maps (Fig. 2, block 1).

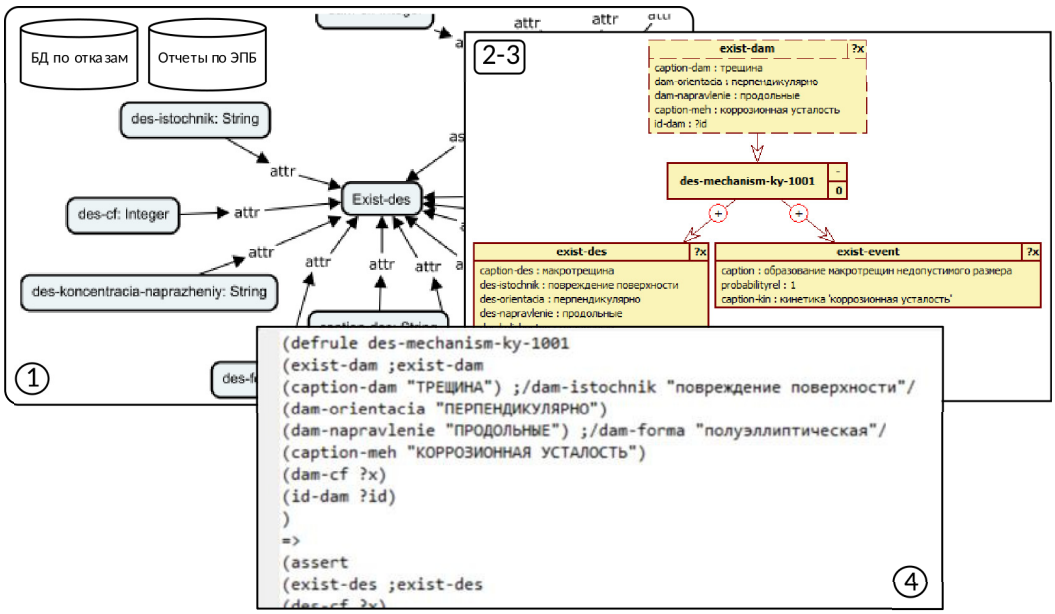

Figure 2: The scheme of developing a prototype of rule-based expert system for "IS Expertise"

In addition to this model, we used information from a database of petrochemical equipment failures[Ber93], as well as the results of an industrial safety inspection in the form of reports containing spreadsheets (we extracted them in the CSV format). As a result, a conceptual domain model was obtained. This model was further considered as a CIM (Fig. 2, block 1).

Stage 2: Forming decision tables as platform-independent models. The formation of platform-independent models was carried out using KBDS, which provided the import of the CIM from the format of concept maps (XML Topic Maps) and its transformation. As a result of this transformation, the concepts and relationships of a conceptual domain model were mapped

with templates for facts and rules represented in the form of RVML schemes (Fig. 2, block 2-3).

Stage 3: The formation of platform-specific models. Target platforms in this case were CLIPS (for describing knowledge base codes) and PKBD (for interpreting expert system specifications). Therefore, a minor modification of the PIM in the form of RVML was required.

Stage 4: Generating program codes and specifications. The generation of program codes and specifications was performed automatically. CLIPS code (Fig. 2, block 4) and specifications for the PKBD interpreter have been obtained. These specifications provide the generation of a user interface for creating, reading, updating and deleting (CRUD) elements of the knowledge base.

Thus, knowledge base was created, including 14 fact templates, 12 rule templates, 4 initial facts and 20 specific rules (for one degradation process).

4.2. Detector Module

Banned messages fall into the category of SPAM messages, the sending of which violates the Russian Federal Law #38 "On Advertising". The specialized web-based software module called "Detector" [4] was developed for a SMS mailing service. The "Detector" aim is to detect SPAM messages and clients who sending them. This module includes an intelligent decision support block with a rule-based knowledge base.

Let's consider the application of the proposed approach for developing this module:

Stage 1: Creating domain models. The creation of domain models allowed us to identify key abstractions and their relationships (Fig. 3, block 1). At the conceptual level, the main concepts are the followings: "SMS message", "Keywords" aka "markers" of SPAM messages and the "Sender". Subsequently, the model was simplified in order to better match the structure of logical rules during formalization.

An analysis of database containing 1 366 490 messages was performed to highlight combinations of markers (keywords). 829 SPAM messages of detected and blocked earlier clients were selected as a training corpus.

As results of the training corpus analysis we identified five main groups of SPAM messages: propaganda of prohibited substances; unfair advertising; fraud; threats and insults; not bearing signs of SPAM. For each group, a possible reaction of the decision support system was determined, in particular: blocking the sending of a message (changing its status) and blocking the client.

Stage 2: Forming decision tables as platform-independent models. The formation of platform-independent models was carried out in the form of a decision table containing information about 487 unique sets of keywords (Fig. 3, block 2). This decision table was developed by automated analysis of a message database. The table structure is formed by listing the properties of all concepts, where each property is a column of this table.

Stage 3: The formation of platform-specific models. The formation of platform-specific models was carried out using PKBD, which provided the import of the developed decision table with subsequent modification of the obtained rules in the form of RVML, including the priority of the rules and "by-default" values (Fig. 3, block 3).

Stage 4: Generating program codes and specifications. The generation of program codes and specifications was performed automatically. Using the PKBD generator, 6 871 lines

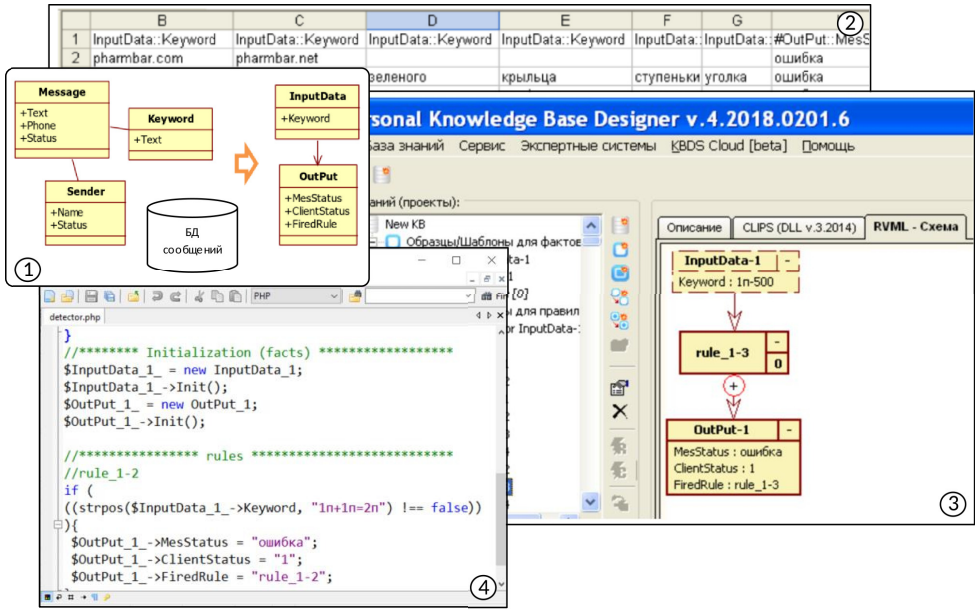

Figure 3: The scheme of developing a prototype of "Detector"

of PHP code were synthesized that describe 487 specific rules (Fig. 3, block 4).

The priorities for rules were taken into account by sorting the atomic conditional operators within the computational block (function). Thus, if the rule was fired than the inference was stopped. In this case, the result of the last fired rule was returned.

Subsequently, the generated code was integrated to the "SMS Organizer" platform. Testing the developed "Detector" module was carried out on the training corpus. In particular, the modified knowledge base contained 498 specific rules, which detected 653 banned messages. Thus the accuracy was 0.83. The average runtime was 0,00026 sec. Checking the database of 1 366 490 messages by the "Detector" module resulted to detection of 1 145 SPAM messages and 25 clients who compromised a SMS mailing service (but who were not previously blocked).

4.3. Emotion Features Interpretation Module

"HR Robot" is a project of a decision support system for selecting candidates for vacancies and checking staff for motivation (study of the psychological situation in the team). "HR Robot" conceptually provides video interview processing, identifica-tion of facial features of emotions and their interpretation. Interpretation of facial features is based on the rule-based knowledge base.

Let's consider the application of the proposed approach for developing this knowledge base prototype:

Stage 1: Creating domain models. The creation of domain models. The obtained models describe face parts and its main elements that will be tracked in detecting emotions. A fragment of one of the models is shown in Fig. 4, block 1.

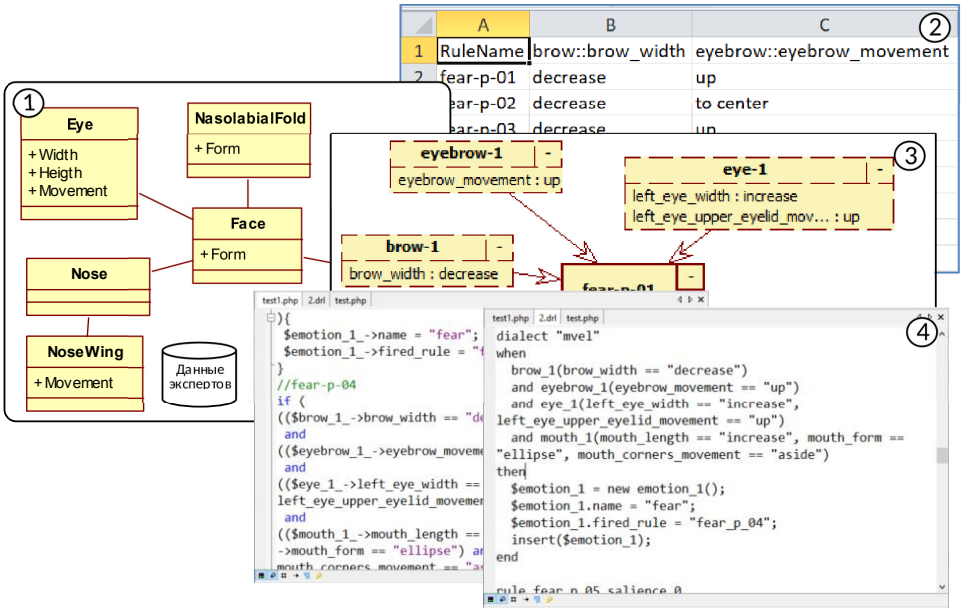

Figure 4: The scheme of developing a prototype of rule-based knowledge base for facial features interpretation

Stage 2: Forming decision tables as platform-independent models. The formation of platform-independent models. On this stage we developed decision tables on the basis of the constructed conceptual domain models and the knowledge of expert psychologists. These tables contain information about combinations of concept values, which can be used for emotions detection, for example, fear (Fig. 4, block 2). In fact, each row in the table is a logical decision-making rule.

Next, PKBD was used. PKBD imported decision tables and transformed them to rule-based knowledge base structures. In the test example, this knowledge base seg-ment for the fear emotion includes: 5 fact templates, 1 rule template, and 11 specific rules.

Stage 3: The formation of platform-specific models. The formation of platform-specific models was also carried out using PKBD. The imported decision tables were refined in the form of RVML diagrams (Fig. 4, block 3).

Stage 4: Generating program codes and specifications. The generation of program codes and specifications was performed automatically for PHP and Drools (Fig. 4, block 4). PHP code in 250 lines and Drools code in 453 lines was generated for this segment of knowledge base.

5. Evaluation

The evaluation of complexity of the proposed approach and tools carried out by the time criterion, using the corpus of test tasks and some results obtained earlier[Yur18, Dor18]. During testing we calculated the time spent for tasks connected with rule-based knowledge bases en-

gineering by the following ways:

- Using PKBD and KBDS in terms of importing conceptual models and decision tables (the MS Excel format is used), as well as automated transformation of tables to knowledge base elements and automatic code generation.
- Manual programming (hand coding) of knowledge base elements in a specialized editor (e.g., Programmers Notepad).

The comparison of these methods showed that the use of the proposed approach can significantly reduce the time spent in comparison with manual coding up to 60%[Dor19]. It should be noted that in this experiment, decision tables were created manually and the test knowledge bases contain 10-12 rules. However, the efficiency can be even higher with the automated generation of decision tables containing a large number of records (487, as in the above example).

The use of decision tables for rule-based knowledge base engineering is a good practice[Ere16], especially with a large number of rules. The use of model transfor-mations and specialized tools reduces the complexity of codification of rules on this context.

6. Conclusion

In this paper, we propose a modified approach for the development of components of rule-based intelligent systems in the form of software decision-making modules. This approach is based on the principles of the standardised model-driven methodology (MDE/MDA) and model transformations.

In our approach we can use spreadsheets as a source of information for creation of computation-independent models (CIM), and reduce the development time for these models. Decision tables are used as an additional formalism for knowledge description that increases the efficiency of processing a large number of rules. The proposed approach supports two additional platforms: Drools and PHP, which simplify the integration of developed decision-making modules with existing web-based intelligent systems.

Our approach is implemented in the form of interacting tools: Personal Knowledge Base Designer (PKBD)[Yur20] and Knowledge Base Development System (KBDS)[Dor17].

Case studies and educational tasks demonstrated the applicability of our proposals.

Acknowledgments

The present study was partially supported by the Russian Foundation for Basic Re-search (Grant no. 19-07-00927). The contribution related to the use of spreadsheets for generating computational-independent models (Sections 2 and 3.1) was supported by the Russian Science Foundation (Grant no. 18-71-10001).

References

[Sil15] A. R. D. Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155, Oct. 2015.

[Ber15] B. F. Berman, O. A. Nikolaichuk, A. Yu. Yurin, K. A., Kuznetsov. Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review. *Chemical and Petroleum Engineering*, 50(1-2):730–738, Mar. 1979.

[Spr10] J. Sprinkle, B. Rumpe, H. Vangheluwe, G. Karsai. Metamodelling: State of the Art and Research Challenges. *Model-Based Engineering of Embedded Real-Time Systems*, 57–76, Oct. 2010.

[Men06] T. Mens, P. V. Gorp. A Taxonomy of Model Transformations. *Electronic Notes in Theoretical Computer Science*, 152:125–142, Mar. 2006.

[Dun08] N. Dunstan. Generating domain-specific web-based expert systems. *Expert Systems with Applications*, 35:686–690, Oct. 2008.

[Nof15] M. A.Nofal, K. M., Fouad. Developing web-based Semantic and fuzzy expert systems using proposed tool. *International Journal of Computer Applications*, 112:38–45, Feb. 2015.

[Rui11] B. Ruiz-Mezcua, A. Garcia-Crespo, J. Lopez-Cuadrado, I. Gonzalez-Carrasco. An expert system development tool for non AI experts. *Expert Systems with Applications*, 38:597–609, Jan. 2011.

[Shu09] L. Shue, C. Chen, W. Shiue. The development of an ontology-based expert system for corporate financial rating. *Expert Systems with Applications*, 36:2130–2142, Mar. 2009.

[Kad13] M. A. Kadhim, M. A. Alam, H. Kaur. 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. IEEE Xplore Press, Katra, India*, 619–622, Dec. 2013.

[Yur18] A. Yu. Yurin, N. O. Dorodnykh, O. A. Nikolaychuk, M. A. Grishenko. Designing rule-based expert systems with the aid of the model-driven development approach. *Expert Systems*, 35(5):1–23, June 2018.

[Cab09] M. E. Cabello, I. Ramos, A. Gomez, R. Limon. 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. IEEE Xplore Press, Dong Hoi, Vietnam*, 208–213, Apr. 2009.

[Can09] J. Canadas, J. Palma, S. Tunez. InSCo-Gen: A MDD Tool for Web Rule-Based Applications. *Web Engineering*, 5648:523–526, June 2009.

[Cha04] G. W. Chaur. Modeling rule-based systems with EMF. *Eclipse Corner articles. http://www.eclipse.org/articles/Article-Rule%20Modeling%20With%20EMF/article.html*, Nov. 2004.

[Dor18] N. O. Dorodnykh, A. Yu. Yurin. A domain-specific language for transformation models. *Proceedings of the 1st Scientific-practical Workshop Information Technologies: Algorithms, Models, Systems*, Irkutsk, Russia, 70–75, Sep. 2018.

[Dor17] N. O. Dorodnykh. Web-based software for automating development of knowledge bases on the basis of transformation of conceptual models. *Open Semantic Technolo-*

gies for Intelligent Systems, 7:145–150, Feb. 2017.

[Yur20] A. Yu. Yurin, N. O. Dorodnykh. Personal knowledge base designer: Software for expert systems prototyping. *SoftwareX*, 11:100411, Jan. 2020.

[Shi17] A. O. Shigarov, A. A. Mikhailov. Rule-based spreadsheet data transformation from arbitrary to relational tables. *Information Systems*, 71:123–136, Nov. 2017.

[Dor19] N. O. Dorodnykh, A. Yu. Yurin. *Technology for rule-based expert system engineering based on model transformations*. Novosibirsk, SB RAS, 2019. (in Russian).

[Ere16] A. P. Eremeev. Development of the apparatus of decision tables in the context of creating hybrid intelligent systems. *Proceedings of the 3rd All-Russian Pospelovsky Conference with International Participation – Hybrid and Synergetic Intelligent Systems*, 121–132, 2016. (in Russian).

[Ber07] A. F. Berman, O. .A. Nikolaychuk. Technical state space of unique mechanical systems. *Journal of Machinery Manufacture and Reliability*, 36(1):10–16, Jan. 2007.

[Ber93] A. F. Berman, V L. Khramova. Automated data base for failures in pipelines and tubular high-pressure apparatus. *Chemical and Petroleum Engineering*, 29(2):63–66, Feb. 1993.