End-user development of knowledge bases for semi-automated formation of task cards

N O Dorodnykh[1], Y V Kotlov[2], O A Nikolaychuk[1], V M Popov[2], A Y Yurin[1,2]

[1]Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences (ISDCT SB RAS), 134, Lermontov str., Irkutsk, 664033, Russia
[2]Moscow State Technical University of Civil Aviation, Irkutsk Branch (MSTUCA), 3, Kommunarov str., Irkutsk, 664003, Russia

iskander@icc.ru

Abstract. The complexity of creating artificial intelligence applications remains high. One of the factors that cause such complexity is the high qualification requirements for developers in the field of programming. Development complexity can be reduced by using methods and tools based on a paradigm known as End-user development. One of the problems that requires the application of the methods of this paradigm is the development of intelligent systems for supporting the search and troubleshooting onboard aircraft. Some tasks connected with this problem are identified, including the task of dynamic formation of task cards for troubleshooting in terms of forming a list of operations. This paper presents a solution to this problem based on some principles of End-user development: model-driven development, visual programming, and wizard form-filling. In particular, an extension of the Prototyping expert systems based on transformations technology, which implements the End-user development, is proposed in the context of the problem to be solved for Sukhoi Superjet aircraft. The main contribution of the work is as follows: expanded the main technology method by supporting event trees formalism (as a popular expert method for formalizing scenarios for the development of problem situations and their localization); created a domain-specific tool (namely, Extended event tree editor) for building standard and extended event trees, including for diagnostic tasks; developed a module for supporting transformations of XML-like event tree representation format for the knowledge base prototyping system – Personal knowledge base designer. A description of the proposed extension and the means of its implementation, as well as an illustrative example, are provided.

1. Introduction

The complexity of creating applications of Artificial Intelligence (AI), including classic expert systems and intelligent decision support systems, remains quite high [1]. One of the reasons for this complexity is the high requirements for qualification in the field of programming for developers of such systems. Through the use of methods and tools based on the paradigm known as End-User Development (EUD), including End-User Programming (EUP) and End-User Software Engineering (EUSE) [1-3], these requirements can be lowered, as well as the risk of manual coding errors is minimized.

One of the problems that requires the use of these methods is the development of intelligent systems to support the search and troubleshooting onboard the aircraft. We defined some tasks connected with this problem, including the task of dynamic (in real-time) formation of task cards for troubleshooting, including the set and sequence of operations.

In this paper, we propose an approach to solving this problem for RRJ-95 (Sukhoi Superjet) aircraft, which implements some EUD principles: model transformations, visual programming, and wizard form-filling. The approach is based on the previously developed PESoT (Prototyping Expert Systems Based on Transformations) technology [4] and includes methods, languages, and software, some of which are discussed in more detail in [4-6]. In this work we extend this approach, taking into account the need to form task cards and support the popular expert method of describing scenarios in the form of event trees while maintaining the principle of successive transformations of information models with different degrees of abstraction.

The main contribution is the extension of the PESoT technology by:

- Supporting event trees as a way to formalize scenarios for the development of problems and their localization;
- Creating a domain-specific tool for building standard and extended event trees, including for diagnostic tasks;
- Creating a module for supporting transformations of the XML-like event tree representation format (EETE) for the knowledge base prototyping system – Personal Knowledge Base Designer (PKBD) [6].

A feature of the PESoT technology and its extension is the redefinition of the main stages of the standardized model-driven development approach. In particular, we proposed to: develop and use specialized languages for modeling logical rules and describing model transformations; expand the set of supported models and standards used in creating computation-independent models; develop and use a set of original software tools integrated both "by data" and through program interfaces.

The paper is organized as follows. Section 2 presents a state-of-art, Section 3 describes a background, including the main principles of the PESoT technology. Section 4 contains the extension proposed including a method, supporting software, and an illustrative example, while Section 5 presents some concluding remarks and future works.

2. State-Of-Art

2.1. End-User Development

End-User Development (EUD) can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend software artifacts [7].

There are quite a lot of different EUD techniques in this area that are used to implement software [2]: component-based, rule-based, programming by demonstration/example, spreadsheet-based, wizard-based, template-based, natural language processing, visual programming (including workflow and dataflow diagrams), model-based, etc. At the same time, the use of some of them in the field of artificial intelligence is quite promising.

2.2. EUD methodologies for the intelligence systems engineering

In the field of AI, the principles of EUD are implemented in several well-known methodologies, such as OSTIS (Open Semantic Technologies for Intelligent Systems) [8], IACPaaS [9], CAKE (Computer-Aided Knowledge Engineering Technique) [10], KADS (Knowledge Acquisition and Documentation Structuring), CommonKADS [11], MOKA (Methodology for Knowledge-Based Engineering Applications) [12], etc. The main element of these technologies is the use of visual and special textual languages, which are mainly designed for the general organization of the knowledge management and large-block modeling, low-level tasks related to model transformations and code (or software) generation weakly considered. In this connection, the OSTIS technology stands out, but it is quite

strongly focused on its modeling, code generation, and interpretation tools, poorly integrating with a variety of languages and standards for knowledge representation and conceptual modeling systems.

The model-driven development approach (some of the principles of which are also implemented in the methodologies considered) is the most promising from the point of view of applied intelligence system engineering with the use of the EUD concept.

2.3. Model-Driven Development

Model-Driven Engineering (MDE) or Model-Driven Development (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 (source) codes [5, 13, 14]. This approach enables programmers and non-programmers (depending on the implementation) to create software based on conceptual models.

The main MDD principles are the following:

- A model is an abstract description of a system (a process) by a formal language. As a rule, models are visualized with the aid of certain graphic notations and serialized (represented) in XML.
- A metamodel is a model of a formal language used to create models (a model of models).
- A four-layer metamodeling architecture is a concept that defines the different layers of abstraction (M0-M3), where the objects of reality are represented at the lowest level (M0), then a level of models (M1), a level of metamodels (M2) and a level of a meta-metamodel (M3).
- Model transformation is the automatic generation of a target model from a source model with the accordance of a set of transformation rules [13, 15]. In this case, each transformation rule describes the correspondence between the elements of source and target metamodels.

There are examples of successful use of MDE for the development of database applications (e.g., ECO, for Enterprise Core Objects), agent-oriented monitoring applications, decision support systems, embedded systems (software components) for the Internet, and intelligence systems [16-20].

Today, the main MDE implementations (initiatives) are the following: OMG Model Driven Architecture (MDA), Eclipse Modeling Framework (EMF), Model Integrated Computing (MIC), Microsoft Software Factories, JetBrains MPS.

MDA is the most standardized MDE that uses the following software standards: MOF, XMI, CWM, UML, and QVT.

So, we formalized MDE [5, 21]:

$$MDE = \left\langle MOF, UML, CIM, PIM, PSM, PDM, F_{CIM\text{-to-}PIM}, F_{PIM-to-PSM}, F_{PSM-to-CODE} \right\rangle,$$

where *MOF (Meta Object Facility*) is an abstract language for describing models (a metamodel description language); *UML (Unified Modelling Language)* is a unified modeling language; *CIM (Computation Independent Model)* is a model that hides any details of the implementation and processes of software, and describes only the software and environment requirements; *PIM (Platform Independent Model)* is a model that hides details of the software implementation that depend on the platform, and contains elements that do not change when the software interacts with any platform; *PSM (Platform Specific Model)* is a model of software that taking into account implementation details and processes, dependent on a specific platform; *PDM (Platform Description Model)* is a set of technical characteristics and descriptions of the technologies and interfaces that make up the platform; $F_{CIM-to-PIM} : CIM \to PIM$, $F_{PIM-to-PSM} : PIM \to PSM$, $F_{PSM-to-CODE} : PSM \to CODE$ are the rules for models transformations.

In this paper, MDE is considered and implemented from the EUD point of view.

2.4. Troubleshooting on board the aircraft

Despite the urgency of the problem of diagnostics of civil aircraft and the existence of automated

diagnostic systems such as AirBus AirNav for A319/A320/A321, A330, A340, and A380, the creation of such systems for Russian aircraft remains an unsolved task. It should be noted that research in this area is conducted and we can distinguish both fundamental solutions of the conceptual level [22, 23], which are not tied to a specific aircraft, and works aimed at digitizing technical documentation and its integration with the onboard maintenance system [24]. At the same time, there are no examples of using artificial intelligence methods to support the search, localization, and elimination of malfunctions, while in other domains they are used for solving similar tasks [25]. In this paper, we propose to apply the previously developed EUD PESoT technology to solve the problem of troubleshooting for RRJ-95 aircraft.

3. Background

We implemented the model-driven EUD principle in the form of the PESoT technology [4-6] that includes methods and tools for prototyping rule-based expert systems and decision-making software components for intelligent systems.

The following formalization of the proposed method is proposed:

$$MDE^{ES} = \left\langle MOF, L^{ES}, CIM^{ES}, PIM^{ES}, PSM^{ES}, PDM^{ES}, F^{ES}_{CIM-to-PIM}, F^{ES}_{PIM-to-PSM}, F^{ES}_{PSM-to-CODE} \right\rangle$$

where L^{ES} is a set of languages and formalisms used for modeling; in our case $L^{ES} = \{UML, CM, DT, CT, RVML\}$ where UML is a Unified Modelling Language; CM is a concept or mind maps formalism; DT is a formalism for the representation of decision tables; CT is a formalism for the representation of canonicalized tables; $RVML$ is a Rule Visual Modeling Language;

CIM^{ES} is a computation-independent model for PESoT, in our case, it is a domain model represented with the aid of L^{ES};

PIM^{ES} is a platform-independent model for PESoT, in our case this model represent logical rules in our notation RVML;

PSM^{ES} is a platform-specific model for PESoT, in our case this model takes into account the features of the programming language, we use RVML;

PDM^{ES} is a set of platform description models for PESoT, in our case $PDM^{ES} = \{CLIPS, DROOLS, PHP, PKBD\}$.

$F^{ES}_{CIM-to-PIM}, F^{ES}_{PIM-to-PSM}, F^{ES}_{PSM-to-CODE}$ are the rules for model transformations.

The process of creating prototypes of knowledge bases and expert systems is represented by the sequence of the following stages: building domain models, building platform-independent models, building platform-specific models, generating source codes and specifications, testing.

The main elements of PESoT technology are the following:

- a method for creating rule-based expert systems and knowledge bases based on the sequential transformations;
- a method for automated creating domain models as computational-independent models based on transformation, both conceptual models (using XMI, XTM) and spreadsheets of a specific structure (using CSV);
- a method for automated creating software components-converters for conceptual models transformations;
- a UML-based graphical notation for designing rule-based models – a Rule Visual Modeling Language (RVML);
- a domain-specific language for the description of transformations – a Transformation Model Representation Language (TMRL);
- Personal Knowledge Base Designer (PKBD) – a tool for creating knowledge-bases and expert systems;
- Knowledge Base Development System (KBDS) - a tool for creating model transformation software components;

A detailed description of the technology is given in [4-6, 26].

4. Proposed PESoT extensions

The extension of the PESoT technology is due to the peculiarities of the current domain problem, in particular, the need to form certain chains of operations on troubleshooting using the EUD concept. So, under this research, we made a new application of our technology, improved our methodology, and extend a set of software tools.

Let's consider these extensions in more detail.

4.1. Methodology

The PESoT technology involves the development of computation-independent models that describes the domain at the most abstract level (including the basic concepts and relationships) as the first stage in the creation of knowledge bases of expert systems. At the same time, it is proposed to use conceptual models and, in a particular case, ontologies, as the most popular technique of the Semantic Web.

However, in the course of solving the domain problem of forming task cards, there was a need to build some scenarios of operations for the localization and elimination of malfunctions. There are several ways to create such scenarios for their further implementation in expert systems:

- Description of the sequence of operations in the ontology within each "malfunction" concept by entering individual types of relationships, using ontological modeling tools (for example, Protégé), with subsequent manual or semi-automatic coding;
- Formation of the matrix of adjacency of operations by external means, followed by manual coding;
- Direct programming of constructs in a general-purpose language or a specialized knowledge representation language;
- Description of scenarios in the form of event trees, followed by transformation to code.

Using event trees is the most preferable from the point of view of domain specialists [27] who generally do not have programming skills, but are familiar with system-wide and specialized formalisms and notations, such as graphs, event trees, and failure trees.

In this connection, we proposed to extend this stage by addition dividing it into sub-stages: the formation of an ontology of aircraft malfunction diagnostics, and the formation of event trees describing the sequence of operations.

Thus, from a formal point of view, we: 1) extend L^{ES} due to the new element *ET*: $L^{ES} = \{UML, CM, ET, DT, CT, RVML\}$; 2) complement $CIM^{ES} = \{CIM_{OM}^{ES}, CIM_{ET}^{ES}\}$, where CIM_{OM}^{ES} is a computation-independent model in the form of an ontology, CIM_{ET}^{ES} is a computation-independent model in the form of anevent tree; 3) clarify $F_{CIM-to-PIM}^{ES} = \{F_{CIM_{OM}-to-CIM_{ET}}^{ES}, F_{CIM_{ET}-to-PIM}^{ES}\}$.

The further chain of transformations remains unchanged.

Figure 1 presents the PESoT technology and its extensions from the point of view of a four-layer metamodeling architecture.

4.2. Software

A set of tools has been developed to support the proposed extension: a specialized event trees editor [28], called Extended Event Tree Editor (EETE), and an additional module, namely PKBD.ExTree, for Personal Knowledge Base Designer (PKBD) [5] to integrate the new software into the PESoT technological circuit.

Extended Event Tree Editor (EETE) is a web-based graphical editor for visual modeling (design) of semantic tree structures in the form of event trees. The editor implements an extended model of event trees, including a "Mechanism" event and thematic (semantic) layers of the tree, as well as a visual graph-like notation for their representation. The editor is aimed at non-programming users (domain experts, analysts, etc.). The main functions of the editor are managing event tree diagrams (creating, opening, closing, deleting); adding, editing, and deleting elements of the event tree, including tree

levels, events, and mechanisms; adding, editing, and deleting parameters of events and mechanisms; exporting created event tree diagrams as XML documents, importing ontological models in OWL format.

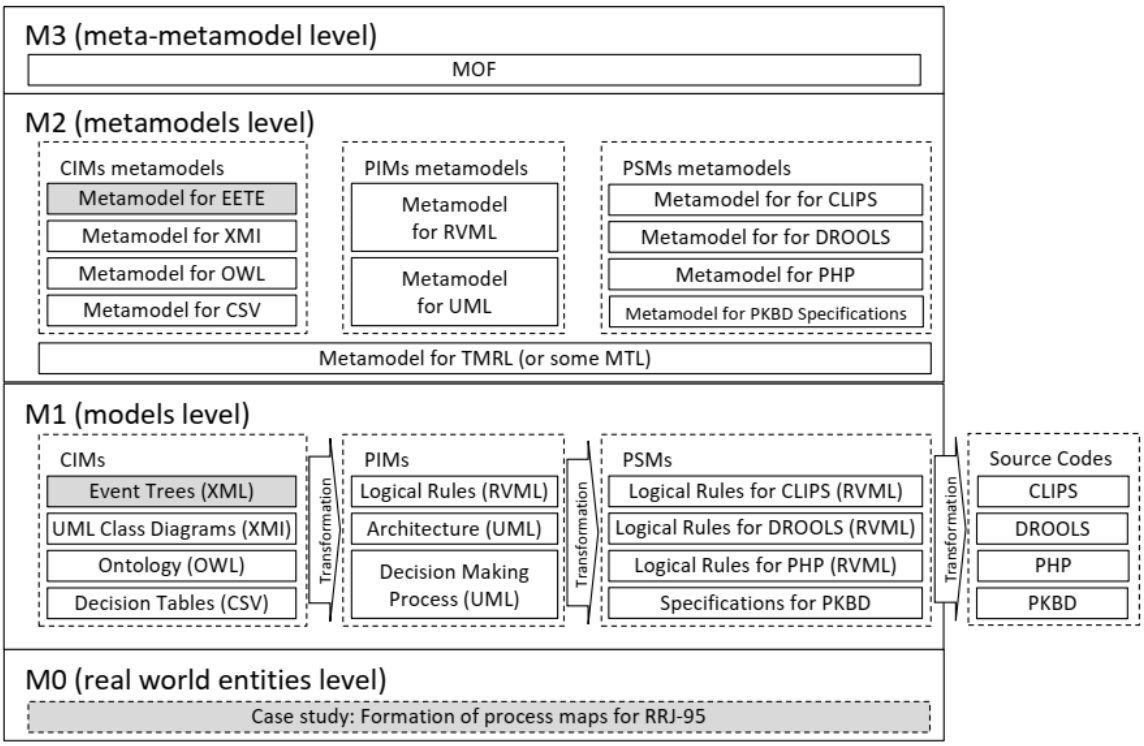
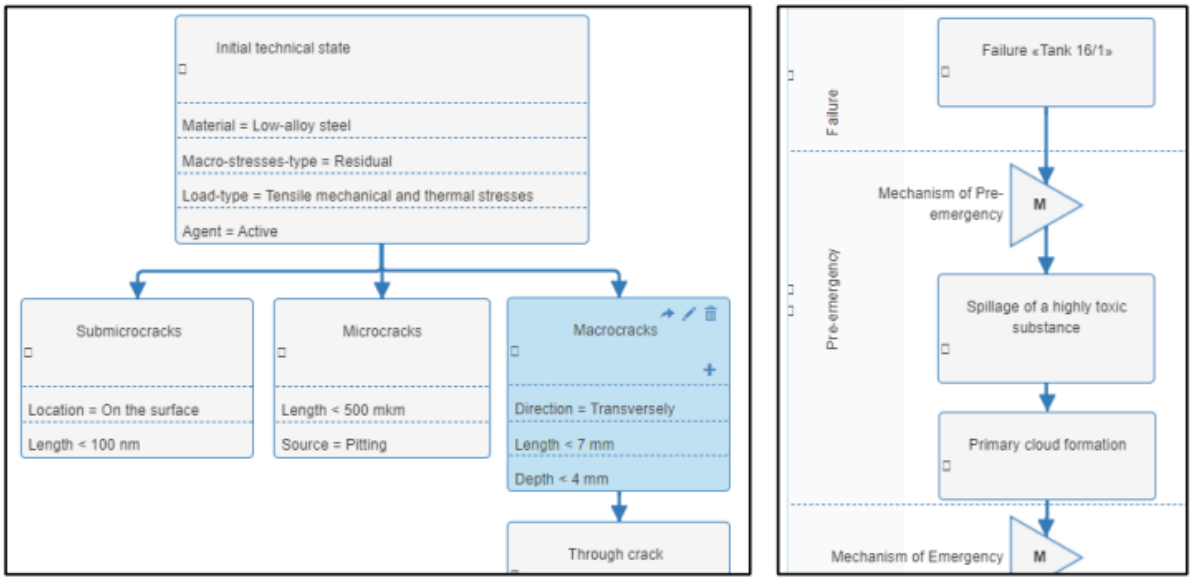


Figure 1. The PESoT technology and its extensions form the point of view of a four-layer metamodeling architecture.



1) 2)

Figure 2. An example of EETE GUI: 1) a standard event tree; 2) an extended event tree.

When importing ontological models, the OWL and EETE formats are compared, in particular, the following basic elements are matched and converted: owl:Ontology → Diagram; owl:Class → Event; owl:ObjectProperty → Operator; owl:DatatypeProperty → Parameter; owl:DatatypeProperty / rdfs:range → Parameter (value) et al.

PKBD.ExTree is an additional module for PKBD, which provides integration of EETE into the PESoT technology. PKBD is one of the technology tools designed to support transformations of computation-independent models into platform-independent models, in particular, implementing the formalism of logical rules. PKBD has a modular architecture (figure 3) that provides the ability to add modules (dynamic link libraries) that generate source codes and integration with domain model designers. Currently, CLIPS, Drools, PHP, IBM Rational Rose, StarUML, XMind, CMapTools, and Microsoft Excel support DLLs are included.

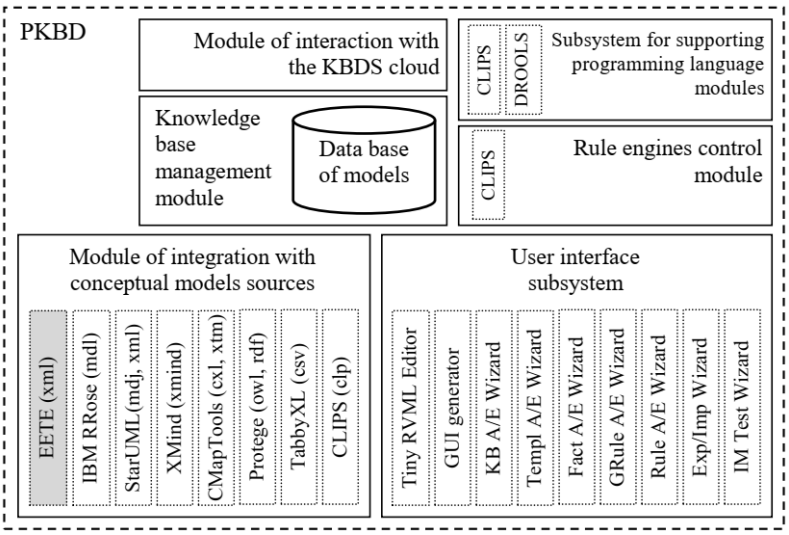


Figure 3. The PKBD architecture [5] with a new module.

To support the *EETE* format transformation a new dynamic link library (xml.eete.i.dll) was created with the aid of Object Pascal. The main purpose of the library is unambiguous mapping of EETE constructs to the PKBD format, in particular, the correspondence and transformation of the following main elements are established: Diagram → KowledgeBase; Event → Template; Operator → Condition; Parameter → Slot; Parameter (value) → Slot (value) et al.

The input of the library is an XML-like description of event tree elements. This description is processed by the `Execute` function. Besides, for automated recognition and linking this library when starting PKBD, the following functions returning its brief description were added: `DllInfo` and `About`.

4.3. An Example

Let's consider an example of creating a knowledge base fragment for diagnosing and troubleshooting the RRJ-95 power supply system.

According to the PESoT technology, the first stage is creating domain models. To automate this stage, previously developed models can be used, but in the current research, the "Troubleshooting the Aircraft Power Supply System" ontology was developed, this ontology includes 335 classes and 1066 axioms. For its development, analysis of the following documentation was carried out: the technical operation manual and the aircraft troubleshooting manual (section 24). The first manual describes the structure and functionality of the aircraft, the second one contains possible failures and malfunctions, the sequence of operations to identify and eliminate them.

The ontology was developed using Protégé.

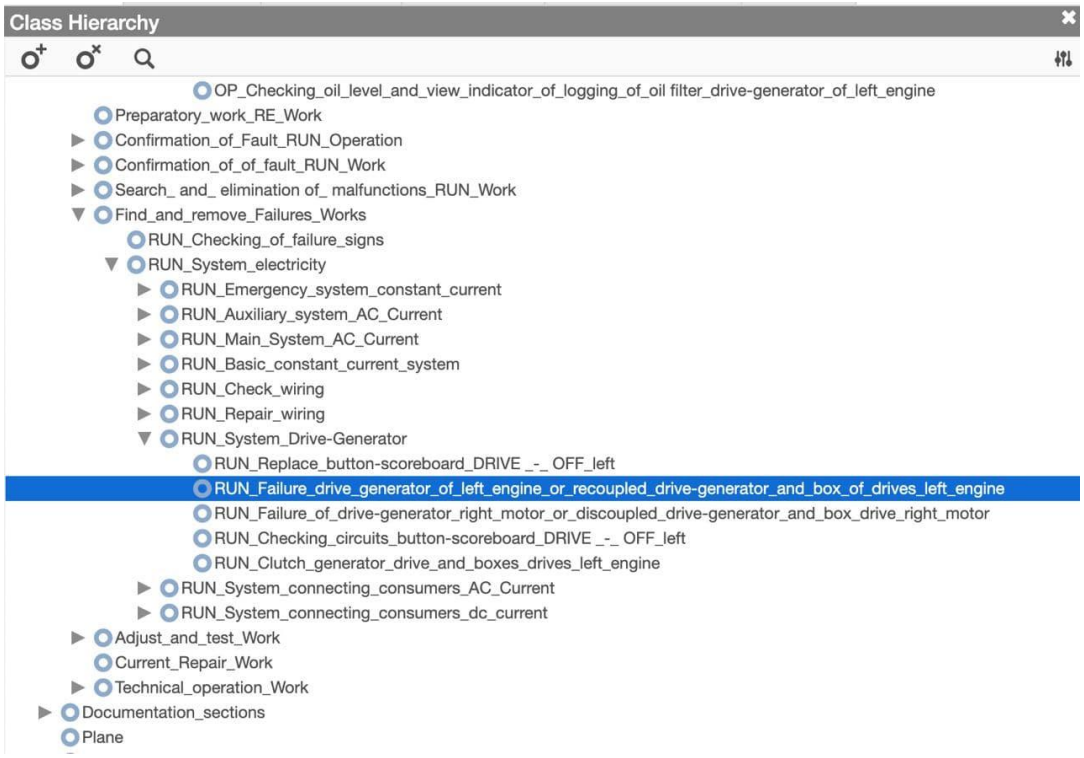


Figure 4. Fragments of the ontology: Class Hierarchy.

Further, a fragment of the obtained ontology was imported into the EETE (figure 5) following the proposed extension (implementing the $F^{ES}_{CIM_{OM}-to-CIM_{ET}}$ transformation). In EETE, cause-and-effect relationships were defined and conditions were clarified.

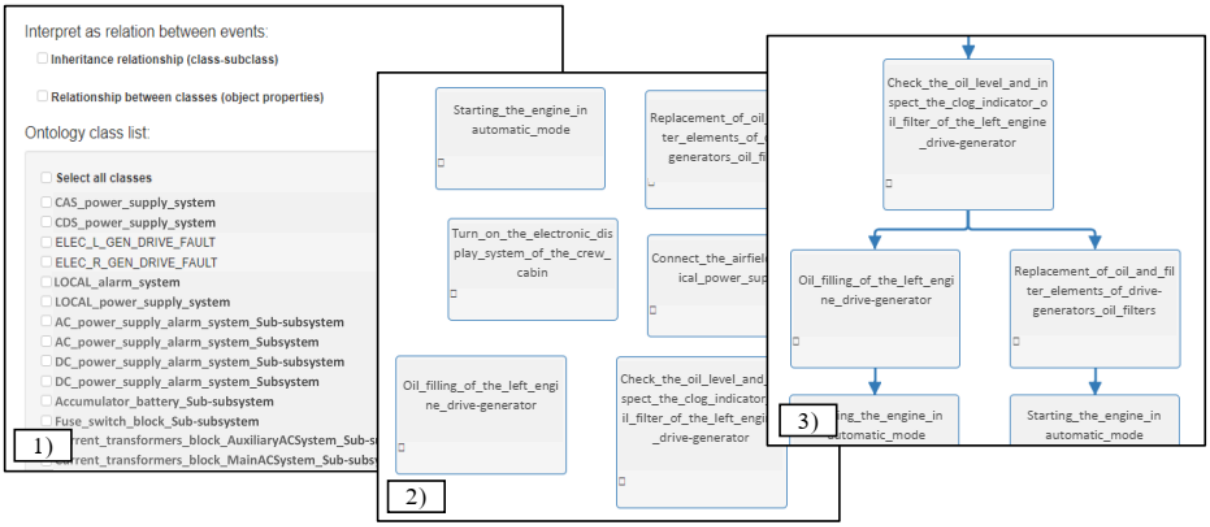


Figure 5. Importing an ontology into the EETE: 1) The selection of concepts and the way of interpreting relationships; 2) Imported concepts transformed into events; 3) A clarified event tree.

In this example, an external sign of a malfunction is the message on the display of engine parameters and emergency messages "ELEC APU GEN FAULT" (the APU GEN symbol is yellow), possible causes: the current transformer unit (TBD); the control unit for the APU generator and ground

power (4-X242); the APU generator (1-X242); the APU GEN display button (power supply panel); the central part of the ceiling panel (5-F311); electrical wiring.

Preparatory operations: 1) Connect the airfield electrical power supply; 2) Turn on the electronic display system of the crew cabin; 3) Check the oil level and inspect the clog indicator oil filter of the left engine drive-generator; 4) ….

Malfunction confirmation operations: 1) Start the APU; 2) Press the APU GEN display button if it is pressed; 3) ....

Troubleshooting operations: If there are signs of a malfunction after starting the APU, then: 1) Turn off the APU generator; 2) Replace the control unit with the APU generator....

Confirmation of troubleshooting operations: 1) Perform a fault confirmation; 2) Make sure that there are no signs of a malfunction.

Final operations: 1) Turn off the APU generator; 2) ....

The next stage of the proposed extended technology: the formation of platform-independent models in the form of rule-based models. At this stage, the $F^{ES}_{CIM_{ET}-to-PIM}$ transformation is being implemented (figure 6) by importing the clarified tree into PKBD (using the PKBD.ExTree module).

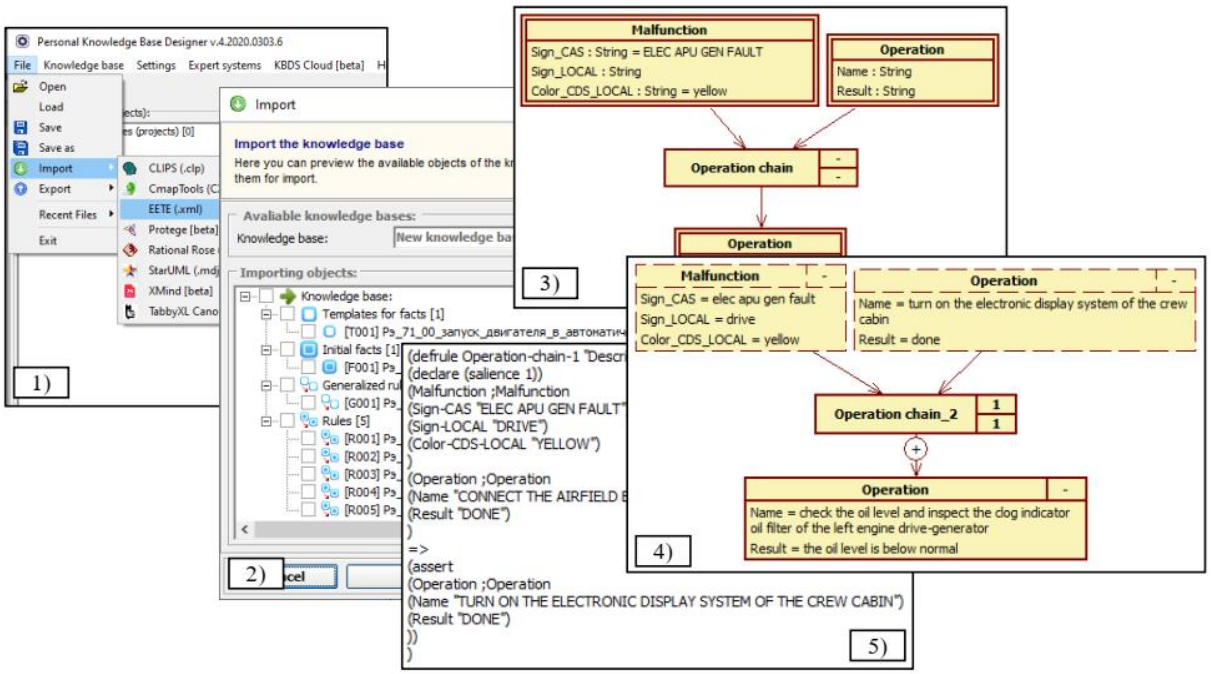


Figure 6. Examples of PKBD: 1) Call the PKBD.ExTree to import trees; 2) Preview of importing elements; 3) A rule template for describing the sequence of operations; 4) A specific rule formed based on the template; 5) Generated code on CLIPS for a specific rule.

Next, rule-based models are modified, and they are tested and debugged following the standard chain of PESoT stages (building a platform-dependent model, code generation, and testing).

5. Discussion

A direct quantitative comparison of the proposed technique with the methodologies discussed in section 2 is difficult for the following reasons: (a) the available other tools have some limitations (private access and the need for modifications by programmers to set up for the task under consideration); (b) the ontology developed by the authors is too large (335 classes and 1066 axioms) for the manual transfer (retyping) to data storages of the available other tools; (c) the means of the

automated conversion of ontologies for the available other tools are absent. However, a qualitative comparison is possible (table 1).

Table 1. The qualitative comparison of some EUD methodologies for the intelligence systems engineering and the PESoT technology.

Criteria / Methodology	Purposes of a methodology	Implemented EUD techniques	Supporting tools	Supporting target languages	Supporting modelling languages	Integration capability
OSTIS [8]	knowledge engineering, code generation, software development	component-based, visual programming, model-based	IMS.OSTIS METASYSTEM	SC*	SC*	average
IACPaaS [9]	knowledge engineering, specification interpretation, software development	component-based, model-based	IACPaaS cloud platform	-	a model description language, hierarchical binary digraphs	-
CAKE [10]	large-block modeling	visual programming, model-based	-	-	extended UML package diagrams	-
CommonKADS [11]	knowledge engineering, large-block modeling	visual programming, model-based	SWI-Prolog …	Prolog …	UML, trees, CML**	average
MOKA [12]	knowledge engineering, large-block modeling	visual programming, model-based, wizard-based, template-based	-	-	MML***	-
PESoT [4-6]	knowledge engineering, code generation, specification interpretation, software development	visual programming, model-based, wizard-based	PKBD, KBDS	CLIPS, DROOLS, PHP	UML, RVML, mindmaps, trees, ontologies	high

*SC – Semantic Code, including: SCg (Semantic Code graphical), SCn (Semantic Code natural) and SCs (Semantic Code string)

**CML – The CommonKADS (textual) conceptual modelling language

***MML – MOKA Modelling Language, an adaptation of UML

The following features (advantages) of the PESoT technology can be defined based on table 1:

- The focus on the software engineering, rather than on the conceptualization and formalization of knowledge;
- The availability of software tools in the public access and ready to the use;
- The focus on widely used programming languages for knowledge bases;

- The support for both specialized conceptual models (RVML, trees) and system-wide models (UML, ontologies, concept and mind maps);
- The possibility of integration "by the data" with other software, and the creation of alienable software modules.

In addition to a formal qualitative comparison with other EUD methodologies for intelligence systems engineering, a comparison of the PESoT technology before and after its extension when solving the task of creating a knowledge base fragment for diagnosing and troubleshooting the RRJ-95 power supply system was made (table 2).

Table 2. The qualitative comparison of the PESoT technology before and after its extension.

Technology/ Criteria	PESoT	PESoT (after the extension)
The main actions	(a) import of an ontological model into PKBD with the transformation of classes and axioms into templates of facts and rules; (b) refinement of the obtained elements; (c) validation by experts; (d) generation of code.	(a) import of the ontological model to EETE with the transformation of classes and axioms into events; (b) refinement of events to form scenarios; (c) validation by experts; (d) import of the scenarios to PKBD with the transformation of events into templates of facts and rules; (e) refinement of the obtained elements; (f) generation of code.
Shortcomings	(a) fragmentary description of fragments of the task card in the form of separate logical rules without the possibility of visual representation of a chain of operations; (b) the high complexity of validation by domain specialists of the developed knowledge base.	(a) additional actions related to the formation of scenarios.
Advantages	(a) fewer actions.	(a) the semantically correct representation of the task cards in the form of scenarios; (b) the use of the domain-specific formalism (event trees) that is familiar for domain specialists; (c) the possibility of validation of the chain of operations by specialists at the stage of scenario formation.

So, we can conclude the following:

- The proposed technology extension contains additional actions that can increase the total time for creating a prototype of the knowledge base.
- The new actions provide a more complete involvement of domain specialists in the development process since they allow them to use the familiar formalism of event trees.
- The use of trees provides a visual and semantically correct representation of the chains of operations that form the task cards, as well as their validation.

Thus, it can be argued that a possible increase in development time due to the introduction of new actions will be compensated by a reduction in the number of meaningful errors in the formation of chains of operations. This statement is planned to be verified in the future by conducting computational experiments on limited fragments of the developed ontology.

6. Conclusion and Future Works

The paper proposes an extension of the EUD PESoT technology in the context of developing a knowledge base prototype of an intelligent system for supporting troubleshooting onboard the RRJ-95 aircraft. The developed knowledge base should provide the dynamic formation of the task card for troubleshooting in terms of creating a list of operations.

The extended technology implements some principles of the EUD: model-oriented development, visual programming, and wizard form-filling.

The main contributions of the paper are the following: the extension of the main PESoT method by supporting event trees formalism, as a popular expert method for formalizing scenarios for the development of problems and their localization; creating a domain-specific tool (Extended Event Tree Editor) for building standard and extended event trees; developing a module for supporting transformations of XML-like event tree representation format (EETE) for PKBD. An illustrative example demonstrating the principal applicability of the approach is given.

The technology is designed for end-users and reduces the time for creating prototypes of AI modules and expert systems by automating the codification stage and using existing domain models.

At the moment, the ontology is being expanded and the software tools are being improved. In particular, the following shortcomings of the proposed approach were identified:

- Ambiguity in the interpretation of relationships between concepts and instances of the ontology when they are transformed into elements of event trees. To solve this problem, the tools will be refined and experiments will be conducted.
- The need to improve EETE in terms of more a user-friendly setting and representation of tree branching conditions.
- Redundancy (bulkiness) of event trees in some situations, because when building operation sequences, it became necessary to replicate the same events. This problem could be solved by using cyclic graphs. In this regard, in the future, it is planned to use concept maps or activity diagrams, as well as implementing their tools (in particular, CmapTools and StarUML) to solve the current problem.
- The need to improve the PKBD.ExTree module in terms of aggregating semantically similar imported events and interpreting the relationships between them. To solve this problem, experiments will be conducted and heuristics will be formulated.

Our approach has a certain level of universality and after its improvement can be used to form knowledge bases that codify scenarios for the development of events in various domains, for example, the sequence of work during the industrial safety inspection [29].

In the future, we plan to improve the EETE and to use other formalisms when describing scenarios that provide overcoming the identified shortcomings, for example, semantic graph structures in the form of UML activity diagrams. It is also planned to make a quantitative evaluation of the proposed technology extension by conducting computational experiments.

7. Acknowledgments

The present study was supported by the Ministry of Education and Science of the Russian Federation (Project no. 121030500071-2 "Methods and technologies of a cloud-based service-oriented platform for collecting, storing and processing large volumes of multi-format interdisciplinary data and knowledge based upon the use of artificial intelligence, model-driven approach and machine learning").

References

[1] Coronado E, Mastrogiovanni F, Indurkhya B and Venture G 2020 Visual Programming Environments for End-User Development of Intelligent and Social Robots, a Systematic Review *Journal of Computer Languages* **58** 100970

[2] Barricelli B R, Cassano F, Fogli D and Piccinno A 2019 End-user development, end-user programming and end-user software engineering: A systematic mapping study *Journal of Systems and Software* **149** 101-137
[3] Santos M and Villela M L B 2019 Characterizing end-user development solutions: A systematic literature review *International Conference on Human-Computer Interaction* 194-209
[4] Yurin A Y 2020 Technology for Prototyping Expert Systems Based on Transformations (PESoT): A Method CEUR Workshop Proceedings **2677** 36-50
[5] Yurin A Y, Dorodnykh N O, Nikolaychuk O A and Grishenko M A 2018 Designing rule-based expert systems with the aid of the model-driven development approach *Expert Systems* **35(5)** 12291
[6] Yurin A Y and Dorodnykh N O 2020 Personal knowledge base designer: Software for expert systems prototyping *SoftwareX* **11** 100411
[7] Lieberman H, Paternò F, Klann M and Wulf V 2006 End-User Development: An Emerging Paradigm *Human-Computer Interaction Series* **9** 1-7
[8] Golenkov V, Shunkevich D, Davydenko I and Grakova N 2019 Principles of organization and automation of the semantic computer systems development *Open Semantic Technologies for Intelligent Systems* 53-90
[9] Gribova V, Kleschev A, Moskalenko P, Timchenko V, Fedorischev L and Shalfeeva E 2019 The methods and the IACPaaS Platform tools for semantic representation of knowledge and development of declarative components for intelligent systems *Open Semantic Technologies for Intelligent Systems* 21-24
[10] https://www.ida.liu.se/divisions/aiics/publications/ECAI-2002-CAKE-Computer-Aided.pdf
[11] Surakratanasakul B 2017 Lightweight CommonKADS in knowledge intensive organization *9th International Conference on Information Technology and Electrical Engineering (ICITEE)* 1-5
[12] Stokes M 2001 Managing Engineering Knowledge – MOKA: Methodology for Knowledge Based Engineering Applications *Professional Engineering Publishing Limited*
[13] Da Silva A R 2015 Model-driven engineering: A survey supported by the unified conceptual model *Computer Languages, Systems & Structures* **43** 139-155
[14] Cretu L G and Florin D 2014 Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice *Apple Academic Press*
[15] Mens T and Gorp P V 2006 A Taxonomy of Model Transformations *Electronic Notes in Theoretical Computer Science* **152** 125-142
[16] Nofal M A and Fouad K M 2015 Developing web-based Semantic and fuzzy expert systems using proposed tool *International Journal of Computer Applications* **112(7)** 38-45
[17] Cabello M E, Ramos I, Gomez A and Limon R 2009 Baseline-oriented modeling: An MDA approach based on software product lines for the expert systems development *Processing of the First Asian Conference on Intelligent Information and Database Systems* 208-213
[18] Canadas J, Palma J and Tunez S 2011 Defining the semantic of rule-based web applications through model-driven development *International Journal of Applied Mathematics and Computer Science* **21(1)** 41-55
[19] Dunstun N 2012 A Hybrid Architecture for Web-based Expert Systems *International Journal of Artificial Intelligence and Expert Systems* **3 (4)** 70-79
[20] Ruiz-Mezcua B, Garcia-Crespo A, Lopez-Cuadrado J and Gonzalez-Carrasco I 2011 An expert system development tool for non AI experts *Expert Systems with Applications* **38** 597-609
[21] Dorodnykh N O, Yurin A Y and Stolbov A B 2018 Ontology Driven Development of Rule-Based Expert Systems *Proceedings of the 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)* 1-6
[22] Perfiliev O V, Ryzhakov S and Dolzhikov V A 2018 *Proceedings of the Samara Scientific Center of the Russian Academy of Sciences* **4(3)** 326-331 (in Russian)
[23] Makarov N N 2008 *News of universities: Aviation Technology* **1** 46-50 (in Russian)
[24] Savvina A M 2019 *Crede Experto: transport, society, education, language* **3(22)** 27-35

[25] Yanchenko A Y 2020 *Azimut of scientific research: economics and management* **3(32)** 413-416 (in Russian)
[26] Dorodnykh N O and Yurin A Y 2021 An RVML extension for modeling fuzzy rule bases *CEUR Workshop Proceedings* **2858** 34-45
[27] Berman A F, Dorodnykh N O, Nikolaychuk O A and Yurin A Y 2019 Event trees transformation for rule bases engineering *Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* 1138-1143
[28] Extended Event Tree Editor, http://eet-editor.knowledge-core.ru/
[29] Berman A F, Nikolaichuk O A, Yurin A Y and Kuznetsov K A 2015 *Chemical and Petroleum Engineering* **50(1-2)** 730-738