Towards designing knowledge bases for aircraft malfunctions diagnostics based on model transformations

Y V Kotlov¹, V M Popov¹ and A Y Yurin¹

¹Moscow State Technical University of Civil Aviation, Irkutsk Branch (MSTUCA), 3, Kommunarov str., Irkutsk, 664003, Russia

E-mail: iskander@icc.ru

Abstract. The paper describes a method of forming prototypes of knowledge bases for the interactive electronic technical manual for RRJ-95 troubleshooting. The method includes three main stages: the formation of an ontology for diagnosing an aircraft malfunction; the formation of event trees that describe the sequence of operations to eliminate specific malfunctions; the construction of knowledge bases that codify the basic concepts of the ontology and trees formed. At each stage, the following software are used: Protégé, Extended Event Tree Editor (EETE), and Personal Knowledge Base Designer (PKBD). Data-driven integration of software is implemented through model transformations. The description of the main stages of the method; the table of correspondences used in the implementation of model transformations, as well as an illustrative example are presented.

1. Introduction

Diagnostics of aircraft malfunctions continue to be an urgent problem that requires the use of semantic Web technologies (in particular, ontologies [1]) and artificial intelligence methods in the form of rule-based expert systems [2]. These technologies provide a formalized representation of domain knowledge, but also decision making through the use of deductive and inductive inference.

In this paper, we consider a method for solving one of the tasks of creating an interactive electronic technical manual for RRJ-95 troubleshooting. This task related to the formation of knowledge bases of scenarios for the localization and elimination of malfunctions. Such scenarios are sequences of operations that form a task card. There are several ways to generate such scenarios for their further implementation in expert systems:

- description of the sequence of operations in the ontology within each malfunction 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 [3] 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 propose a method that provides the description of scenarios of operations and the synthesis of source codes. The proposed method contains three main stages: the formation of an ontology of aircraft malfunction diagnostics; the formation of event trees describing the sequence of operations to eliminate specific malfunctions; the construction of knowledge bases with logical rules. At each of the stages, certain software tools are used: Protégé, Extended Event Tree Editor (EETE) and Personal Knowledge Base Designer (PKBD) [4]. The interaction of the tools is implemented based on “on data” integration through model transformations.

2. Related Works
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 civil 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 [5, 6], which are not tied to a specific aircraft, and works aimed at digitizing technical documentation and its integration with the onboard maintenance system [7]. 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 [8].

Among the popular methods of solving the problem under investigation, we can select ontologies, as a way of conceptualizing knowledge [1, 9] and rule-based expert systems [2, 10], which still attract experts with their visual simplicity of knowledge representation and transparency of decisions made.

3. Proposed Method
The task of creating knowledge bases for localization and troubleshooting scenarios is solved within the classical paradigm of expert systems engineering [2], which includes such processes as identification, conceptualization, formalization, and codification. At the same time, the proposed method includes three main stages (Figure 1):

- formation of the ontology of aircraft malfunction diagnostics;
- formation of event trees describing the sequence of operations to eliminate specific malfunctions;
- building knowledge bases containing logical rules.

Each stage is implemented with the following integrated software: Protégé, Extended Event Tree Editor (EETE) and Personal Knowledge Base Designer (PKBD).

Let's describe the stages in detail.

**Formation of the ontology.** To form the ontology, the following documentation was analyzed: the technical operation manual and the aircraft troubleshooting manual for RRJ-95. The first manual describes the structure and functionality of the aircraft, the second one lists possible failures and malfunctions, the sequence of operations to identify and eliminate them. An analysis of the manuals on the electro-supply system was carried out (section 24 of manuals) as part of our method testing.

We used Protégé to perform conceptualization and to define the basic concepts and relationships. At the same time, the ontology of aircraft malfunction diagnostics contains 355 classes and 1066 axioms and includes the ontology of technical objects, the ontology of operations (Figure 2), which is divided into the ontologies of preparatory operations, maintenance operations, and troubleshooting operations, etc.
Figure 2. A fragment of the ontology of operations.

**Formation of event trees.** The purpose of using event trees is to build scenarios or operation sequences. As practice has shown, solving this problem using Protégé is quite cumbersome and time-consuming. In this regard, it was decided to provide the transformation of the ontology of operations to the Extended Event Tree Editor (EETE) format. EETE is specialized software for building classic event and failure trees, as well as an extended version of them using the concept of "Mechanism" and thematic levels [3, 11]. As a result of the transformation, a set of available events (operations and activities) is formed in the EETE. An important aspect of the successful construction of such scenarios is the presence of the "Result" property in the description of the operations, which provides their semantic "gluing". The result of this stage is the formation of tree structures that describe task cards for certain aircraft malfunctions.

**Building knowledge bases.** The purpose of this stage is to build knowledge bases. At the moment, there are practically no tools for generating knowledge base codes from event tree-like semantic structures. In this regard, the EETE format was transformed into the format of the Personal Knowledge Base Designer (PKBD). PKBD provided refinement of the resulting structures in the form of logical rules, as well as their testing in the built-in interpreter. Based on the results of debugging, code generation is possible in CLIPS (C Language Integrated Production System) [12], DROOLS [13], or PHP (Hypertext Preprocessor) for further integration and use.

All transformations are performed under the concept of model transformations of the model-driven
approach [14]. To implement such transformations, both specialized languages (for example, ATL, Epsilon, TMRL [15]) and general-purpose languages are used. In this paper, transformations are implemented using the general-purpose languages PHP and Object Pascal. Some correspondences of transformed elements in a simplified form were represented in Table 1.

Table 1. Main correspondences between elements of OWL, EETE, and PKBD

| OWL                      | EETE          | PKBD          |
|--------------------------|---------------|---------------|
| owl:Ontology             | Diagram       | KnowledgeBase |
| owl:Class                | Event         | Template      |
| owl:ObjectProperty       | Operator      |               |
| owl:ObjectProperty / rdfs:domain | Operator (lhs) | Condition     |
| owl:ObjectProperty / rdfs:range | Operator (rhs) | Action        |
| owl:DatatypeProperty     | Parameter     | Slot          |
| owl:DatatypeProperty / rdfs:domain | Parameter (class) | Slot (name)   |
| owl:DatatypeProperty / rdfs:range | Parameter (value) | Slot (value)  |

4. An Example

As an example, let’s consider building a fragment of the knowledge base for a sequence of "ELEC APU GEN FAULT" troubleshooting operations.

Formation of the ontology. So, as noted above, based on the analysis of documentation, a hierarchical ontology was formed, including information about malfunctions, their manifestations (signs), and operations to eliminate them.

In our 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.

Malfunction confirmation operations: 1) start the APU; 2) press the APU GEN display button (if it is not pressed); 3) ... and so on.

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... and so on.

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

Final works: 1) turn off the APU generator; 2) .... and so on.

All these operations form the ontology classes.

Formation of event trees. Using the import capabilities, information about the operations is transferred to the EETE, which allows one to form their sequences (Figure 3).

Building knowledge bases. Using the import capabilities, information about operations and their relationships is transferred to the PKBD, creating templates for the "Operation" and "Malfunction" facts, a template for a rule describing the operation relationships, and specific rules describing the concrete relationships between operations. After correcting the imported information, the obtained fragment of the knowledge base was tested and the source code (Figure 4) was generated, which can be supplemented if necessary.
5. Conclusion

The paper describes a method for creating prototypes of knowledge bases for the interactive electronic technical manual for RRJ-95 malfunctions troubleshooting. The method allows one to conceptualize the domain, form troubleshooting scenarios, and generate source code in a specific knowledge representation language. Each of the considered tasks is implemented by a specific software tool: Protégé, Extended Event Tree Editor (EETE), and Personal Knowledge Base Designer (PKBD), which are integrated "by the data" through the use of the principles of model transformations.

An example illustrates the method proposed and shows its fundamental applicability. Our approach has a certain level of universality and can be used to form knowledge bases that codify scenarios of events in various domains, for example, the sequence of operations during the industrial safety inspection procedure [10].

Figure 3. A fragment of an event tree for malfunction troubleshooting and the corresponding XML-like code (EETE)

Figure 4. Examples of a template for rules, a specific rule (in the RVML notation [16]), and the corresponding generated CLIPS code (PKBD)
References

[1] Guarino N 1998 First International Conference on Formal Ontology in Information Systems (FOIS’98) 46 3-15
[2] Wagner W P 2017 Expert Systems with Applications 76 85-96
[3] Berman A F, Dorodnykh N O, Nikolaychuk O A and Yurin A Y 2019 Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) 1138-1143
[4] Yurin A Y and Dorodnykh N O 2020 SoftwareX 11, 100411
[5]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
[6] Makarov N N 2008 News of universities: Aviation Technology 1 46-50
[7] Savvina A M 2019 Crede Experto: transport, society, education, language 3(22) 27-35
[8] Yanchenko A Y 2020 Azimut of scientific research: economics and management 3(32) 413-416
[9] Gribova V V, Petryaeva M V, Okun D B and Shalfeeva E A 2018 Ontology of designing 8(1) 58-73
[10] Berman A F, Nikolaichuk O A, Yurin A Y and Kuznetsov K A 2015 Chemical and Petroleum Engineering 50(1-2) 730-738
[11] Berman A F, Dorodnykh N O, Nikolaychuk O A and Yurin A Y 2018 CEUR Workshop Proceedings. Information Technologies: Algorithms, Models, Systems (ITAMS 2018) 2221 25-31
[12] CLIPS: A Tool for Building Expert Systems, http://clipsrules.sourceforge.net/
[13] Drools https://www.drools.org/
[14] da Silva A R 2015 Computer Languages, Systems & Structures 43 139-155
[15] Dorodnykh N O and Yurin A Y 2018 CEUR Workshop Proceedings of Information Technologies: Algorithms, Models, Systems (ITAMS’18) 2221 70–75
[16] Dorodnykh N O and Yurin A Y 2021 CEUR Workshop Proceedings of the 1st International Workshop on Advanced Information and Computation Technologies and Systems 2858 34-45