A CONCEPTION OF EXPERT SYSTEM FOR MIG-29 AIRCRAFT

Received 16th June; accepted 12th July.

Krzysztof Butlewski, Piotr Golański

Abstract:

This paper presents the concept of the MiG-29 aircraft-dedicated diagnostic expert system. Subsequent stage of evaluation expert system from knowledge acquisition to software implementation has been described. As implementation tools a combination of expert systems shells CLIPS and LabWindows data acquisition libraries has been chosen. The whole system has been integrated in BorlandC++ environment.

Keywords: *diagnostics, expert systems, expert systems shells, CLIPS, Mig-29 aircraft*

1. Introduction

This issue is concerned on the problems of exploit new information technologies for creation computer diagnostics system software for some type of aircrafts exploit in the polish Air Force (Fig.1). The Computer diagnostic system likewise any information system [1], can be created with the aid of various languages and various developing environments. Most often, it takes advantage of imperative programming language like C or Pascal. However, since many years in literature citation, you can find an example of adaptation the expert systems shells for creation a diagnostic software [3, 8, 11, 12]. Expert systems shells are founded on syntax and semantics of declarative language like Lisp [7] or Prolog [4]. The essential difference between expert system and traditional problem solving system inheres in coding method. In traditional approach this method is coded in the form of algorithm and corresponding data structures. In expert system's approach the solving method is coded in data structures exclusively. This approach will be presented here in context of aircraft diagnostics system.

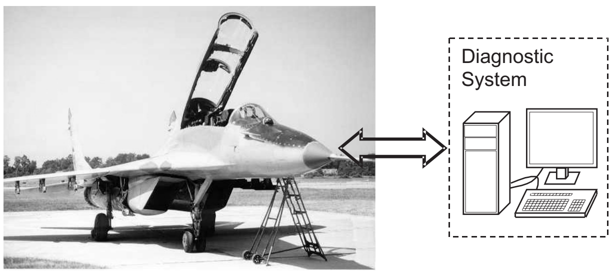

Fig. 1. Aircraft's computer aided diagnostics.

In the next paragraphs subsequent stage of evaluation expert system from knowledge acquisition, across diagnostic model definition and tools specifications to software implementation will be presented.

2. Problem Analysis

Architecture of typical expert system is presented in fig. 2. This architecture consists of following components:

- inference engine,
- knowledge base,
- user interface,
- working storage,
- hardware interface.

A knowledge base is an implementation of knowledge of domain expert, created by a knowledge engineer. An inference engine acts on knowledge base and generates a solution of problem, defining by user. User interacts with the system using the User Interface (usually Graphical User Interface - GUI). To the structure proposed in [9] is added a Hardware Interface, necessary for exchange data between the system and on board devices of an aircraft.

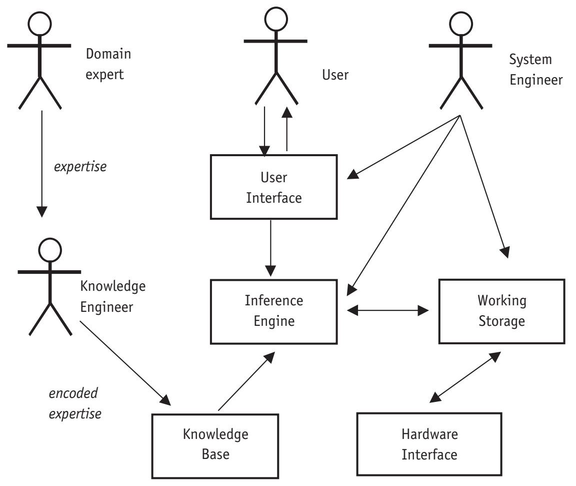

Fig. 2. Architecture of aircraft's diagnostic expert system.

The first step for the expert system creation is a knowledge acquisition in the particular domain of knowledge. In literature citation we can find different methods of knowledge acquisition. For example in [2] authors presents a diagnostic expert system, that knowledge acquisition has been founded on measurement date from the experiment, executed on the object, for which the system has been dedicated.

In our case, three sources of domain knowledge exist:

1) the Russian technical documentation of diagnostic device currently applied for aircraft,
2) the result of verification of the above documentation,
3) knowledge of the diagnostic device operators.

Founded on this knowledge a diagnostics model has been defined [6]:

$$M=(Q, \Sigma, \delta, q_0, F) \qquad (1)$$

where:
Q - finite set of states,
Σ - finite entrance alphabet,
q_0 - initial state ($q_0 \in Q$),
δ - transient function $\delta : Q \times \Sigma \rightarrow Q$,
F - set of final state ($F \subset Q$).

Finite set of states is the set of stages defining diagnostic procedures consisted of accomplishment of required connections by operators and measurements or generating appropriate signals by diagnostic device on a specific input and output of an aircraft.

- Finite entrance alphabet consists of two elements: Z - correct and N - failed.
- Transient function δ, in dependence of actual state and result of checking, transfers an automaton to appropriate state.

An exemplary fragment of a model is presented in Fig. 3. This model has form of a directed graph, that nodes are appropriate states of an automaton, identified by the name (e.g. P-008-00). The states are related by the transient function . A value of this function depends on result of a checking function *test* (e.g. *test 7*). Especially when checking not occurred, from one to only one state transient is possible. This situation appears when either stimulation signal is generated or failed message is displayed (eg. message „block BNO-3PP failed" in state P-012-00).

After modelling stage it is possible to come to implementing the system by means of existing expert systems building tools.

3. System implementation

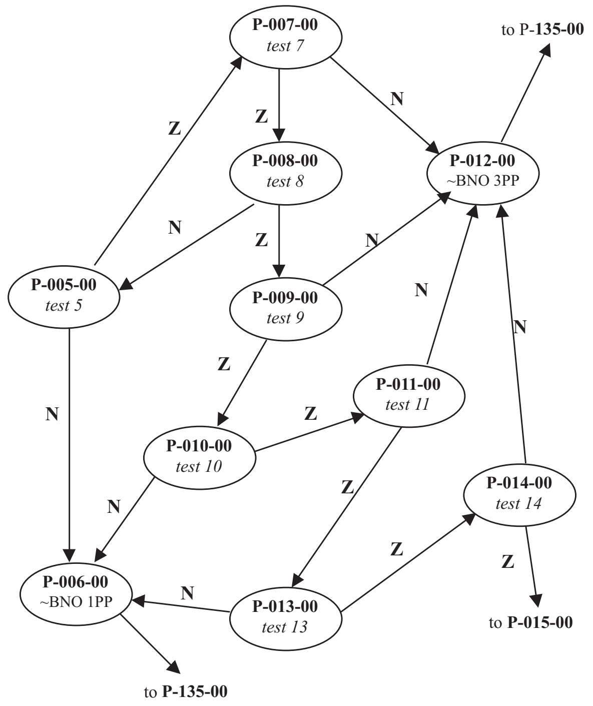

Fig. 3. An aircraft diagnostics model.

Most essential elements of an expert system are: a set of data declarations depending on problem, named knowledge base and an independent on problem (but depending on data structures) program named inference engine. Knowledge base can be represented as: logical, net, production or frame models [10]. Inference engine can realize one or both of two inference techniques: forward chaining and backward chaining.

From an aircraft diagnostic model presented in Fig. 3 arises, that knowledge base of an expert system for an aircraft has to be modelled by production rules and his inference engine has to realize forward chaining. As a result of searching for implementing tools, a CLIPS expert systems shells has been chosen. By means of this tool system can be formalized and implemented.

Taking advantage of CLIPS language [5], nodes and edges of the conceptual graph have been defined. Fig. 4 presents a definition of a node in a form of structure deftemplate with three slots: First slot *nazwa* identifies node of graph thus state of automaton. In slot *wynik*, test result for given state is stored. Slot *parent* is an auxiliary field, permitting to implement test path.

```
(deftemplate MAIN: :parametr
    (slot nazwa (type SYMBOL) (allowed symbols    P-004-00
                                                  P-005-00
                                                  P-006-00
                                                  P-007-00
                                                  P-008-00
                                                  P-009-00
                                                  P-010-00
                                                  P-011-00
                                                  P-012-00
                                                  P-013-00
                                                  P-014-00
                                                  P-015-00
                                                  P-135-00))
slot wynik (type SYMBOL) (allowed-symbols NIEZDATNY ZDATNY)
slot parent (type FACT-ADDRESS SYMBOL) (allowed-symbols no-parent)))
```

Fig. 4. A definition of automaton state in CLIPS.

A definition of two edges of graph in a form of CLIPS rules is presented in Fig. 5.

```
(defrule MAIN: :od5do7
    ?node <- (parametr       (nazwa P-005-00)
                             (wynik ZDATNY)
=>
    (duplicate ?node         (nazwa P-007-00)
                             (wynik (test 7))))

(defrule MAIN: :od5do6
    ?node <- (parametr       (nazwa P-005-00)
                             (wynik NIEZDATNY)
=>
    (komunikat NIESPRAWNY-BNO-1PP)
    (duplicate ?node (nazwa P-006-00)))
```

Fig. 5. Two edges of conceptual graph in CLIPS.

Given rule is activated if all of its conditional elements are satisfying. In this case, if system is in the state P-005-00 and the test result of this state are CORRECT then *od5do7* rule is activated. When the rule is chosen, the new state P-007-00 is created and function test with the parameter 7 is calling. Result of this function is returned in slot *wynik*. In some cases a message is generated (e.g. BNO-1PP DAMAGE).

In Fig. 6 software architecture of proposed diagnostic expert system is presented. For implementation *test* function a C DAQ libraries of LabWindows are applied. DAQ library permits to data exchange with on board devices of MiG-29 aircraft. Furthermore, standard GUI interface is removed from the open source CLIPS application and in this place a customized user interface libraries from

LabWindows environment are inserted. Knowledge base is stored in a form of text files that include structures and rules written in the Lisp language. Knowledge base rules are activated and fired by inference engine embedded in a diagnostic module. This module, including CLIPS source files, is compiled in BorladC++ 5 environment. The whole system is compiled and consolidated in BorlandC++ environment.

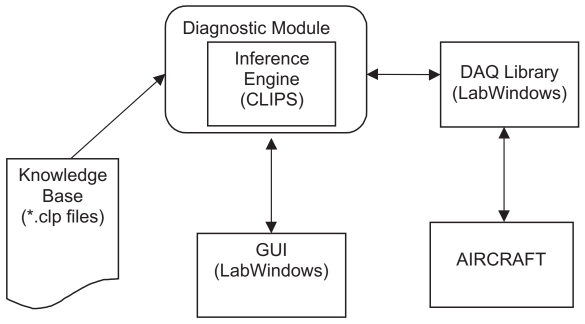

Fig. 6. Architecture of Aircraft diagnostic Expert System.

4. Conclusion

A presentation of conception of an expert system application to MiG-29 aircraft diagnostics has been a goal of this issue. This conception is founded on advantages of CLIPS expert systems shells combined with LabWindows libraries. The whole system has been integrated in BorlandC++ environment.

This solution is tested on simple MiG-29 diagnostic procedure. Positive results of this work let suppose, that given idea allows building diagnostic expert system for MiG-29 aircraft. Obviously to obtain complete system, big effort of coding knowledge base in Lisp notation is needed.

Additionally, after modifications, this system can be applied for checking technical condition of other Russian-made aircrafts owned by Air Force in Poland.

ACKNOWLEDGEMENTS
The authors would like to express his thanks to P. Weryński, Air Force Institute of Technology, for his aircraft terminology consultation.

AUTHOR
Krzysztof Butlewski and **Piotr Golański*** - Air Force Institute of Technology, ul. Księcia Bolesława 6, 01-494 Warszawa, skrytka poczt. 96, Poland; e-mails: krzysztof.butlewski@itwl.pl, piotr.golanski@itwl.pl.
* corresponding author

References

[1] Beynon-Davies P.: *Inżynieria systemów informacyjnych* [Information Systems Development. An Introduction to Information Systems Engineering, Third Edition, Macmillan Press Ltd, 1998], Pol. trans. by Małgorzata Szadkowska-Rucińska, WNT Publishing House, Warszawa 1999.

[2] Borowczyk H., Kwieciński R.: *Tworzenie bazy reguł ekspertowego systemu diagnostycznego z wykorzystaniem zidentyfikowanego modelu diagnozowanego obiektu* [Creation of base of diagnostic system rules applying identified model of diagnosed object], Proceedings of *8th International Conference Airplanes and Helicopters Diagnostics,* AIRDIAG'2005, 27-28.10.2005, Warsaw, Air Force Institute of Technology.

[3] Gartner D., Sheppard J., W.: *An Experiment in Encapsulation in System Diagnosis*. Test Technology and Commercialization - Conference Record, AUTOTEST-CON'96, pp. 468-472.

[4] Gatnar E., Stąpor K.: *Prolog - język sztucznej inteligencji* [Prolog - the language of artificial intelligence], PLJ Publishing House, Warszawa 1991.

[5] Giarratano J.C.: *CLIPS - User's Guide*, http://www.ghg.net/clips/download/documentation/userguide.pdf, 2002.

[6] Homenda W.: *Elementy lingwistyki matematycznej i teorii automatów* [Essentials of mathematical linguistics and automation theory] Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.

[7] Jurkiewicz Z., Lao M. J.: *Język programowania LISP* [Programming language LISP] PWN, Warszawa 1990.

[8] Liu S.T., Kelly G.E.: *Rule-Based Diagnostic method for HVAC Fault Detection*. Proceedings of Building Simulation'89, pp. 319-324.

[9] Merritt D.: *Building Expert Systems in Prolog*, http://dcis.nohyd/ernef.in/courses/AAI/CD-1/ExpertSystemsInProlog/xsip_book.pdf, 2000.

[10] Pospiełow D.A.: *Sprawocznik Iskusstwiennyj intielekt.* Kniga 2. Modieli i mietody. [Artificial intelligence. A guide. Book 2. Models and methods], Radio i Swjaz', Moskwa 1990.

[11] Weissman G., "Tworzenie systemu ekspertowego" [Expert system creation], *Software* 10/96, pp. 24-26.

[12] CLIPS'94, *Third Conference on CLIPS Proceedings* (Electronic Version) http://www.ghg.net/clips/download/documentation/3cpp.pdf, Lyndon B. Johnson Space Center, September 12-14, 1994.