AN APPLICATION OF AN EXPERT METHOD FOR M-28 AIRCRAFT MAINTENANCE SUPPORT

ZASTOSOWANIE METODY EKSPERTOWEJ DO WSPOMAGANIA EKSPLOATACJI SAMOLOTU M-28

Piotr Golański

Instytut Techniczny Wojsk Lotniczych
e-mail: piotr.golanski@itwl.pl

Abstract: *A concept of an expert system that can be used for supporting aircraft diagnostic and maintenance processes is presented in the paper. The paper discusses general principles of an expert system creation as well as main tools used for implementation of such a system. A method of application of one of those tools – CLIPS expert system shell, for solving M-28 aircraft maintenance problems is presented in the paper.*

Keywords: *aircraft maintenance, expert systems, CLIPS.*

Streszczenie: *W artykule przedstawiono koncepcję wykorzystania systemu ekspertowego do wspomagania obsługi i diagnostyki statku powietrznego. Na wstępie omówiono ogólne zasady budowy systemów ekspertowych. Krótko omówiono podstawowe narzędzie służące do ich implementacji. Przedstawiono sposób wykorzystania jednego z narzędzi jakim jest powłoka systemów ekspertowych CLIPS, do rozwiązania problemów obsługowych samolotu M-28.*

Słowa kluczowe: *obsługa statku powietrznego, systemy ekspertowe, CLIPS.*

1. Introduction

Computer-aided maintenance systems right from their early beginnings have always been using methods related to the artificial intelligence and were and are, in fact, expert systems [1, 9]. The main difference between expert systems and conventional problem solving systems is the way a problem is solved. In conventional approach, a problem is solved by means of an algorithm which processes data. In an expert system's approach, a problem is described and solved only by means of a properly prepared data structures. Therefore, to build the expert systems, a completely different set of tools must be used. In conventional systems mainly imperative programming languages based on C and Pascal are used, whereas in expert systems – declarative languages such as Lisp [7] or Prolog [4]. Very often for an expert system creation specialize tools – expert system shells are used. They are, in fact, ready for use expert systems without any built-in knowledge.

Usage of one of expert system shells for creation of the M-28 aircraft maintenance supporting system is presented in the paper. Fundamentals of such a system creation and performance, as well as an application of an expert method in a simple diagnostic task, is also showed.

2. Problem analysis

Expert systems, when used as a maintenance management system, apply models of technical objects maintenance processes. A model of such a process, which would also describe the way an expert system works, can be presented using a finite state machine mathematical model [6].

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

where:

Q - finite set of states,

Σ - finite input alphabet,

q_0 - initial state ($q_0 \in Q$),

δ - state-transition function $\delta: Q \times \Sigma \rightarrow Q$,

F - set of final states ($F \subset Q$).

The finite set of states Q shows possible states of the expert system knowledge about the serviced object. Finite input alphabet Σ is a set of symbols which enable the dialogue between the user and the system. State-transition function δ , depending on actual system state and results of user-system dialog, changes the state of the system knowledge. The set of final states F determines states of the knowledge in which the system finds a solution to the problem.

The model (1) introduced above describes well also basic components and performance of an expert system. However, there are also other ways of describing the structure of an expert system. In this paper, a description of an expert system structure described in the work [10] is used.

In this structure, an expert system consists of four parts (Fig. 1):

- *Knowledge Base* – a declarative representation of the expertise knowledge, often in the form of if <conditions> then <actions> rules;
- *Working Memory* – data that is specific to a problem being solved;
- *Inference Engine* – the code being the core of the system which draws conclusions basing on data from both: the *Knowledge Base* and problem-specific data in *Working Memory*;
- *User Interface* – the code which controls the dialog between the user and the system.

Fig. 1 Expert System Structure.

The key element of an expert system is a set of data declarations called knowledge base, which is problem-specific, and a program called the *Inference Engine*, which is independent from a problem but highly dependent on data structure. A *Knowledge Base* is an implementation of a domain expert knowledge created by a knowledge engineer as an initial knowledge state q_0 . An inference engine uses the *Knowledge Base* as a transient function δ and generates a solution of a problem defined by the User. The User interacts with the system using the User Interface (usually GUI – Graphical User Interface) by means of elements of finite input alphabet Σ .

Inference engine can apply one or both of the two inference techniques: *forward chaining* and/or *backward chaining*. *Forward chaining* is a top-down method in which facts are taken into consideration when they become available. The system attempts to draw conclusions (from met conditions in rules) which are then executed as actions. Such a process is continued to the point where goals is achieved. Backward chaining is a bottom-up procedure which starts from goals and is searching through the knowledge base until it finds rules which actions fit the expected goal. If the condition of such a rule isn't met, it is being added to the list of goals. Backward chaining is rather more a verification than an exploration process and is usually applied in technical diagnostics systems [4].

3. Implementation and performance of the system

In the case of the system presented in the paper the CLIPS - *C Language Integrated Production System* (an expert system shell developed by the Software Technology Branch – NASA/Lynnon B. Johnson Center [5]) was used to the implementation of the *Inference Engine* and the *Knowledge Base*. CLIPS is a tool which can be found in many expert system applications connected with technical diagnostics [2, 3, 8, 11, 12]. CLIPS operation is based on using inference mechanisms, which is a part of its *Inference Engine*, and a set of rules and facts saved in the m-28.clp file. Rules have a form of the implication:

$$\text{LHS} \Rightarrow \text{RHS}$$

where:

LHS – antecedents of rule (*Left Hand Side*),

RHS – consequent of rule (*Right Hand Side*).

As a result of the inference engine action, rules in which antecedents (LHS) match the facts in the *working memory* are chosen. In the next step rules are being triggered according to their priority and their consequent (RHS) is executed. In this way *the forward chaining* is being executed in the system.

In the case of the system presented in the paper in the initial state q_0 a no-goal rule (Fig. 2) is being triggered. It is due to the fact that after loading the working memory from the m-28.clp file there is no defined goal which means that no service activity was yet chosen. Executing the consequent of this rule causes the system to wait for the user to choose a service activity.

```
(defrule no-goal ""
  (not (goal $?))

  =>

  (format t "read new goal ")
  (assert(goal is {cel})))
```

Fig. 2 Selection rule of the service activity.

After choosing the service activity, i.e. determining the goal, the inference engine is looking for indirect goals, which are the condition of performing chosen activity, by triggering the **propagate-goal** rule (Fig. 3). If, for example, the activity identified as pg44.diag (diagnostics of the RESZTA PALIWA lamp) was chosen, facts presented in Fig. 4 will be matching the propagate-goal rule. As the result of the execution of the rule consequent two goals: pg44.test and PG44 will be assigned. It can be seen that even though the CLIPS inference engine is executing the forward chaining, properly defined facts allow the system to perform the backward chaining and goals that are being set can be found in rules antecedents.

```
(defrule propagate-goal ""
  (goal is ?goal)
  (rule (if ?variable $?)
        (then ?goal ? ?value))
=>
  (assert (goal is ?variable)))
```

Fig. 3 A goal propagation rule.

```
(rule (if pg44.test is ON and
        PG44 is OFF)
      (then pg44.diag is przepalona))

(rule (if pg44.test is ON and
        PG44 is ON)
      (then pg44.diag is sprawna))
```

Fig. 4 Facts containing the pg44.diag goal.

Repeated triggering of the propagate-goal rule leads to establishing goals – initial activities.

When this state is reached, the dialogue rules, which check whether goals are reached, are triggered. In the Fig. 5 the ask-question-legalvalues-and-instruct rule is presented. This rule generates the question: has a ?variable variable a ?value value ? This question is sent to the operator by the *User Interface*. By using this rule the system can, for example, ask whether the given device is turned on. After receiving the reply, the system modifies the set of facts in the working memory and changes it's state to a new one q_i and the process of looking for a problem solution is being continued.

```

(defrule ask-question-legalvalues-and-instruct ""
  (declare (salience 15))
  (legalanswers ? $?answers)
  ?f1 <- (goal is ?variable)
  ?f2 <- (question ?variable ? ?text)
  ?f3 <- (instruction ?variable ? ?text_instr)
  ?f4 <- (rule (if ?variable ? ?value $?))

  =>
  (retract ?f1)
  (format t "%s " ?text) ;wersja z interpreterem
  (printout t ?answers " ") ;wersja z interpreterem
  |
  (bind ?reply (test (str-cat "Test&" ?variable "&" ?value "&" ?text)))
  (printout t ?answers)

  (if (member (lowercase ?reply) ?answers)
    then (if (eq ?reply nie)
      then (komunikat (str-cat "Ustaw&" ?variable "&" ?value "&" ?text_instr))
      ;(format t "%s&n " ?text_instr) ;wersja z interpreterem
      (assert(goal is ?variable))
      else (assert (variable ?variable ON))
      (retract ?f2)
      (retract ?f3)) ;)) ;tu zakonczenie wersji debug ))
    else (stop ?variable) (halt) )) ;tu zakonczenie wersji release

```

Fig. 5 Example of a dialogue rule.

4. Conclusion

A concept of an expert methods usage for supporting the service operation of the M-28 aircraft was presented in the paper. This concept is based on CLIPS expert systems shells.

Proposed solution was tested on a simple service procedure of the M-28 aircraft. Positive results of this work indicate that, with extended knowledge base, it is possible to build a complete diagnostic expert system for this aircraft. What's more important, such a system, after necessary modifications, could also be used for diagnostics of other aircrafts.

5. References

- [1] Barker V. E., O'Connor D. E.: Expert systems for configuration at Digital: XCON and beyond. Communications of the ACM, 32(3):298--318, March 1989.
- [2] Dapeng T., Li Peiyu L., Pan Xiaohong P., Embedded Fault Diagnosis Expert System Based on CLIPS and ANN, Computational Science – ICCS 2007 Lecture Notes in Computer Science Volume 4490, 2007, str. 957-960.
- [3] Gartner D., Sheppard J., W.: An Experiment in Encapsulation in System Diagnosis. Test Technology and Commercialization - Conference Record, AUTOTESTCON'96, ss. 468-472.
- [4] Gatnar E., Stapor K.: Prolog – język sztucznej inteligencji. Wydawnictwo PLJ, Warszawa 1991.
- [5] Giarratano J.C.: CLIPS – User's Guide. 2002.

- [6] Homenda W.: Elementy lingwistyki matematycznej i teorii automatów. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.
- [7] Jurkiewicz Z., Lao M. J.: Język programowania LISP. PWN, Warszawa 1990.
- [8] Long H., Wang X.: Aircraft fuel system diagnostic fault detection through expert system,” Proc. IEEE Sixth World Congress on Intelligent Control and Automation, WCICA2008, 2008 str. 7104-7107.
- [9] Mazurkiewicz D. Computer-aided maintenance and reliability management systems for conveyor belts. Eksploatacja i Niezawodność –Maintenance and Reliability 2014; 16 (3): 377–382.
- [10] Merritt D.: Building Expert Systems in Prolog. Amzi! inc., 2000.
- [11] Yan C., Ma S., Zhou G., Fang J., Fault Diagnostic Expert System of Rolling Element Bearing Based on CLIPS, Journal of Information & Computational Science 10:10 (2013) (<http://www.joics.com>). July 1, 2013, str. 3053–3062.
- [12] CLIPS’94, Third Conference on CLIPS Proceedings (Electronic Version), Lyndon B. Johnson Space Center, September 12–14, 1994.

Piotr Golański works as assistant professor in Training Systems Division of Air Force Institute of Technology. His main interests are mathematical modeling and the implementation of models in training simulators.