Author version, published in:
Annual Review in Automatic Programming, volume 17, pp. 201-206, 1992.

Knowledge specification and representation for an “intelligent” interface devoted to process monitoring and supervision

E. Le Strugeon, M. Tendjaoui, C. Kolski

Laboratoire d’Automatique Industrielle et Humaine - URA CNRS 1118, Université de Valenciennes et du Hainaut-Cambrésis, B.P. 311, Le Mont Houy, 59304 Valenciennes Cedex, FRANCE

Abstract. This paper presents our approach of cooperation between a human operator and a decision aid tool by means of an “intelligent” interface manager: the Decisional Module of Imagery (D.M.I.). The “heart” of the D.M.I. is an expert system which manipulates three main objects (the WHAT, WHEN and HOW objects) described next. Knowledge specification and representation for the expert system, and the way we chose to implement it, are then explained.

Key words. Process control, man-machine systems, expert systems, knowledge engineering, inference processes.

INTRODUCTION

The increasing complexity of industrial processes necessitates the design of control, supervision and decision support tools that are able to evolve along with the control system. In such conditions, the man-machine interface plays a vital role where the information being manipulated becomes more and more complex, i.e. safety systems, production control systems and environment protection systems (Rasmussen, 1986; Sheridan, 1988; Millot, 1988, 1990).

In a way to improve the co-operation between the human operators and the decision aid tools in the control rooms of industrial processes, our work concerns us with the study of an ‘intelligent” interface, and is aimed at realizing such an interface supervised by an ‘intelligent” manager called the D.M.I. (Decisional Module of Imagery). Our approach consists of using an expert system to ensure this co-operation (Tendjaoui *et al*., 1990). It is currently validated in the laboratory and is integrated into an experimental platform. The objective is to compare the operator’s behavior with his performance whether an ordinary or “intelligent” interface is being used. This comparison is done by using a set of failure scenarios.

This paper describes first the D.M.I. and the objects manipulated by this approach of “intelligent’ interface. Then we explain how the “heart” of the D.M.I. is implemented using artificial intelligence techniques.

THE DECISIONAL MODULE OF IMAGERY

Our approach is tailored to the area of supervisory process control. Its goal is to design an intelligent imagery manager called “Decisional Module of Imagery” (D.M.I.). This approach can be integrated into the global model of the Man-Machine system in automated process control rooms to obtain an overall assistance tool (Figure 1). The supervisory calculator centralizes the whole process scored data. These data are accessible by both the decision support expert system and the D.M.I. Using this data, the decision support expert system infers information such as predictive, diagnosis or recovery procedures. This set of information is transmitted to the D.M.I., which selects those that can be presented to the operator. This selection is based on a task model to be performed by the operator, and on “operator” model containing information about the operator.

The task model is currently restricted to problem solving tasks and results from a previous analysis of fixed tasks which have to be performed by the operator. This model is based on the qualitative general model of Rasmussen (1980). Whereby a task is built through four information processing steps: event detection, situation assessment, decision making and action. This task model contains a set of process significant variables used by the operator while performing his different tasks.

The operator model integrates a set of ergonomic data which is presently limited to: (i) three possible levels of expertise for the human operator (unskilled, experienced, expert), (ii) the type of displays associated each of the operator’s cognitive behavior, corresponding to Rasmussen’s model, (iii) the representation mode associated to each type of display.

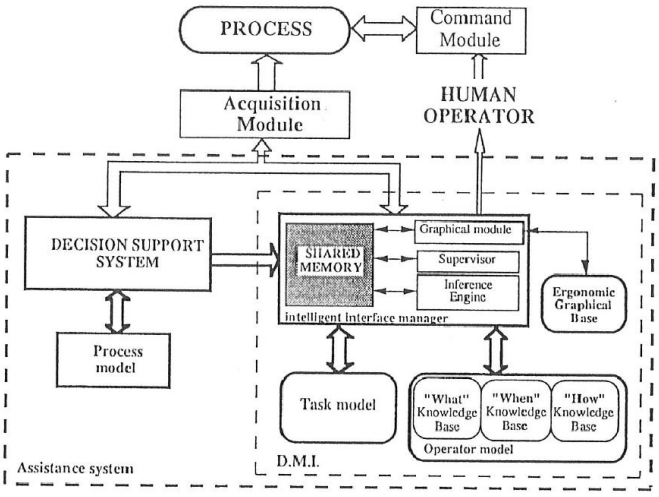

Fig 1. Global Man-Machine system integrating the Decisional Module of Imagery (Kolski *et al*., 1990)

The aims of the D.M.I. are: (i) to select the data that can be displayed on the screen taking into account both the operational context of the process and the informational needs of the operator, in order to enable the operator to supervise the process and to define possible corrective actions when a failure appears; (ii) to define the ergonomic parameters associated with presentation of this information in order to make the human operator's understanding easier; (iii) to add to this supervisory imagery the corrective advice given by the decision support expert system in order to justify its reasoning and thus to prevent possible conflicts between the system and the human operator.

THE OBJECTS MANIPULATED BY THE "INTELLIGENT" INTERFACE

Information "intelligent" selection is based on a model of the tasks to be performed. Alternatively, the selection may be based on an operator model containing knowledge of the three ergonomic considerations shown in Figure 2 (Tendjaoui *et al*., 1991a, 1991b): (j) What to present to the operator (we consider here that "what" contains the "why" by justifying the information displayed); (ii) When shall we display it, (iii) How shall we display it.

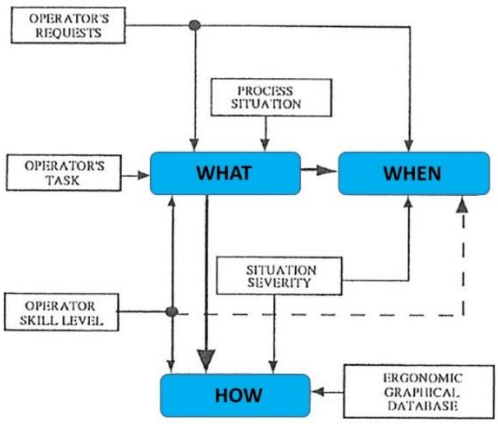

Fig. 2. The "What - When - How" concept

An ergonomic database centralizes ail the presentation modes that can be selected and displayed by the graphics module. This graphics module is controlled by the inference mechanism through a shared memory whose access is controlled by a supervisor. The D.M.I. is developed using the "C" language on a VAX/VMS. The software hierarchy is described in more details in a paper by Tendjaoui *et al*. (1991b).

The "What", "When" and "How" problems are described below.

The "WHAT" problem

The problem concerning what is to be displayed to the operator depends essentially on three criterion:

Operator requests. If the operator, when performing his supervisory tasks, requests information on for example, the state of a variable or to justify an action, then the D.M.I. has to supply this information.

Operator classification. If the decision support tools perceive an error in the system and propose advice or action, the operator can (i) be in agreement with this advice and act accordingly or, (ii) disagree and request some justification for the advice. The level of detail in this justification will depend on the operators skill level, e.g., a novice operator will tend to require more detailed justification than an experienced operator.

The operators task in relation to different process operations. The operator tasks and therefore his informational needs will change according to the state of the process. For example, in a transition situation, the operator may need some advice on starting the process, whereas to assess the effects of a corrective action, the operator needs information about the progression of the correction. During an abnormal process state, alarms are automatically selected and displayed, but where an "uncommon" abnormal situation develops, the D.M.I. can focus on variables that can affect the production and/or the security of the system, by suppressing all information or alarms that do not affect either production or safety.

The "WHEN" problem

The problem concerning "when" relevant information has to be presented to the operator depends on the nature of the information that he requested and also on the seriousness of any impending situation. Severity evaluation is bound by production targets and security constraints. To know when the D.M.I. has to display information, we first have to know what this information actually represents (the "WHAT"). For example, alarms are displayed to the operator as soon as they occur, whereas information on action or advice proposed by the decision support tools is only displayed to the operator when requested or when a process situation becomes hazardous.

The D.M.I. has, however, to evaluate the mental workload of the operator in order to determine whether any additional information could be adequately assimilated by him. Mental workload depends on many factors, e.g., the number of potential hazardous situations encountered, their severity, the operators skill level, and so on.

The "HOW" problem

To know how to present a piece of information to the operator, we first have to know what this information represents and the severity of the process fault at that moment. Information is then displayed in accordance with predefined ergonomic modes (Figure 2). If several different presentation modes are available to the operator, which can be used with the same efficiency, then he can configure the interface according to personal preferences. Some examples of "HOW" are: (i) to indicate the progression of a variable during the process, graphical curves might be appropriate (they inform the operator about trends in the variables history), (ii) color, red for example, can be used to indicate process status.

KNOWLEDGE SPECIFICATION AND REPRESENTATION FOR THE "INTELLIGENT" INTERFACE

The "heart" of the D.M.I. is an expert system that is in charge of giving to the supervisor means for controlling screen displays. Its function is, more precisely, to give answers to the three questions WHAT, HOW and WHEN (described earlier). Several methodological and technological choices were necessary to be made before any implementation of the expert system.

Knowledge specification and representation

A first choice, the most important of all, is the choice of the knowledge specification. The expert system needs some knowledge defining the current state of the parameters. Those parameters characterize, on one hand, the supervised unit and, on the other hand, the operator that supervises this unit. The

problem is to decide what is the knowledge required by the expert system to give interesting results. That means, in a more technical way to choose the facts that will be manipulated by the expert system.

Nine facts are currently used to represent parameters characterizing both supervised process and human operator: the facts “Functioning situation”, “Situation_severity”, “Operator’s_task”, “Operator’s class”, “Operators_request”, “What” (must be displayed), “Previous_what” (was displayed at the last step), “When” (the screen modifications will occur) and “How” (which type of display). These facts are interdependent. They are multivalued: they may have several values simultaneously because they are used in the management of the screen displays on which more than one change can occur at the same time. In example, in an abnormal situation of the process, the supervision being made by a novice operator, the system may have to display, in the same time, a view showing the evolution of the variables and a view with acts advices. In that case, two answers are given to the question (WHAT) “What must be presented to the operator?“.

Three of the nine facts group together the results given by the expert system after inferences: the facts WHAT, WHEN and HOW. The rules base is split up into three “sub-bases” corresponding to those three results facts.

To allow associations between a fact and a value in a coherent way, an annex database, called “Possible Facts Base”, was created. It is used to check the validity of such associations. It contains all of the possible values for each fact. Its structure is identical to the established facts base’s one. In example, the further information can be found in it: the fact “Fact1” can take the values A, B and C. The association between Fact1 and value D would be recognized as disabled.

The data structures for the facts bases and for the rules base are illustrated in Figures 3 and 4. Both of these structures are based on the “chained lists” technic. So, the facts base is represented by a variable of the type “facts list pointer”. Each cell “fact” of the list contains another list with its current values. In the same way, the rules base is represented by a pointer on a chained list of rules. Each rule is composed of a conditions (or premisses) list and a conclusions list. Conditions, like conclusions, are of the type “fact” (described in the appendices), excepted that a single value is associated to each of them. Examples of rules are given in appendix 2.

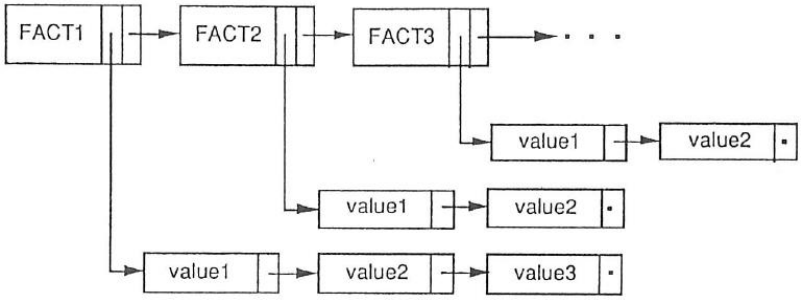

Fig. 3. The facts data base

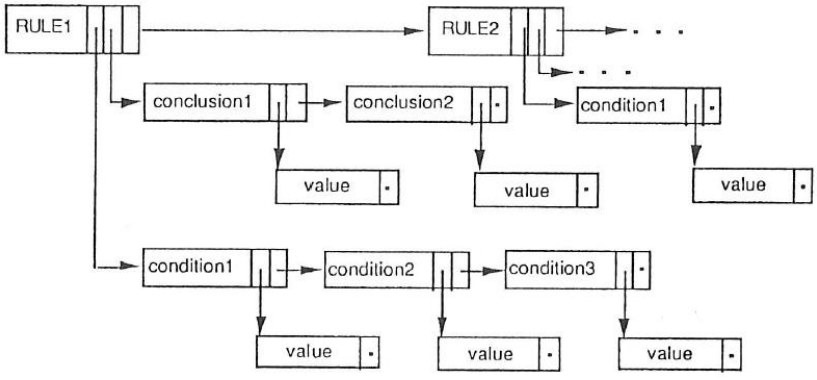

Fig. 4. The rules database

The kind of inference

A second development choice purpose is the inference mechanism. The problem that must be solved is the kind of inference to implement to run the inference engine: data or goal-driven inference? In the D.M.I. context, the data-driven Inference seemed to be the most adapted. There are two reasons for that: (j) the searched goal is to establish a maximum of facts and to deduce all of the values that it is possible to deduce for one fact, (ii) a lot of rules have an identic conclusion, that is why the time needed by a goal-driven research would be too important.

Methodology for building the rules

A third choice concerns the way to build and handle the rules database. “One of the attractions of the expert system approach lays in the possibility of “teaching” progressively the initial system in revising or completing the knowledge base put at its disposal” (Farreny and Ghallab, 1987). For all that, it is essential for the rules base to be easily modifiable, aiming at a progressive refining. Some rules are added, deleted or modified to improve, step by step, the rules database. In that goal, the software FirstClass is used as described in the Figure 5.

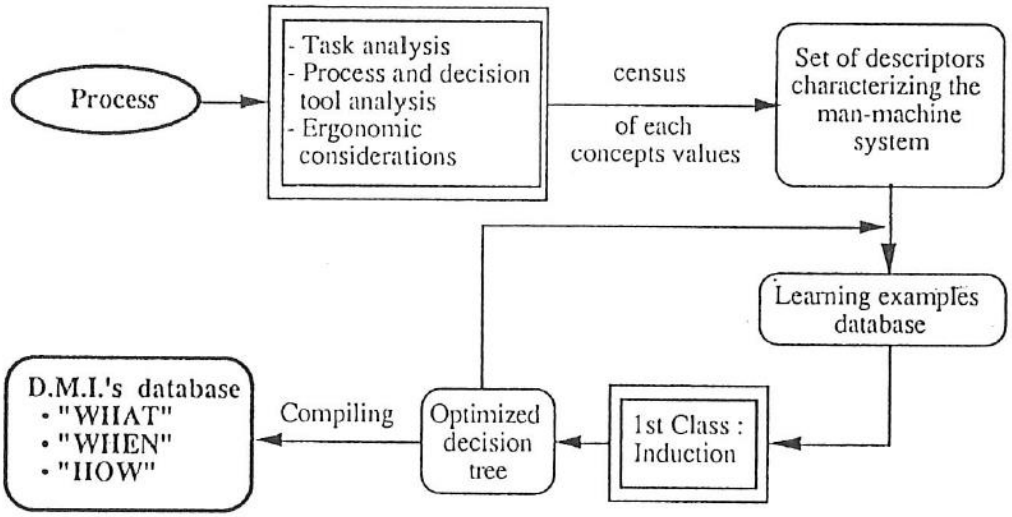

Fig. 5. Method for constructing the knowledge database
(Tendjaoui *et al*., 1991a)

With a set of descriptors (characterizing the simulated process) and a set of examples, the software FirstClass generates a decision tree, by the use of the Quinlan’s induction algorithm “ID3” (1979). That tree describes a hierarchy among descriptors. Although such a tree is “readable” by humans, il is not yet exploitable in that shape by the expert system. Actually, the expert system uses production rules formed on the model: **IF conditions THEN conclusions**. The decision tree must be modified to become understandable for the expert system. This modification (or “compilation”) of the tree is described now.

The technique used to translate decision trees into production rules is based on syntactic analysis. The program that ‘reads” the decision tree and produces the corresponding rules is a kind of “transducer” or “compilator”: it translates the tree written in a particular format (a language we called L1), into rules written in another format (a language we called L2). The compilator makes the operation of translation T such that: T(L1)=L2. The grammar used to specify the language L1 is described in appendix 3.

The compilation module is composed of four important functions (the functions P, S, T and C) that correspond to the left members of the derivation rules. With a stack system and with pointers, the tree from FirstClass is “deciphered” and the production rules are built in a format useful for the expert system. This operation is repeated three times in order to build the three modules of the rules base (the modules “WHAT”, “WHEN” and “HOW”). All the rules owning to the module “WHAT” have, in the conclusion part, a descriptor as followed: **WHAT = a value**. Both other modules are built the same way. In the actual advancement state of the D.M.I., the rules base is composed of 335 rules for the three modules.

That methodology has several advantages:

1. Data (descriptors and examples) are easily modifiable.
2. To create or modify the rules base, the reflection medium is not production rules, but examples, which are far closer to the natural way of reasoning than rules.
3. The obligation to input the values of all descriptors for every examples is more an aid than a constraint. It imposes to define very precisely each of the particular cases that are the examples, because all of the parameters must be specified, even those that are not taken into consideration. So, if the descriptor A does not take place in any way in a given situation that must be specified.
4. If in a particular situation, no solution has been proposed (the required example does not exist in the input data) the software FirstClass reports "no-data". It allows revealing lacks that can exist among rules. One has to notice, however, that not all possible situations have to be described as examples in an exhaustive way.
5. With FirstClass, input examples can "run". It is like a guided route in the decision tree. It allows checking quickly if a specific situation is specified in the examples set and if the end result (which is a leaf of the tree) is the previewed one.

With this methodology, is is possible and easy to add, retry and modify elements of the rules base, to adapt and adjust the system to our needs. A simple methodology was also used in order to facilitate changes that may occur among the set of facts and values that are associated to them. Facts names are written in a file, which is modifiable with a classical text editor. In the initialization step, the system just reads that file to update the Possible Facts Base.

Justifying the expert system' reasoning

Finally, the subject of the last choice is the justifications of the expert system reasoning. During the adjusting stage of the D.M.I., the rules base is being modified (refereeing to last paragraph) because solutions given by the expert system are not always the best that could be expected. It is important to know in detail the inference course that leads to the imperfect result.

The expert system thought process must either be justified to the operator to allow him to understand why such answers are given by the system and, on that foundation, be able to decide if his own argument is correct.

So as to justify the expert system logical line of reasoning, some data are memorized: the initial context (like a photography of the facts base before any inference), the numbers of the activated rules and the final context (the current state of the facts base after the inferences). This information is stored into memory. With this trace, the operator can understand which knowledge was brought into operation to deduce new facts and get out his conclusions about it. Actually, some explanations are very useful if operator and machine do not agree on what have to be done to solve one particular problem. They allow solving potential clashes.

CONCLUSION

This paper submitted our works contributing at the "intelligent" interface notion, in the field of complex process control. We focused here on the way by which the "heart" of the interface is implemented with the help of artificial intelligence techniques. The management of the interface displays is condensed in the three questions WHAT, WHEN and HOW, to which the expert system gives the adequate answers. Now and in order to validate this approach, we are implementing a testbed. The results of that experimental step will be the subject of other articles.

REFERENCES

Farreny, H., and M. Ghallab (1987). Eléments d'intelligence artificielle. Editions Hermes, Paris.

Kolski, C., M. Tendjaoui, and P. Millot (1990). An "intelligent" interface approach. The second International Conference on Human aspects of advanced manufacturing and hybrid automation, Honolulu, Hawaii, USA, August 12-16.

Millot, P. (1988). Supervision des procédés automatisés et ergonomie. Editions Hermes, Paris, December 1988.

Millot, P. (1990). Coopération homme-machine, exemple de la téléopération. Journées du GR Automatique, 17-19 October 1990, Strasbourg, France.

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), “Expert systems in microelectronic Age”, Edinburgh University Press.

Rasmussen, J. (1980). The Hurnan as a System Component. In “Human Interaction with computer”, H. T. Smith and T.R.G. Green (Editors), London Academic Press, 1980.

Rasmussen, J. (1986). Information processing and human-machine interaction. An approach to cognitive engineering. North-Holland series in system science and engineering, A.P. Sage.

Sheridan, T.B. (1988). Task allocation and supervisory control. In “Handbook of Human-Computer Interaction”, M. Helander (Ed.), Elsevier Science Publishers B.V., North Holland, 1988.

Tendjaoui, M., C. Kolski, and P. Millot (1990). Interaction between real-time aid expert system, intelligent interface and human operator. International Symposium Computational Intelligence 90 “Heterogeneous knowledge representation systems”, September 24-28, Milano, Italy.

Tendjaoui, M., C. Kolski, and P. Millot (1991a). Knowledge based interface approach for real-time aid expert system. IFAC/IMACS “SAFEPROCESS’91” Symposium, September 10-13, Baden-Baden, Germany.

Tendjaoui, M., C. Kolski and P. Millot (1991b). An “intelligent” approach for ergonomic design of Man-Machine interfaces in process control. International Journal of Industrial Ergonomics, 8, 345-361.

APPENDICES

A1. Implementation

The type “fact”, written in C, is implemented as following:

```
typedef struct fact
  {  word name;                /* name of the fact          */
     pt_value list_of_values;  /* values list of the fact   */
     struct fact * next;       /* pointer on the next fact  */
  } fact, * pt_fact;
```

A2. Examples of rules

```
IF      Operator_Class    = 3
AND     Severity          = 2
AND     Request           = No_Request
AND     Situation         = Abnormal
THEN    What              :=  Actions_Plan
AND     What              := Deep_And_Justified_Fluence_Graph

IF      Request  = Help_To_Stop
AND     What     = Help_To_Stop
THEN    When     := Now

IF      Operator_Class    = 2
AND     Previous_What     = Variables
AND     What              = Deep_And_Justified_Fluence_Graph
THEN    How               := Detailed_Fluence_Graph
```

A3. Grammar

The language L1 is specified by the following Chomsky grammar:

- the finite terminal vocabulary =

  {line_number, fact_name, fact_value, ??, :, &, -}

- the finite auxiliary vocabulary = {S, T, C}

- the axiom (or “sentence”) = {A}

- the set of “context-free” derivation rules:

A::= line_number fact_name ?? S

S::= {line_number fact_value : T}

T::= A I C

C::= {-} value_Conclusion_fact {line_number & C}