NÉOGANESH: TOWARDS A GENERIC KNOWLEDGE-BASED SYSTEM FOR THE CONTROL OF MECHANICAL VENTILATION.

M. DOJAT [1], **F. PACHET** [2]

(1) INSERM Unité 296, Faculté de médecine, 94010 CRETEIL.
(2) LAFORIA Institut Blaise Pascal, Université Paris VI, 75005 PARIS.

I. INTRODUCTION.

Although computers have several functions in clinical practice such as data-base management, data acquisition and physiological signal processing, sophisticated systems based on techniques involving Artificial Intelligence (AI) which might provide advice in the choice of therapy and assistance in diagnosis are not yet widely used. Several such systems are described in the literature, but only a few are routinely used in clinical practice (see (1) & (2) for a collection of papers on medical AI). "Intelligent" decision support systems will help the physician to make better clinical decisions but they have yet to be tested at the patient's bedside.
We describe here a rule-based system for monitoring physiological data in real time and for controlling the mechanical assistance provided to a ventilated patient. This system is now being tested at the Henri Mondor Hospital in Créteil, France. First we will define the medical problem and technical constraints then we will describe our system and the knowledge representation and lastly, we will consider the generic aspect of our system.

I.1. The medical problem

The mechanical assistance provided to a patient with respiratory insufficiency must be well adapted to his physiological needs: too high a level of assistance results in unacceptable *hyperventilation*, and too low a level leads to extra work for the patient's respiratory muscles. The clinician must appreciate the *respiratory comfort* of the patient and the evolution of his ventilation, and set the ventilator settings accordingly.
Control algorithms which automatically maintain a physiological parameter, such as respiratory frequency or inspired volume per minute, under or above the threshold set by the clinician have severe limitations (3). They do not reflect the physician's attitude which takes into account the general context of the patient's ventilation and its evolution. In this respect, a rule-based approach is *a priori* better adapted to represent the clinical expertise on which the physician's attitude is based.
The second task of a system designed to assist the clinican in Intensive Care Unit (ICU), is to reduce the mechanical assistance gradually, until the patient is able to breathe alone. This is known as the *weaning process*, and for some patients who have been on prolonged mechanical ventilation it can be complex. In such cases, the clinician relies on the considerable expertise acquired during long clinical practice. He or she must assess the patient's capacity to breathe alone before any disconnection from the ventilator, in order to avoid repeated connections and disconnections.
Strategies for patients who are difficult to wean have been proposed (4,5), but they are seldom used in practice, because they need constant monitoring of the patient's ventilation.
The system presented here implements such a strategy. It is based on the knowledge of weaning management acquired by the clinical staff of the ICU at the Henri Mondor Hospital.

[1]Offprint requests to: M. DOJAT
INSERM U.296 Faculté de médecine, 8 av du Général Sarrail, 94010 Créteil France.
phone: 33 1 48 98 46 03 fax: 33 1 48 98 17 77 e.mail: dojat@laforia.ibp.fr

Contrarily to other approches (6,7,8,9), our system deals with a volontarily limited problem:

- only one mode of ventilation is managed by the system: Pressure Support Ventilation (PSV). Recent studies have shown that this mode is of interest because it reduces respiratory muscle work while allowing satisfying gas exchange (5,10).
- patients ventilated with the system should have spontaneous respiratory activity.

These limitations allowed us to design a **closed-loop system** which controls the ventilator without any intervention by the clinician.

I.2. Backgrounds

To solve the problems of control and weaning, we designed an initial prototype: the Ganesh system (11). This system proved valid when used at the Henri. Mondor Hospital, and has shown good results in clinical application (12). However, the nature of the knowledge representation in the inital environment (a conventional 0+ expert system shell) severely limited its extension (13). Clearly, the Ganesh system suffers from: 1) a poor representation of the domain objects 2) a lack of modularity 3) a lack of facility for building a powerful interface 4) an unclear separation between various aspects of the reasoning (knowledge and control). Therefore to have an extensible and generic system at our disposal and also to solve our problems of knowledge representation we designed a new knowledge-based system: **NéoGanesh**.

II. METHODS

II.1. Programming environment

Because object orientation is well adapted to increasing the modularity and the quality of representation, we chose the Smalltalk-80 language (14), which includes an interactive programming environment. However, object-oriented languages do not provide any inference facility. To cope with inference needs, Laursen & Atkinson proposed a rule-based system called Opus (15) which was integrated into Smalltalk-80, and Pachet recently designed NéOpus, an extension of Opus (16, 17). This environment constitutes a first order forward-chaining inference engine. NéoGanesh based on NéOpus was designed to model medical expertise used in ICU. We chose to implement our system on a PC compatible that is the type of micro-computer widely used in ICU. We therefore used Smalltalk-80 v.4 with the Windows 3.0 environment.

II.2. The overall achitecture of NéoGanesh

At the patient's bedside, NéoGanesh acquires and processes a set of data. We use three parameters to appreciate the respiratory comfort of the patient: *respiratory frequency* (Fr), *tidal volume* (Vt), and *end-tidal* CO_2 *pressure* (PCO_2). Respiratory frequency is the most important index because in many circumstances it is a precise reflection of respiratory muscle adaptation to the imposed workload (18). The *level of pressure support* (PS) provided to the patient is used to evaluate the quality of the respiratory system. All these parameters are obtained from external devices via serial communication (Fr, Vt, PS are obtained from a ventilator, and PCO_2, from a CO_2 analyzer); the system modifies the ventilator settings (PS and ventilatory mode). From the physiological data, it evaluates the respiratory comfort of the patient and assesses the stability of the breathing

pattern. All the physiological parameters except one, which is obtained from a CO_2 analyzer (Ohmeda 5200, France), are measured by the ventilator (Veolar, Hamilton Switzerland) which was especially adapted to our study by the manufacturer. Smalltalk methods were defined for the management of communications with the connected devices.

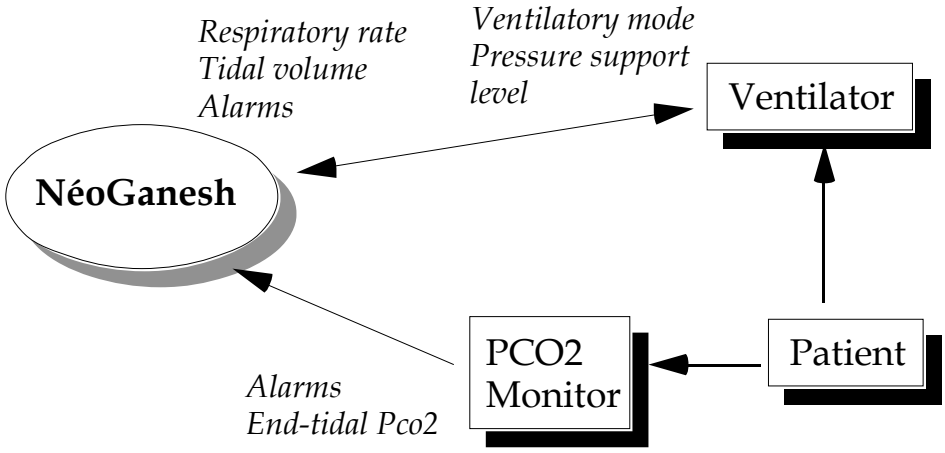

Fig. 1. The architecture of NéoGanesh

II.3. The interface

The interface was designed with the Model View Controller paradigm (14). By acting on the mouse, the user feeds the system with information about the patient and launches a ventilation expert consultation. A simulation mode is also available enabling the user to interact with the system and modify the physiological parameters, in order to test typical situations.

III. RESULTS

III.1. Knowledge representation

Our system uses a three-axis knowledge paradigm to represent knowledge: **domain objects** defined by Smalltalk classes, **rules** which represent the knowledge of the expert, and **meta-rules** used to control the firing of rules.

III.1.1. The domain objects

The system has a representation of every real object involved in the ventilation process. According to the object-oriented paradigms (14), these real objects are represented as Smalltalk *objects*, which are instances of Smalltalk *classes*. Three categories of classes are defined: the *Clinician,* the *Patient* and the *Device*. We describe below the most important class: the *VentilationExpert* class, a subclass of the *Clinician* class.
The *VentilationExpert* can modify the settings of the ventilator. Its knowledge is represented by *production rules,* as explained in the next section.
The expert object responds to messages that evaluate the patient's status, such as *ventilation* which represents the level of ventilation (normal, polypnea...) or *severity* which depends on the level of assistance needed by the patient, and also about temporal aspects of the reasoning such as *patientBeingStableSince, startOfWeaningObservation*.

The expert also manages various thresholds needed to qualify correct ventilation (e.g., *frequencyThreshold,* the respiratory frequency must be below this threshold) .
The expert object may also have to change the duration of the data acquisition performed to update the physiological values before any new expertise.
All these messages are used in the conditions parts of the rules that implement the expert's knowledge.
The ventilationExpert 's actions are also represented in terms of Smalltalk methods.
We distinguish between four kinds of actions:

- Setting of a diagnosis: *respiratoryCheckUp* method
- Modification of the ventilator settings: the *increaseAssistance* method (resp. *decreaseAssistance*) is used to raise the level of pressure support (resp. decrease), and the *switchToMechanicalMode* method changes the mode of assistance (pressure support) to controlled mechanical ventilation,
- Anticipation of the next consultation: the *nextEvaluationIn: aTime* method sets the duration (aTime) of the next data acquisition for the signalProcessor,
- Predictions for the weaning of the patient: the methods *predictionForWeaning, weaningObservationTime,* and *instabilityTolerance* are defined for this purpose.

III.1.2. Rules

Data-driven methods to generate therapeutic suggestions are generally well adapted to represent the clinician's reasoning (11, 19).
In our system, the knowledge of the expert is indeed represented by first-order (i.e., with variables) production rules, operating in forward-chaining. Since, as already stated, the Smalltalk environment provides no inference facility, we use the NéOpus system, which basically adds a production rule facility to the Smalltalk language. The NéOpus system (16, 17) is a rewriting of the Opus system (15). It adds several new extensions to the original Opus system that are primarily used to represent control knowledge (see next section).
The main characteristics of the NéOpus system are to combine the advantages of the Smalltalk-80 language in the rules 1) by allowing any Smalltalk object to be matched by a rule, and 2) by allowing any Smalltalk expression in the condition and conclusion parts of the rules
The rules consist of a declaration part, in which variables representing Smalltalk objects are declared according to their class; a condition part (yielding boolean results) and an action part. Condition and action parts are expressed as Smalltalk expressions.

In our system, each rule has only one variable, that is matched against an expert object (an instance of the *VentilatorExpert* class, describe above). The rule conditions are designed to test the values of the physiological data or information about the patient, using the messages described in the *VentilatorExpert* class (see preceding section). The rule actions set the expert objet in motion thereby leading to side-effects, such as modifications of the overall check-up and adaptation of the assistance.

III.1.3. Examples of rules

Here are some examples of rules taken from our knowledge base.
In the first example, the goal is to appreciate the quality of the patient's ventilation. Firstly, the expert checks the values for PCO_2 and tidal volume. If they are within physiological limits, the expert decides to test respiratory frequency.
This is represented by the *frequencyTest* rule (Fig. 2). The physiological limits are defined by two methods: *pco2Threshold* and *volumeThreshold*. The decision to test the frequency corresponds to an action by the expert (message *ventilation:*).

When the decision to test the frequency has been taken, there is a possibility of polypnea. The `polypnea` rule defines the conditions under which the patient is polypneic in the same way as the *frequencyTest* rule.
In this example, it is clear that the *polypnea* rule should be fired after *frequencyTest* rule. Rule sequencing is defined, among other things, by the rule protocol (Fig.2, in parenthesis). The notion of protocols is discussed in §4.

```
frequencyTest (ventilation protocol )
"tests the physiological parameters that qualify the ventilation"
        |VentilationExpert expert|

        expert patientPco2 <= expert pco2Threshold.
        expert patientVolume >= expert volumeThreshold.
actions
        expert ventilation: #frequencyTest.
        expert report: 'Now, I am going to test the frequency'.
        expert modified.

polypnea (frequencyTest protocol)
        |VentilationExpert expert|

        expert patientFrequency > expert frequencyThreshold.
        expert patientFrequency < expert maxFrequency.
actions
        expert ventilation: #polypneic.
        expert report: 'The patient is polypneic. Resp. rate too high'.
        expert modified.
```

Fig. 2. Example of rules to test the quality of ventilation.

The next two rules determine whether an action must be taken on the ventilator, and the duration of the next data acquisition.

```
observation (ventilatorAction protocol)
"observation of the patient, no modification of the mechanical assistance"
        |VentilationExpert expert.|

        expert respiratoryCheckUpIsStable.
        expert ventilationIsNormal.
        expert periodOfStability < expert observationDuration.
actions
        expert stability: #observation.
        expert nextEvaluationIn: 2.
        expert incrPeriodOfStability.
        expert report: 'The patient is correctly assisted'.
        expert modified.
```

```
incrementAssistance (ventilatorAction protocol)
"tests whether action must be taken on the ventilator"
        |VentilationExpert expert.|

        expert stabilityIsErratic.
        expert ventilationIsPolypneic.
actions
        expert respiratoryCheckUp: #Unstable.
        expert nextEvaluationIn: 5.
        expert increaseAssistance.
        expert report: 'I increase the assistance.'.
        expert modified.
```

Fig.3. Example of rules to determine whether action must be taken on the ventilator .

These rules are independent of any control strategy. The next problem is to represent adequately the control linked to this rule base. This is done by means of meta-rules that control the firing of the rules.

III.2. Control knowledge

Control of the reasoning is a complex part of the expert's knowledge. In our system, control knowledge is far from trivial because it involves *sequencing*, which is inherently difficult to represent in a rule-based scheme (20), as well as the notion of *alarms*.
An appropriate architecture is therefore needed to represent the *control knowledge* which directs the reasoning. Such declarative architecture of control is represented in NéOpus by two features: the meta-rules (21) and the rule base inheritance.

III.2.1. Structure of rule bases

Rules are organized into rule bases, which correspond to Smalltalk classes. Rules are defined as methods for their rule base, but are parsed and compiled with a particular compiler into a Rete network (22) for efficiency.
The management of rules is thus similar to the management of methods, and the rules that make up a base are grouped into protocols.
In our application, the rule base *ClinicRules* contains all the rules relevant to the expertise of the clinician. In this base, nine protocols are defined representing various knowledge units of this expertise. For instance, protocol *severity* groups together the rules relating to the overall appreciation of the patient's state; *ventilation* includes the rules for the characterization of the present respiratory pattern; *stability*, those that assess the evolution of the ventilation with time; *ventilatorAction,* those for defining the appropriate action on the ventilator; and *weaning* the rules for the implementation of a special observation phase designed to permit weaning of the patient. Protocols may be considered as independant clusters of knowledge.
By classifying the rules in this manner, we obtain a method for partitioning knowledge in a way that reflects human expertise in the domain concerned.
However, protocols are not simply containers used for organizational purposes. On the contrary, they are used by the control architecture to represent the sequential aspect of the reasoning, thanks to the application of appropriate meta-rules.

III.2.2. Meta-rules control rule firing

The rule base *ClinicalRules* is intended for activation in forward chaining. When a set of objects matches all the premises of a rule, this rule is said to be fireable, and is added to a conflict set. In this scheme, a standard cycle consists in firing rules until the conflict set is

empty (22). However, we have mentioned above that the control involved in our expertise is not trivial. To implement this control without modifying the *ClinicalRules* rule base, which represents the knowledge domain, we used the declarative architecture for control of NéOpus. In NéOpus a rule base can be controlled by another rule base, which is called its *meta-base*. The rules governing the meta-base are called meta-rules. The meta-rules which syntactically do not differ from standard NéOpus rules, activate the rule base linked to the meta-base. The objects matched against the meta-rule premises are particular objects called *ruleBaseEvaluators* . Fuller details on the use of meta-rules can be found in (21). The interesting aspect of these meta-rules is that they allow domain knowldege and control knowledge to be defined independently, thereby increasing the generic nature of our system (see §IV).

Defining a particular control consists in writing meta-rules which directly affect on the activation of the controlled rule base.

Here are some examples of the use of meta-rules to represent our control strategies:

Sequencing: The expert's reasoning has a sequential structure. For instance, stability must be estimated after the current ventilation has been characterized. To represent this sequencial structure we use the Smalltalk protocols described in III.2.1. A special set of meta-rules is designed to handle the sequencial character of the reasoning by organizing a list of protocols and firing rules according to this list.
When the system is initialized, the list of protocols is initialized (*listOfProtocols* <- #(#Severity #Ventilation #Stability #VentilatorAction)).
However, this list may vary, depending on the status of the patient, as ascertained during the consultation.
For example (Fig. 4), the *afterVentilatiorAction* meta-rule represents the following sequence: the appropriate action on the ventilator is performed (ventilatorAction protocol) followed by the selection of a particular protocol list. The *listProtocolsEnd1* list (which includes the weaning protocol) is selected in case of a favorable prediction for the weaning. If the prediction for weaning is not favorable, this protocol is ignored and another list is selected (*listProtocolsEnd2*).

```
afterVentilatorAction
  | RuleBaseEvaluator e. OpusConflictSet c|
        e status = #loop.
        c <- e conflictSet.
        e agenda currentProtocol = #ventilatorAction.
        c hasRuleInProtocol: #ventilatorAction.
actions
        c fireRules: (c rulesInProtocol: #ventilatorAction)
        e agenda changeProtocolList:
        (e expert predictionForWeaning = #favorable
                ifTrue: [Clinic1 listProtocolsEnd1]
                ifFalse:[Clinic1 listProtocolsEnd2]).
        e modified.
```

Fig. 4. This meta-rule changes the list of protocols according to the patient status.

Alarms management: Alarms are actions that must be taken regardless of other pending actions or reasoning. The alarm problem is solved by the use of specific meta-rules. For instance (Fig. 5) is a meta-rule that will detect any rules that can be fired from protocol *alarms*. On detection, such rules are instantly fired.

```
alarmsHandling
  | RuleBaseEvaluator e. Local rules conflictSet |
        e status = #loop.
        conflictSet <- e conflictSet.
        rules <- conflictSet rulesFromProtocol: #alarms.
actions
        Transcript show: 'Alarm. Check the patient'.
        conflictSet fireRules: rules.
        e suspend. conflictSet modified. e modified
```

Fig. 5. Alarms management.

III.2.3. Rule base inheritance

The rule base inheritance mechanism allows rule bases to be represented in an inheritance tree, very similar to the usual class inheritance trees (14). This mechanism provides a means of organizing rules according to a generalization/specialization scheme. Rules defined in super-bases are inherited by all sub-bases, thus enabling rule re-use. In addition, the rules defined down in the hierarchy are considered *more specific,* and will be preferred for conflict resolution.
We used this inheritance mechanism to organize meta-rules. Instead of defining a single meta-base to represent control knowledge, we split this base into several hierarchical meta-bases, using the rule base inheritance mechanism. We distinguished five levels of meta-base, each defining a particular control structure.
In this way we implemented the following inheritance tree: *DefaultMeta*, *MetaProtocols*, *MetaProtocolsOneShot*, *MetaClinicProtocols*, *MetaClinicAlarms.* The *ClinicRules* rule base is controlled by the *MetaClinicAlarms* meta-base. This meta-base is in charge of the activation of the *ClinicRules* base, and contains all the control knowledge of the expert. This inheritance tree is represented graphically in Fig. 6.

DefaultMeta, MetaProtocols, and *MetaProtocolOneShot* are general and not particular to our application. The meta-base *MetaProtocolOneShot*, a sub-base of `MetaProtocol` (itself a sub-base of *DefaultMeta*) manages the notion of protocols for rule firing. The rules are fired according to a *fixed list of protocols* given by the user. With this structure we implement static sequencing.
This static sequencing has to be changed in some cases, to adapt it to the context (see the meta-rules in fig. 4). The *MetaClinicProtocols* meta-base, a sub-base of *MetaProtocolsOneShot*, is peculiar to our expertise and implements the dynamic sequencing. These meta-rules must be activated before the meta-rules of *MetaProtocol. MetaClinicProtocols* is therefore a sub-base of *MetaProtocols.*
Alarms are preferable to other pending actions. Consequently the notion of alarming cases is well represented by the lowest meta-base in the hierarchy and its meta-rules (included in the *MetaClinicAlarms* meta-base) will be fired in preference to other meta-rules defined in higher meta-bases.

IV. DISCUSSION

The medical expertise involved in the process of reasoning we have described can be represented with production rules, as we showed previously with the Ganesh prototype (11). We designed NéoGanesh using Smalltalk with NéOpus to solve certain knowledge representation problems and obtain an open environment to develop specific tools adapted to clinical expertise. We now discuss the generic aspect of our system.

IV.1. The use of Smalltalk

The system benefits from 1) the powerful features of the Smalltalk environment (browsers, debugger) and 2) from its portability on different platform window systems (Xwindows, Windows, OpenWindow, and MultiFinder), as well as an active user community.

Our prototype has to work in real time. The fact that Smalltalk may be slower than C for some numerical applications (23) is not a problem for us, for two reasons: our system does not require intensive computations, and our time constraints are weak because ventilation processes are slow (respiratory rate is less than 0,5 Hz). The sample frequency used for data acquisition is 0.1 Hz. In alarming cases, a response time of 1 or 2 seconds is tolerated. The sophisticated management of memory implemented in this version of Smalltalk is therefore correct for our application because no inopportune garbage collecting disturbs the working process.

The user's interface is an important element of a computerized system and the MVC paradigm provided us with good tools for tailoring the interface to the clinical staff's specific needs.

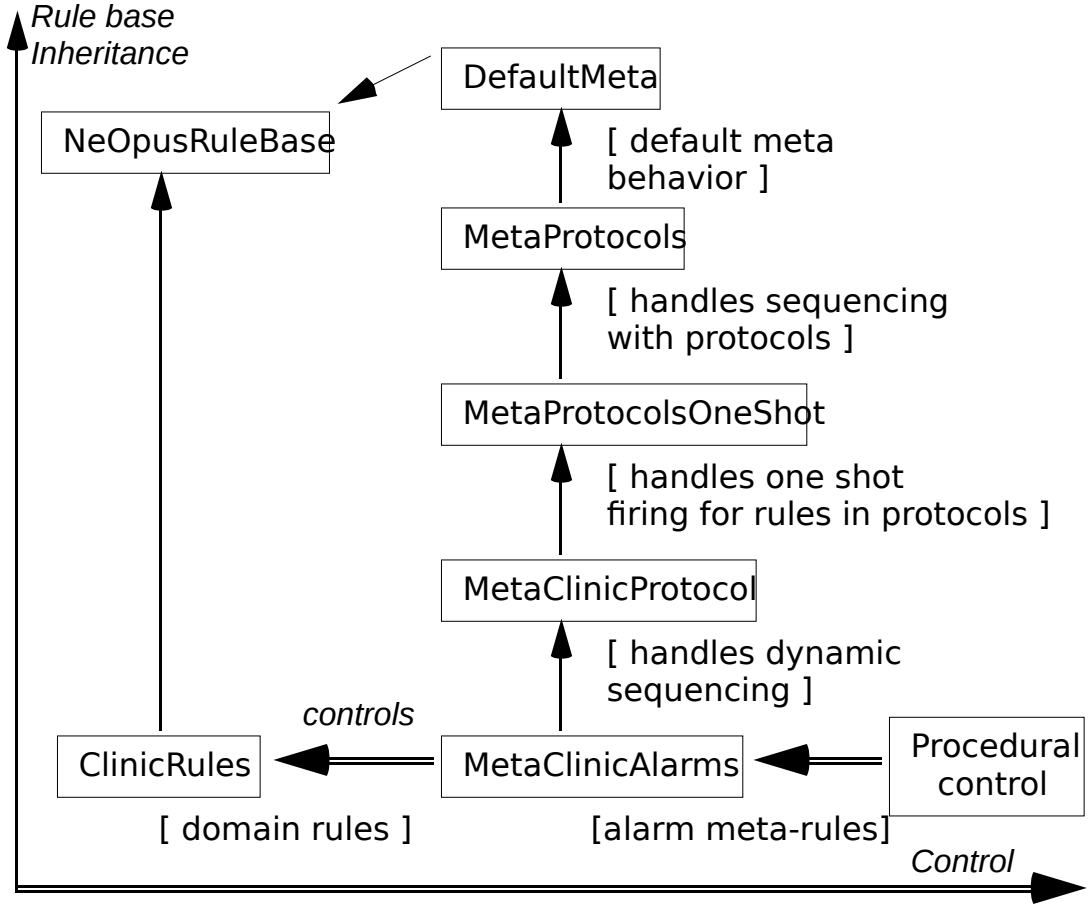

Fig. 6. Inheritance tree architecture

IV.2. Towards a generic environment

In our opinion, three major aspects of our system make it generic and suitable for designing medical knowledge-based systems.

1) **Class inheritance**
Object orientation gives us the advantage of the flexibility and reusable code. For instance, we created the *Device* class to represent the devices connected to a patient. Because of the lack of standardization, the interface of each device has to be specified individually. Subclasses such as *Monitor* and *Ventilator* were introduced to represent particular devices. Similarily, the programmer describes new devices by defining *Monitor* or *Ventilator* subclasses. For instance, *Veolar* (a subclass of Ventilator) and *Ohmeda* (a subclass of *GasMonitor*, itself a subclass of *Monitor*) were created for the two types of apparatus we need for the clinical application of NéoGanesh. Thanks to inheritance only the protocol of communication has to be redefined for the new device class.
Inheritance can also be used to modify or refine the expert's behavior. Thus, several subclasses of the *VentilationExpert* class can be created to represent various interpretations of the messages used in the rules. For instance, the *pco2Threshold* method (Fig. 2), which yields a maximum threshold for PCO_2 may be redefined. In the *VentilatonExpert* class, this method is defined so as to return 50. In subclasses, this method may be redefined to yield an other value. Subclasses of *VentilationExpert* can also include additional specific structures (i.e. instance variables) to account for particular types of expertise.

2) **Rule base inheritance**
The rule base inheritance is a powerful feature of NéOpus, and may be used to refine or modify the expert's knowledge. For instance, to implement a new weaning reasoning, a sub-base of the *ClinicalRules* rule base (say *ClinicalRules2*) will be created. In this sub-base, only the rules from the *Weaning* protocol will be modified.

3) **Generic aspect of the control reasoning**
The control reasoning is implemented without modifiying the knowledge domain in a very elegant way; it is not embedded in the code thereby allowing a clear separation between knowledge and control. As the control is represented by meta-bases, it can be re-used with other domain rule bases (for instance *ClinicalRules2* defined above). If a new reasoning strategy is needed, it can be obtained by linking a more appropriate meta-base to the rule base concerned.

4) **Multi-tasks and multi-expertise**
In its current state, the system is sequential, and only one task is executed at a time. This can give rise to difficult problems. For instance, an alarm can occur while the system is reasoning. An interesting extension to our system is that it allows different tasks such as checking alarms, data acquisition and processing, and reasoning to run concurrently.
Another function of multi-tasking is to implement a distributive problem-solving structure (24). In this scheme, several experts cooperate to find the solution by sharing and exchanging information. For instance, one expert may be may be specialized in ventilation, and another, in the sensor control.
Multiprocessing primitives are available in the Smalltalk-80 environment in the form of *Process* and *Semaphores* classes. Actalk (25) is an extension of Smalltalk that provide concurrent facilities. Its integration into our system is now in progress.

The open environment we have proposed is well adapted to research purposes and has allowed us to test some new concepts designed to improve the representation of clinical expertise in intensive care.

V. CONCLUSION AND FUTURE WORK

NéoGaneh is going to replace Ganesh in the ICU of H. Mondor Hospital. Our aim is to use it to implement more complex expertises than the expertise present in Ganesh. To explore the creation of a general architecture for knowledge-based systems that would support clinical procedures in intensive care and anesthesia, our environment must be extended so as to permit the handling of new paradigms designed to improve knowledge representation. Temporal logic must be introduced to manage the patient's history correcty. Uncertainty will be also useful, for instance to improve the representation of physiological thresholds. The extension of the present system to a multi-expert environment is now in progress in order to develop asynchronous communications between objects.

VII. REFERENCES

1- MILLER P.L. "Selected topics in medical artificial intelligence". New York, Springer Verlag, (1988).

2- CLANCEY W.J., SHORTLIFFE E.H. (eds) "Readings in medical Artificial Intelligence : the first decade". Readings, MA, Addison Wesley, (1984).

3- BOYER F., BRUNO B., GAUSSAGUES P., JAS-LAMONNERY S., ROBERT D. "Aide inspiratoire avec asservissement du niveau de pression : volume ventilé minute versus fréquence respiratoire". Réan. Soin. Intens. Med. Urg. vol. 5, n° 4, 227-232, (1989).

4- BROCHARD L., PLUSKWA F., LEMAIRE F. "Improved efficacy of spontaneous breathing with inspiratory pressure support". Am. Rev. Resp. Dis. 136, 411-413, (1987).

5- MAC INTYRE N. "Respiratory function during pressure support ventilation". Chest, 677-683, (1989).

6- FAGAN L.M. "Representing time dependent relations in a medical setting". PhD dissertation, Dept. of Computer Sciences, Standford University, Palo Alto, Ca., (1980).

7- HERNANDEZ-SANDE C., MORET-BONILLO V. ALLONSO-BETANZOS A. "ESTER : An expert system for management of respiratory weaning therapy". IEEE Trans. Biom. Eng. vol. 36, n° 5, 551-564, (1989).

8- RUDOWSKI R., FROSTELL C., GILL H. "A knowledge-based support system for mechanical ventilation of the lungs. The KUSIVAR concept and prototype". Comput. Biomed. Res. vol. 30, 59-70, (1989).

9- SITTIG D.F. "A computerized patient advice system to direct ventilatory care". PhD dissertation, Dept. of medical informatics, University of Utah, (1986).

10- BROCHARD L., HARF A., LORINO H., LEMAIRE F. "Inspiratory pressure support prevents diaphragmatic fatigue during weaning from mechanical ventilation". Am. Rev. Resp. Dis. 139, 513-521, (1989).

11- DOJAT M., BROCHARD L., HARF A. "A knowledge-based system for the management of the weaning procedure of mechanically ventilated patients". Proceedings of 12th International Symposium on Computer Assisted Decision Support and Data Base Management in Anesthesia Intensive care and Cardiopulmonary Medecine. Rotterdam, October 2-4, (1991).

12- DOJAT M., HARF A., BROCHARD L. "Evaluation d'un système d'aide à la décision pour le sevrage des patients ventilés artificiellement". XXème Congrès de la Société de Réanimation de langue Française, Paris, January 16-18, (1992).

13- DOJAT M., PACHET F. "Representation of a Medical Expertise Using the Smalltalk environment: putting a prototype to work". Proceedings of TOOLS 7, Dortmund, Germany, March 31-April 2, (1992).

14- GOLDBERG A. "Smalltalk-80: The interactive programming environment". Addison-Wesley, (1984).

15- LAURSEN J. , ATKINSON R. "Opus: A Smalltalk Production System". OOPSLA '87 377-387, (1987).

16- PACHET F. "NéOpus mode d'emploi". Rapport LAFORIA, n° 14/91, Paris, (1991).

17- PACHET F. "Reasoning with objects: the NéOpus environment". Proceedings of EastEurOOpe conference, Bratislava , September, (1991).

18- TOBIN M.J., PEREZ W., GUENTHER S.H. SEMMES B.J., MADOR M.J. ALLEN S.J., LODATO R.F., DANTZKER D.R. "The pattern of breathing during successful and unsuccessful trials of weaning from mechanical ventilation". Am. Rev. Respir. Dis. 134, 1111-18, (1986).

19- SITTIG D.F. "Clinical evaluation of computer-based respiratory care algorithms". Int. Jour. Clin. Mon. Comp. vol. 7, 177-185, (1990)

20- GOMEZ F., CHANDRASEKARAN B. "Knowledge organization and distribution for medical diagnosis". IEEE Transactions on systems, man, and cybernetics, vol. smc-11, n°. 1, 34-42, (1981).

21- PACHET F. "Du bon usage des méta-règles en NéOpus". Rapport LAFORIA, n° 16/91, Paris, (1991).

22- FORGY C. L. "Rete : A fast algorithm for the many pattern/many object pattern match problem". Artificial Intelligence, vol. 19, 17-37, (1982).

23- DEUTCH P.L. "The past , present, and future of Smalltalk". Proceedings of ECOOP '89, pp. 109-130, Cook, London, (1989).

24- BOND A. “Readings in distributed artificial intelligence” San Mateo, Ca, Morgan Kauffmann, Grasser L. eds., (1988).

25- BRIOT J. P. "Actalk: a testbed for classifying and designing actor languages in the Smalltalk-80 environment". Proceedings of ECOOP '89, 109-130. Cook, London, (1989).

NÉOGANESH : VERS UN SYSTEME À BASE DE CONNAISSANCES GENERIQUE POUR LE CONTROLE DE LA VENTILATION ARTIFICIELLE DES PATIENTS.

M. DOJAT, F. PACHET

RESUME :

Nous décrivons un système à base de connaissances en cours de validation à l'hôpital Henri Mondor (Créteil, France). Le système NéoGanesh est destiné à adapter l'assistance ventilatoire mécanique fournit à un patient insuffisant respiratoire selon ses besoins physiologiques. Le système acquiert des données physiologiques en temps réel, établit un diagnostic, et agit directement sur les réglages du ventilateur connecté au patient. Il développe un raisonnement pour sevrer le patient du ventilateur. Le malade est ventilé avec une aide inspiratoire en pression. Nous utilisons l'environnement de programmation à objets Smalltalk -80 pour représenter les objets du domaine. Un moteur d'inférences d'ordre 1, NéOpus, écrit en Smalltalk, est utilisé pour implémenter le raisonnement. L'architecture complète de contrôle du déclenchement des règles est présentée. Nous insistons finalement sur les aspects génériques de notre système. Intégré dans un environnement sophistiqué, NéoGanesh constitue un premier pas vers un système générique de représentation d'expertises médicales utilisées en réanimation et anesthésie.

MOTS CLES : Expertise médicale - système à base de connaissances - ventilation mécanique - sevrage.

NÉOGANESH: TOWARDS A GENERIC KNOWLEDGE-BASED SYSTEM FOR THE CONTROL OF MECHANICAL VENTILATION.

ABSTRACT:

We describe a working rule-based prototype now being tested at the Henri Mondor Hospital (Créteil, France). The system NéoGanesh is designed to adapt the mechanical assistance provided to a patient with respiratory insufficiency to his physiological needs. NéoGanesh monitors physiological data in real time, establishes a diagnosis and acts directly to modify the settings of the ventilator connected to the patient. It implements a strategy to wean the patient from the ventilator. The patient is ventilated with Pressure Support Ventilation. We chose the object-programming environment Smalltalk-80 to represent all the domain objects. The rule-based system NéOpus, a first order foward chaining engine embedded in Smalltalk, is used to implement the reasoning. The complete architecture that controls the firing of the rules is described. We consider the generic aspect of our system by showing how new expertise can be modelled from the existing one. Embedded in a powerful environment, our system is intended to constitute a first step towards the creation of a generic system designed to represent medical reasoning used in intensive care and anesthesia.

KEY WORDS: Clinical expertise - knowledge-based system - mechanical ventilation - weaning.