Dojat M. et al. NéoGanesh: a Working System for the Automated Control of Assisted Ventilation in ICUs, Artif. Intell. in Med., 11 (1997) 97-117.

NéoGanesh: a Working System for the Automated Control of Assisted Ventilation in ICUs

M. DOJAT[a], F. PACHET[b], Z. GUESSOUM[b], D. TOUCHARD[a], A. HARF[a,c], L. BROCHARD[a,d]

[a]Institut National de la Santé et de la Recherche Médicale, U.296, Faculté de Médecine 8, rue du G[al] Sarrail 94010 Créteil Cedex, France
[b]Laboratoire Formes et Intelligence Artificielle, LAFORIA-IBP, Université Paris 6, Boite 169, 4, Place Jussieu, 75252 Paris Cedex, France
[c]Département de Physiologie; [d]Service de Réanimation Médicale, Hôpital Henri Mondor, Créteil, 51, avenue de M[al] de Lattre de Tassigny 94010 Créteil Cedex, France.

Address for correspondence: M. Dojat
INSERM U296 Faculté de Médecine
8, rue du Général Sarrail
94010 Créteil Cedex FRANCE
phone: 33 1 48 98 46 03
fax: 33 1 48 98 17 77
email: dojat@im3.inserm.fr

Abstract:

Automating the control of therapy administered to a patient requires systems which integrate knowledge of experienced physicians. This paper describes NéoGanesh, a knowledge-based system which controls, in closed-loop, the mechanical assistance provided to patients hospitalized in Intensive Care Units. We report on how new advances in knowledge representation techniques have been used to model the medical expertise. The clinical evaluation shows that such a system discharges the medical staff from routine tasks, improves the patient's care, and efficiently supports medical decision regarding weaning. To be able to work in closed-loop and to be tested in real medical situations, NéoGanesh deals with a voluntarily limited problem. However, embedded in a powerful distributed environment, it is intended to support future extensions and refinements and to support reuse of knowledge bases.

Keywords: Intensive-Care Monitoring, Knowledge Representation, Distributed Architecture, Closed-Loop Control, Mechanical Ventilation.

1. INTRODUCTION

There is a growing need for computerized systems able to assist the clinical staff in decision making, especially in medical environments such as operating rooms or intensive care units (ICUs), where the flow of information is abundant, false positive alarms are common and life-threatening situations should be prevented. These *intelligent patient monitoring systems* must reason about complex situations under real-time constraints such as resource limitations and guarantee of timely response. Building such systems is a challenging goal for the emerging research area of real-time Artificial Intelligence (AI) [31] and more specifically 'adaptive intelligent systems' [22]. This paper describes an intelligent patient monitoring system for the automatic control of mechanical ventilation.

For a number of reasons, including the complexity of medical reasoning, interference from noise, considerations of liability, and social and cultural factors, most of intelligent patient monitoring systems are *open-loop systems* with respect to planning and control. Sepia [38] for monitoring patients hospitalized in hemato-oncology departments is a good example of such systems. However, there are specific well-defined medical problems, in particular planning drug therapy [11] where *closed-loop systems* can be proposed. Such closed-loop systems can further improve the management of patient's care, because they operate continuously on a daily basis.

Computers have been used in clinical practice for traditional tasks such as data-base management, data acquisition and physiological signal processing. Sophisticated systems 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 of them are routinely used in clinical practice. '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.

Five years ago in collaboration with the ICU of the Henri Mondor Hospital (Créteil, France), we started a project to explore the feasibility and the clinical interest of a system providing automated control of mechanical ventilation and decision for extubation. We decided that such an intelligent system should be able to 1) handle in real-time a huge mass of information about the patient's state, 2) diagnose observed situations, 3) predict the evolution of the patient's state, 4) construct action plans with prompt reaction in alarming cases and 5) execute planned actions.

In this paper we report how new advances in knowledge representation techniques, in particular the association of the object-oriented paradigm with production rules, temporal reasoning and distribution of knowledge, have been used to design the system. We report on the clinical results obtained with our prototype, called NéoGanesh, which has been used to ventilate a large number of patients in our ICU. To be able to work in closed-loop and to be tested in real medical situations, NéoGanesh deals with a voluntarily limited problem. However, embedded in a powerful distributed environment, it is intended to support future extensions and refinements and to allow reuse of the knowledge bases developed so far.

Our paper is structured as follows. Section 2 defines the medical problem, details the different levels of control for the management of mechanical ventilation and briefly outlines several systems that address a similar problem. Section 3 describes the main characteristics of the NéoGanesh system. Section 4 and Section 5 respectively detail the distributed architecture of NéoGanesh and how medical expertise is modeled. Section 6 is devoted to the clinical results. Finally, we conclude on the interest of our approach and discuss future extensions of NéoGanesh.

2. THE CONTROL OF MECHANICAL VENTILATION

The mechanical assistance provided to a patient with respiratory insufficiency must be well adapted to his or her physiological needs. The clinician must assess the *respiratory comfort* of the patient and the time-course of his or her ventilation and must set the ventilator parameters accordingly. A second task of the clinician is to reduce mechanical assistance gradually until the patient is able to breathe alone. This task is known as the *weaning process*.

During weaning from mechanical ventilation, the respiratory muscles of the patients, who are often weakened by denutrition, sepsis, disuse atrophy or electrolyte disorders, have to bear high workloads, due to lung infection, high airway resistance and the presence of the endotracheal tube and the ventilator circuit. Therefore, in most instances, failure of weaning from mechanical ventilation results from the inability of the respiratory muscles to cope with the imposed workload. This is why modes of ventilation designed to support respiratory muscle work have been developed and used as means of gradually separating the patient from the ventilator [3, 4, 30]. In the Pressure Support Ventilation (PSV) mode, the patient triggers the assistance automatically by his inspiratory effort and switches it off on expiration. Since its introduction, PSV has proved to be a useful and practical mode of ventilation. As a result, its physiological effects on the breathing pattern have been described more extensively than for any other mode of partial respiratory support. Some guidelines have been proposed to use PSV as a full respiratory support and the level of pressure needed by the patient has been proposed as a guide to decide for the right moment to perform extubation [5, 19]. In addition, a protocol allowing gradual withdrawal from mechanical ventilation using this mode has proved to be particularly useful in difficult-to-wean patients [18]. At best, these strategies need a constant monitoring of the ventilated patient, in order to adjust continuously the ventilator settings to the evolution of the patient's respiratory state. For this reason, they may be hard to use in clinical

practice. To fill this task, we have designed NéoGanesh, a knowledge-based supervisor which controls, in closed-loop, the ventilation in the pressure support mode.

2.1. Different Levels of Control

Recent developments in methods of ventilation and computer technology have made closed-loop control of ventilation feasible and have the potential to make ventilation and weaning safer and more comfortable [29, 42]. In the context on mechanical ventilation we identified three levels of control (see Fig. 1). The complexity of the levels and the response time increase from the lowest to the highest level of control. Each level controls the levels below and is controlled by the levels above:

- The first level (L1) is the generation of the assistance. L1 is the basic loop of each ventilator: it controls the shape of the flow or the pressure sent to the patient by driving a servo-valve. L1 is designed using methods of classical control theory and relies on mathematical models of the physical components of the ventilator. L1 is highly reactive (response time ≈ 1ms.).
- The second level (L2) determines the mode of ventilation. L2 monitors a physiological parameter and uses it as a variable for the servo-control of a parameter of the ventilator. For instance, the adjustment of the level of PSV on the ventilator maintains either minute volume or spontaneous frequency at a specific target level fixed by the clinician. The response time of L2 is approximately of 1 or 2 cycles (few seconds).

  Several modalities of automatic control of PSV have been proposed, such as those of Hamilton and Taema, with Veolar (MMV: Mandatory Minute Ventilation) and César (VAIV: Ventilation en Aide Inspiratoire Variable VAIV) ventilators respectively, both of them based on conventional algorithms. More sophisticated automatic controllers have been proposed (ARIS [7], PAV [47], ALV [28]).

  All these controllers are constructed on a fixed, more or less complex, physiological model of the patient using mathematical relations between physiological parameters. Clearly, these models cannot be applied to all pathologies and alarming situations

where the underlying assumptions of these models are generally no more valid (for instance some parameters considered as constant in the model change over time with the disease's evolution). The algorithms they are based on do not reflect the attitude of the physician who adapts the therapeutic strategy over time.

- The third level of control (L3) is the adaptation of the assistance, using information to define the current state of the patient and its evolution. This level of control is traditionally realized by the physician and is based on specific medical knowledge and is driven by therapeutic strategies. The response time in this level varies from a few seconds in alarming situations to a few minutes in routine patient observation.

L1 and L2 may be directly assimilated to *low-level control* and L3 to *high-level planning* [9].

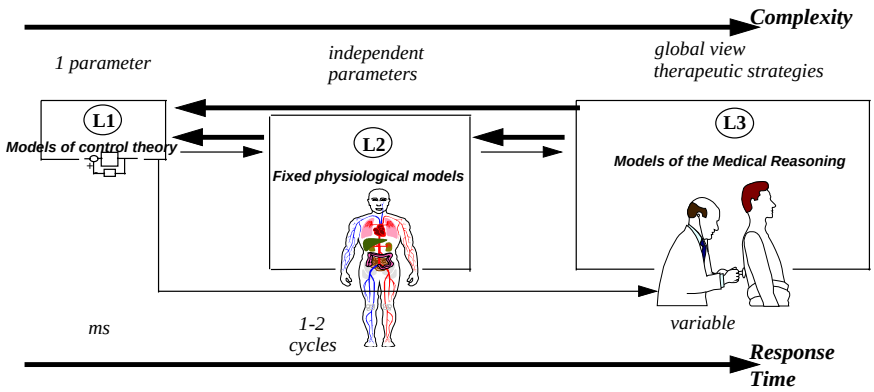

Fig. 1: Different levels of control.
Thick arrows indicate the control from the highest levels over the lowest levels. Thin arrows indicate the information flux (alarms, acknowledgments, ...) which goes through the hierarchy.

In order to perform a diagnosis, determine a therapy and act on the ventilator, we claim that a substantial part of L3 should be integrated in the overall system. Indeed, integrating L3 in the system allows to build a more comprehensive view of the time course of the patient's state, thereby giving it the ability to manage several ventilation strategies depending on the patient's state. We show in this paper that a knowledge-based approach is suitable to model a substantial part of the clinician's expertise relevant for L3.

2.2. Related research

Numerous systems have been proposed in the literature for the management of mechanical ventilation. They may be classified in two categories:

- the '*Specialists*' try to solve a specific clinical problem. These systems use clinical heuristics and may introduce mathematical models of physiological functions. Reusing these systems in another context could be difficult. VentEx [39] that recommends changes in ventilator settings and Weanpro [45] for the weaning of patients, are recent members of this category.
- the '*General architectures for intelligent monitoring*' explore new techniques or concepts which might be integrated in future systems. The short term application of the designed prototypes is not envisaged. VentPlan [37], which combines qualitative and quantitative computation in a ventilator-management advisor and Guardian [23], an intelligent agent based on a specific control architecture to satisfy real-time constraints, illustrate this second category of systems.

NéoGanesh lies at an intermediate position between these two categories: on one hand the goal is to solve a precise clinical problem and to use the system obtained at the patient's bedside; on the other hand the system is constructed on a distributed architecture using specific knowledge representation techniques and temporal abstractions to facilitate future extensions and reuse of knowledge bases.

3. THE NÉOGANESH SYSTEM

NéoGanesh is based on the knowledge of ventilation management acquired by the clinical staff of the ICU at the Henri Mondor Hospital (Créteil, France). In contrast to other systems [23, 37], NéoGanesh deals with a voluntarily limited problem: only one mode of ventilation is managed by the system, i.e., pressure support ventilation (PSV) used to ventilate patients who can have spontaneous respiratory activity. These

limitations allowed us to design a *closed-loop system* that controls the ventilator without any intervention by the clinician and to test it in real clinical situations. The expected advantages of such a system are 1) a 24-hour a day adaptation of respiratory assistance to the needs of the patient, 2) a reduced need of monitoring, 3) better weaning outcomes, and finally 4) a reduction of the duration of mechanical ventilation.

3.1. Architecture of NéoGanesh

The global architecture of NéoGanesh, shown on Fig. 2, is designed on the basis of the three fundamental abstract tasks in medical reasoning: monitoring, diagnosis and therapy planning [35], which may be refined in several subtasks [17].

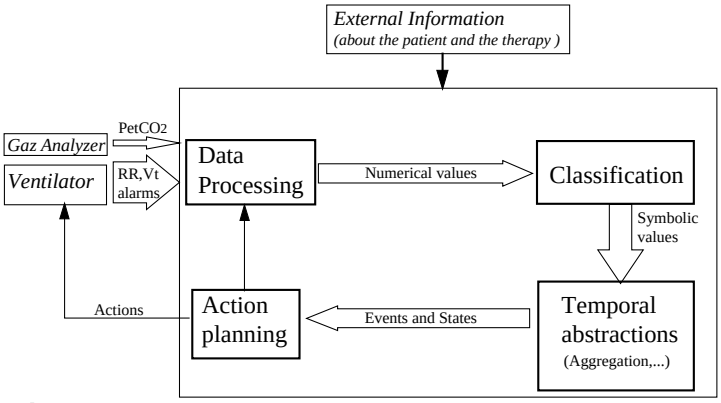

Fig. 2: Architecture of NéoGanesh.
Bold rectangles indicate the different subtasks which constitute NéoGanesh.

NéoGanesh acquires and processes information (data processing subtask) about the patient's respiratory state via three parameters: *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 indication of respiratory muscle adaptation to the imposed workload [44]. The *level of pressure support* provided to the patient is used to evaluate the quality of the respiratory system. All the physiological parameters except one, which is obtained from a CO_2 analyzer (Novametrics 1260, Medical Systems Inc. USA), are measured by the ventilator (Veolar, Hamilton Switzerland) which was especially adapted to our study by the manufacturer. The diagnosis task is performed in two subtasks: 1) processed data are classified

(classification subtask) to diagnose the current respiratory state of the patient and 2) temporal abstractions are then generated (temporal abstraction subtask) to assess the time course of the patient's disease. Finally, the action planning subtask determines the therapeutic actions to be performed on the ventilator. In order to check that the therapeutic actions are effective, the planning subtask informs the data processing subtask that specific parameters should be acquired. The actual comparison between the expected state of the patient, determined by the planning subtask, and the actual state of the patient, as diagnosed by the classification subtask, is performed by the temporal abstraction subtask.

Specific information concerning the patient (morphology, type of pathology, ...) and the therapy envisaged (kinetic of the decreasing of the assistance, ...) are given by the clinician in charge at the initialization of the system.

3.2. User Interface and Implementation

The user interface (see Fig. 3) shows all useful information to the clinical staff i.e. respiratory parameters and ventilator settings. The user interacts with NéoGanesh by clicking with a mouse on the buttons of the display. Any action on the keyboard receives immediate attention. The interface was kept intentionally simple and straightforward, since it has to be used in a real world environment by clinicians who are not necessarily familiar with sophisticated user interfaces.

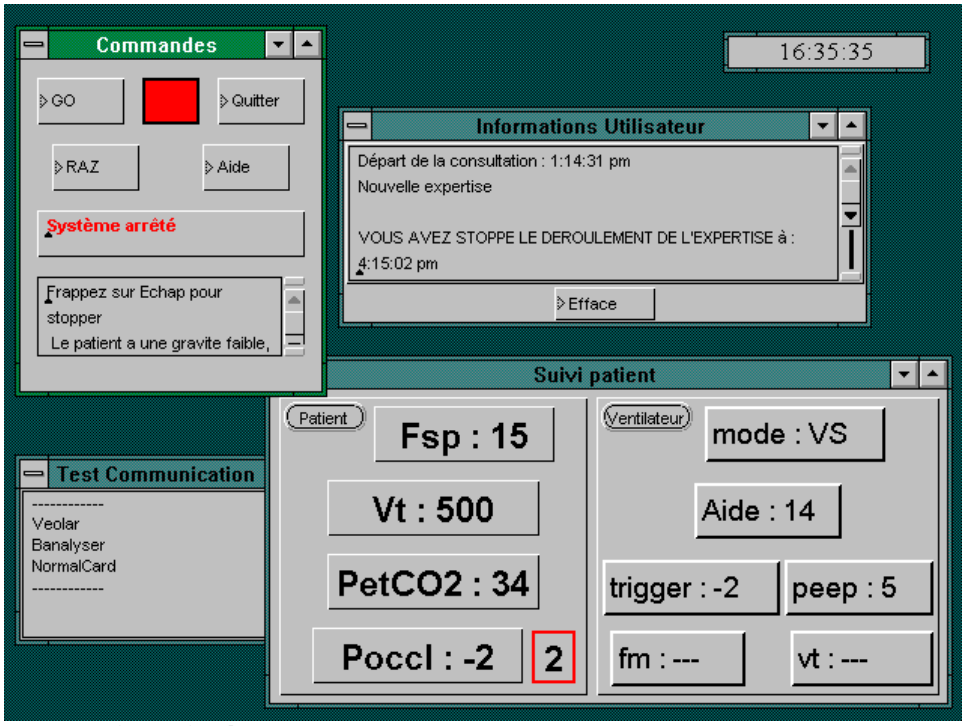

Fig. 3 : The user interface of NéoGanesh.
The window labeled 'Commandes' (Command) is used to start and initialize NéoGanesh. The window labeled 'Information Utilisateur' (Information for the Clinician) gives information about the current state of the expertise. The window 'Test Communication' (Connected Devices) is used for testing the serial communications with the connected devices. The window 'Suivi Patient' (Patient Follow up) consists in two parts:
- in the left part, the physiological parameters of the patient, Fsp = Respiratory rate, Vt= tidal volume, PetCO2= end-expiratory CO_2 pressure and Poccl= occlusion pressure (this parameter is monitored but not used by the knowledge base in the current version). The number of acquired samples for the calculation of a new mean value for each parameter is printed (here equal to 2).
- in the right part, the settings of the ventilator are shown. Fm (machine frequency) and Vt delivered by the machine are nil in the mode chosen i.e. VS, Spontaneous Ventilation with pressure support set to 14 cm H_2O.

NéoGanesh runs on a PC-compatible, is implemented using the Smalltalk-80 language (under Windows 3.1 environment). It uses the NéOpus system, a first-order inference engine embedded in Smalltalk-80 [32] one of the prominent features of which is the declarative specification of control with *metarules* [33]. Every 2 or 5 minutes depending on the last action on the ventilator, NéoGanesh diagnoses the current state of the patient and may perform an action on the ventilator. Physiological data are sampled at the frequency of 0.1 Hz. The average duration of one cycle of reasoning is about 1 second on PC/486 66 MHz. Technical aspects of NéoGanesh, in particular the

satisfaction of real-time constraints, are discussed extensively in [16]. Asynchronous communication between independent agents has been implemented using Actalk [2], an actor language based on Smalltalk-80.

4. A DISTRIBUTED ARCHITECTURE

Distributed Artificial Intelligence and especially multi-agent architecture provides a powerful paradigm for the modeling and the development of complex systems. It is based on the decomposition of systems into several interacting and autonomous entities. Recent applications show the growing interest of this paradigm in the medical domain [21, 27]. According to the definition of Shoham [40] (1993, p.52), "an agent refers to an entity that functions continuously and autonomously in an environment in which other processes take place and other agents exist". To facilitate the design and future extensions of NéoGanesh, we have adopted a distributed architecture based on the multi-agent paradigm. Each task of NéoGanesh is associated to an agent. Two agents correspond to tasks not illustrated in Fig. 2 : `RespiratorManager`, which communicates directly with the ventilator, and `ClinicianAgent` which owns information about the patient, the therapeutic strategy and exchange information with the real clinician. Each agent has specific capabilities and exchanges information by message-passing. The Fig. 4a shows the distribution of the overall expertise between agents. The arrows indicate the default flow of information between the agents in routine mode, i.e. when no particular alarming event occurs. For example, agent `Classifier` diagnoses the current state of the patient, and indicates to the `TemporalReasoner` that the current ventilation is `Normal`.

Communication links between agents are dynamically adapted in response to specific situations. For instance the arrows in Fig. 4b show the flow of information in an alarming situation: an alarm (no inspiration) is detected by the `DataProcessor`, then the `Classifier` diagnoses an apnea and the `ActionPlanner`, after receiving this information, changes immediately the respiratory mode, initially set to PSV, to controlled mechanical ventilation (CMV).

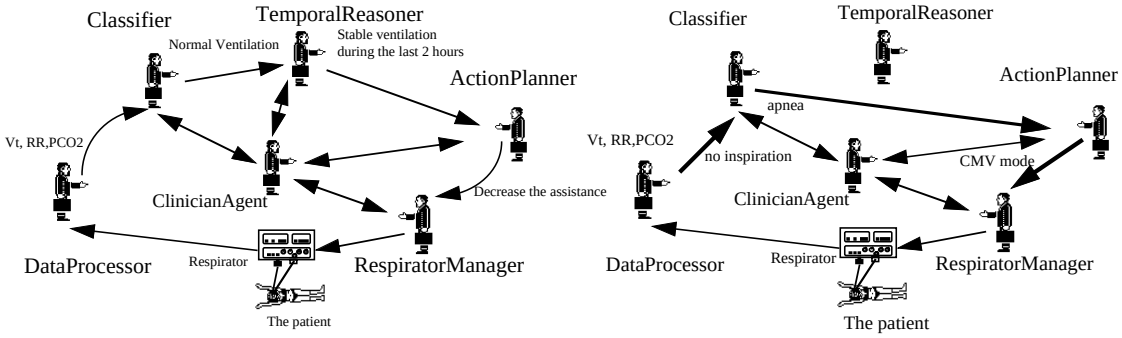

Fig. 4a-4b: Distribution of the medical expertise between agents.
Fig. 4a (left) : Flow of information in routine mode
Fig. 4b (right): Flow of information in an alarming situation

All the agents composing NéoGanesh are based on the model detailed below.

4.1. A Real-Time Agent Model

To be useful for real-time domains such as intelligent patient monitoring, multi-agent systems must (1) handle asynchronous events, (2) manage resource overload and time constraints and (3) ensure efficient control of distributed autonomous entities. Recently, Z. Guessoum [20] has proposed a hybrid agent model which achieves these goals by integrating smoothly so-called *reactive* abilities (to meet hard deadlines) and *cognitive* abilities (to act rationally by using knowledge and to reach a fixed goal when constraints are relaxed). This model (see Fig. 5) defines agents as 1) a collection of up to

three modules and 2) a supervisor that schedules the interactions between modules. All these modules, as well as the supervisor run concurrently and interact asynchronously. Typical modules are the perception module (reactive), the reasoning module (cognitive) and the communication / action module (which can be either reactive or cognitive).

- the *perception* module manages the interactions between the agent and its environment. It monitors sensors, translates and filters acquired data. The obtained data set is mainly used by the reasoning module.
- the *reasoning* module is responsible for generating the adequate answers to the messages transmitted by the communication module or to the changes detected by the perception module. It relies on two kinds of capacities: operative, represented by the standard behavior of the associated Smalltalk objects, and cognitive, embodied in a NéOpus-based asynchronous production system [20]. This production system mainly comprises: (1) a rule base which includes objects describing the agent's environment and rules representing suitable operations over these objects; (2) an inference engine which includes specific coordination mechanisms (see below); and (3) a *metabase* which provides a declarative representation of the control of reasoning.
- the *communication/action* module allows the agent to receive and send messages asynchronously. It effects the direct actions (modification of the environment via effectors) and indirect actions (information transmission to other agents) as indicated by the reasoning module.

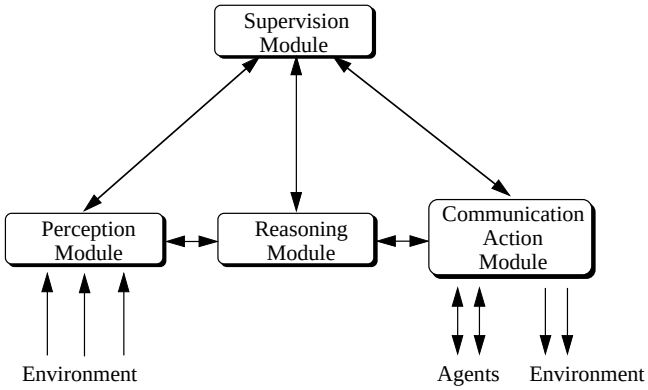

Fig. 5: The internal architecture of an agent.
Arrows indicate the flux of information.

The supervisor allows the agent to adapt in real-time its behavior to changes in its surrounding world. It synchronizes the execution of the concurrent actions of the other modules. It is based on *states* and *transitions*. States qualify the context as perceived by the other modules, and changes in the context are reflected as transitions between states. States and transitions build up an ATN (Augmented Transition Network). The various signals received by the agent's modules represent the conditions of transition and the actions of transition change the state of the various modules (activate reasoning, terminate reasoning, ...). When these conditions are satisfied, the transition actions are executed and the agent's state is modified.

All the agents in NéoGanesh have the same structure described above but they differ in several criteria: the presence of a sensor-driving layer, the nature of the know-how represented, and the domain and control knowledge encoded in the reasoning module. Agents are classified according to these criteria. The Fig. 6 shows the general hierarchy of agent types and indicates the types that correspond to the agents composing NéoGanesh. Note that, for instance, agent *DataProcessor* has only a simple behavior to acquire and process, whereas, agent *TemporalReasoner* exhibits a complex

behavior to appreciate the time-course of the patient's ventilation.

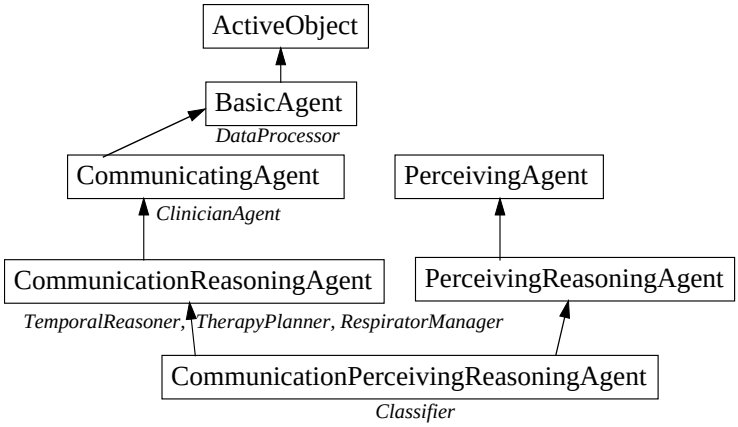

Fig. 6: Hierarchy of agents in NéoGanesh.
Each agent of NéoGanesh (named in italic) belongs to a specific agent type.

4.2. Control Mechanisms

We are interested in situations where an agent shares a collection of resources with other agents. Thus, agents must adapt themselves to take advantage of resources as needed, but must coordinate their actions to avoid inconsistencies and deadlocks. Most recent architectures for patient monitoring [23, 43] are based on the Blackboard paradigm. The control of the reasoning in these systems is centralized: agents (so-called knowledge sources) communicate indirectly with each other via a global structure. We consider that the reasoning control strategies must be separated from the domain knowledge of each agent and must be managed by the agent itself. Thus in our model, each agent communicates directly with the other agents, and manages its own activity (*auto-control*) while its interactions with the other agents are handled via specific *coordination mechanisms*. We will now outline these two control levels.

4.2.1. Auto-Control

The auto-control is managed by the supervisor of each agent (*adaptative control*) and the reasoning module (*intelligent control*). The supervor uses the ATN described in

4.1 to specify the behavior of the agent depending on its internal states and on its surouding world. The reasoning module uses a reflexive architecture to specify declaratively the control of the reasoning. In this reflexive architecture, the execution of a rule-base is entirely specified by another rule base called metabase. The metabase provides a declarative specification of rule firing through metarules. This declarative representation of rule firing requires specific *control objects* (called *Evaluators*) which reify the state of the reasoning process at a given time [33].

The Fig. 7 illustrates the two mechanisms used for the *auto-control* of an agent.

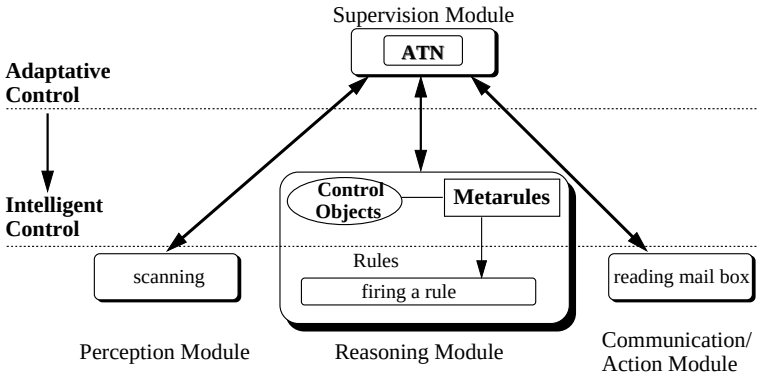

Fig. 7: Control mechanisms internal to the agent.
The figure shows the two levels of internal control. Arrows indicate the flux of information used for controlling the agent's activity.

4.2.2. Coordination Mechanisms

At the agents society level between agents, two coordination mechanisms are used:

- a *dependency mechanism* propagates modifications of information used by several agents,

- an *anti-inference mechanism* prevents simultaneous modifications of sharable information. Interference exists among two rules if there is a shared object that both rules access and at least one of them modifies it. These mechanisms are detailed in

[21]. Briefly, for the dependency mechanism, each agent is provided with a dependency graph associating each shared object with the list of other agents which use it. It is updated gradually when rules are triggered and a message is sent to the other agents to update their dependency graphs. For the anti-inference mechanism, we add a test in the inference engine of each agent to verify, before triggering a rule, that no shared object that is modified by the selected rule is also being modified by some other agent. In this case, the selected rule is triggered, otherwise another fireable rule is selected. The modification of shared objects is apparent from the 'objects-in-use' collection owned by each agent and updated before each rule triggering.

4.3. Concurrency

The agent model accommodates several agents and/or modules running on a single processor. To simulate parallelism, we have chosen a process allocation strategy at two levels: 1) supervisor level: the agent suspends its activity at the end of each transition by the ATN interpreter, and 2) perception, reasoning and communication modules level: process control is performed after each rule firing, after each mail box reading and at the end of each period of sensor processing.

5. KNOWLEDGE REPRESENTATION IN NÉOGANESH

Interpreting data over time is an essential task of diagnostic and control processes. The time course of a process, determined from the evolution of a set of representative parameters, is central to predicting its future behavior and to choosing actions over time to influence it. Since the early work in AI, many formal studies have been conducted about change and time representation [10, 46]. However, the

complexity of temporal constraint propagation algorithms have limited the integration of a time manager into real-time systems such as intelligent monitoring systems. Moreover, the expressive power of formal approaches is generally insufficient to cope for the complexity and variety of real situations. In the medical domain, specific methods may be used for the interpretation of time-ordered physiological data, detection of trends or definition of various temporal data abstractions (see [1]).

5.1. Temporal Reasoning

Recently, we have proposed a temporal reasoning model [17], on which NéoGanesh is based, in order to introduce an explicit time and change representation while ensuring computational tractability. Our ontology divides the modeled world between 'existing' domain *atemporal entities* that, by essence, have no temporal dimension (static description of the world), and *temporal entities* that are time stamped and are used to develop a temporal discourse about the changing world. We consider the recognition of change, which is context-dependent, as the central point in the perception of time. Similarly to Kowalski and Sergot's Event Calculus [25], we consider the notions of event, property, time-point and time-interval as primitives and define a model of change in which events happen at time-points and initiate and/or terminate time-intervals over which some property holds. We introduce a distinction between active proposition (event-type) to represent the occurrence of actions which modify the state of the world (such as “increasing the assistance”), and passive properties (state-type) (such as “patient-is-weanable”). Thus, we have introduced two basic domain-independent entities: `state` and `event`. `State` and `Event` refine the general concept of `TemporalObject` [17] which associates a property with a temporal lapse. They are

similar to the notion of *time-objects* introduced in [24] with a set of temporal, causal and structural relations. A state `State(Value, Ival)` represents a property whose value (`Value`) holds during the validity interval `Ival` associated to it. For instance, an assertion such as `RespiratoryState(Normal, [t1, t2])` means that the patient's ventilation is considered as normal during the period included between the instants `t1` and `t2`. The occurrence of an event initiates or terminates a state. According to the weak interpretation of Event Calculus, we consider that an event initiates a property if this property does not already hold [8]. The assumption of default persistency [25] indicates implicitly that all properties are downward-hereditary [41]. In medical applications, this is not necessarily true and we introduce the notion of *non-convex intervals* [26] where a property is true but not at each point of the interval.

To reason about temporal objects, we introduce two abstraction mechanisms: *aggregation* of similar situations and *forgetting* of non relevant, redundant, or out-of-date information. These abstractions allow the incremental interpretation of observations as they are acquired, and the determination of the expected evolution of the state of the patient. Because perception of changes is context-dependent, these mechanisms are activated and adapted depending on the context. For instance in the context of weaning (i.e. the patient is ventilated with a low level of assistance), we tolerate short instabilities and interpret situations valid on nonconvex intervals separated by variable gaps.

5.2. Object-Oriented Rule-Based Programming

To represent the medical expertise, including the temporal reasoning aspect, we use a hybrid knowledge representation scheme mixing object-oriented programming languages and production rules. By refinement of domain-independent entities and

inferences, we added domain-specific knowledge to manage the clinical therapy. Thanks to this combination of objects and rules, specific mechanisms allow us to represent various dimensions of context-dependency, and facilitate what we have defined as *limited reuse* within a given domain (see details in [14]).

Each agent's reasoning module owns a first order forward chaining rule base associated with a metabase. Classically, classes are used to represent domain concepts and relationships among concepts. For instance, classes `Event` and `State` represented the basic objects of our temporal ontology, and are subclassed to introduce more specific temporal objects (`VentDiagnosis` or `RespiratoryState` for instance). The Fig. 8 shows part of the hierarchy of temporal objects, their links and their relations with atemporal objects.

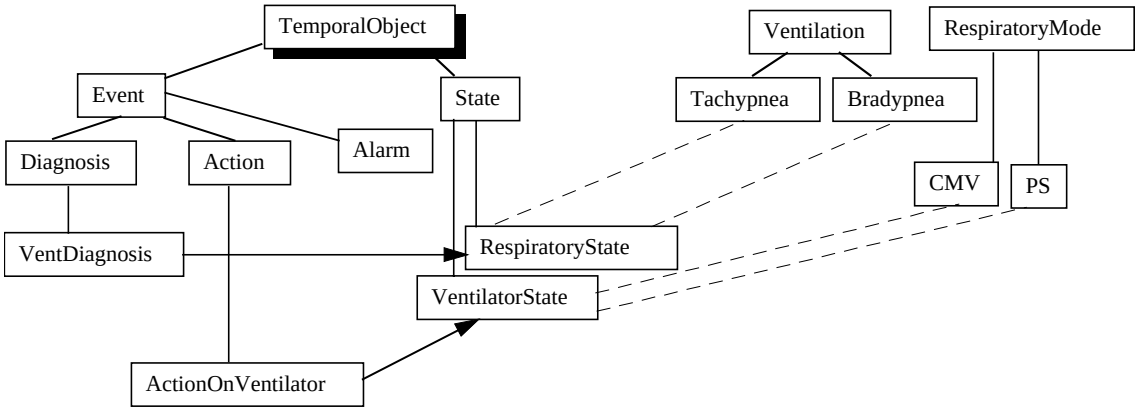

Fig. 8: Two hierarchies of temporal and atemporal objects.
Lines and dashed lines indicate respectively a kind-of relation and a link between temporal and atemporal entities. Arrows indicate the relation (initiates or terminates) between Events and States.

Production rules are used to capture domain knowledge and control knowledge. For instance a rule for temporal aggregation may look like the following:

```
Rule aggregation
"The rule indicates that we omit a short instability s2 that
is bounded by two similar states s1 and s3"

For any s1, s2, s3 instances of RespiratoryState
IF
 s3 is persistent.
 s1 is similar to s3.
 s2 is not similar to s3.
 s2 is between s1 and s3.
 duration in expertise of s2 <= 1.
 duration in expertise of s3 > 1.
then
   forget s2.
   aggregate s1 and s3
```

The natural typing mechanism [32] allows the pattern-matcher to consider direct instances of a class as well as instances of subclasses to be matched by rule variables for a given rule. Using this mechanism, condition and action parts of rules are dependent on the context represented by the set of objects that match the rules. In our representation framework (NéOpus), we transposed the class inheritance mechanism to rule bases. Using this mechanism, called Rule Base Inheritance, a rule base may be defined as a sub base of an existing rule base, thereby inheriting all its rules. Similarly to class inheritance, a rule base may only add new rules, or redefine an inherited rule into a more specific rule. Class inheritance and rule base inheritance allow to gradually introduce context information in rules, by specifying at each level of the inheritance tree only the differences between the rule base and the inherited ruleBase.

The declarative control of the reasoning via the use of metarules facilitates the specification of the sequencing of tasks and also allow to change the strategy during the reasoning [15]. For instance the metarule `startWeaning` triggers the `WeaningPlan` when specific conditions are met. In this metarule, *c* and *e* represent rule variables, and are declared, respectively, as instances of `ClinicConflictSet` (a specific conflict set) and `Evaluator` (representing the control objects).

```
Metarule startWeaning
"the metarule indicates the conditions to trigger the weaning plan"
For any e instance of Evaluator and c instance of ClinicConflictSet
IF
 the status of e is ‘loop’.
 c has a rule in protocol ‘startWeaning’.
THEN
 c fire rules in protocol ‘startWeaning’.
 execute weaning plan.
```

5.3. The knowledge base in its current state

The system includes a representation of every concrete object involved in the ventilation process. We defined 81 classes and about 350 methods to describe the medical problem.

The knowledge of the intensivist is represented by NéOpus rule bases. The system contains eleven rule bases associated to four classes of agents (see Fig. 6). Seven rule bases, containing a total of twenty-one rules, are dedicated to the diagnosis of current ventilation and to the definition of therapy. For these rules, conditions parts typically test the values of the physiological data, information about the patient, or the time-course of the patient's ventilation. Action parts typically build objects representing the diagnosis of the patient or representation of actions on the ventilator. Rule bases dealing with action plans are organized in a rube base inheritance hierarchy. In this hierarchy, five rules are inherited and seven are redefined in sub-bases.

Four rule bases are dedicated to the representation of temporal reasoning. These rule bases are also organized in an inheritance hierarchy. They contain a total of twenty rules (four are inherited in sub-bases and six are redefined in sub-bases).

As outline above, control strategies were introduced explicitly using the declarative representation of control and rule base inheritance for metabases. A total of nine metabases were defined. The control of the diagnosis reasoning and of the definition of therapy, is represented by eighteen metarules (five are inherited and nine are redefined in sub-bases). The control of the temporal reasoning requires nineteen metarules (five are inherited and eleven are redefined in sub-bases).

6. EVALUATION

In NéoGanesh, both synchronous communication and asynchronous communication between agents are possible, although the latter has not been tested yet in a clinical environment. Preliminary results [21] showed that the response time (the elapsed time between the perception of an event and an action on the ventilator) and the reaction time (the elapsed time between the perception of an event by `DataProcessor` agent and its processing by the `Classifier` agent) are shorter with the introduction of asynchronous agents than with synchronous communication. This is reinforced in distributing the system on two machines.

Automated control systems for mechanical ventilation have the advantage of providing 24-hour a day management, potentially allowing continuous adaptation of the level of assistance and a reduction of the duration of mechanical ventilation. It is possible to evaluate several aspects of NéoGanesh, reflecting benefits for the patient, user or health care institution. Our evaluation has been a long iterative process. It focused on the quality of the assistance provided by NéoGanesh to the patient and on the reliability of the decision about the withdrawal of respiratory assistance. The evaluation of NéoGanesh with synchronous communications between agents has been performed at the Henri Mondor Hospital in two steps.

The aim of the first step was to evaluate the capability of NéoGanesh to maintain the patient in a zone of respiratory comfort defined as: 12 < Respiratory Rate < 28 cycles / min (or 32 in case of neurologic disorders), tidal volume > 300 ml (or 250 ml for patients with weight <= 55 Kg) and end-tidal CO_2 pressure < 55 mmHg (or 65 in case of COPD). Patients (n=19) were divided in two groups according to their results to a

battery of tests: Group 1 (n=10) which gathered good candidates for the weaning and Group 2 (n=9) which gathered more severe patients considered as bad candidates for the weaning. Patients in Group 1 were kept out of critical zones (respiratory rate <12 breaths / min, or respiratory rate > 35 breaths / min or tidal volume < 300 ml) for 99% of the duration of total ventilation and patients in Group 2 for 90% of this duration. We have presented the details of this evaluation in [12].

In a second study, 5 patients were ventilated randomly 24 hours with and without NéoGanesh. Results show (see Fig. 9) that NéoGanesh maintained the patients within a comfortable zone of ventilation during 91±8% of the total duration of the ventilation, compared to 71±18% without it. Patients spent 4±7% of the total duration in severe situations compared to 18±15% without the use of the system.

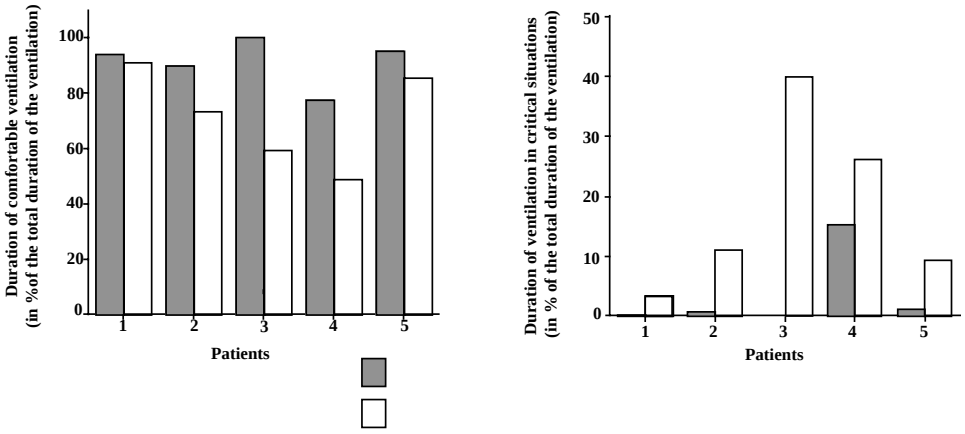

Fig. 9: Clinical Results

Globally, patients were ventilated with a similar level of pressure support (18±4 cmH_2O and 17±6 cmH_2O with NéoGanesh and without respectively). Thus, the system reached the goal fixed by the clinician.

When the patient is ventilated with a low level of assistance (9 cmH_2O or 5 cmH_2O if the patient is tracheotomized), NéoGanesh may recommend to the clinician to wean the patient. The aim of the second step of the clinical evaluation was to test whether NéoGanesh could correctly predict the ability of patients to tolerate total withdrawal from respiratory support. In 38 patients, the suggestions of NéoGanesh were compared to a conventional procedure (including tolerance of T-piece). The negative predictive value was identical for the two procedures and equal to 100%. However, the positive predictive value of NéoGanesh was 89% versus 77% for the conventional procedure. Details concerning this study have been published in [13].

Thus, we conclude that NéoGanesh ensures appropriate patient management during the weaning period and improves our ability to predict responses to weaning.

7. CONCLUSION

NéoGanesh is based on current AI technologies: sophisticated knowledge representation and temporal reasoning in a distributed architecture. We have chosen an environment which combines actors, objects, and production rules. We fully exploit the well known mechanisms of object-oriented programming and by using the inheritance mechanism our system can be easily extended.

Preliminary clinical studies performed at Henri Mondor Hospital have demonstrated that patients show less signs of respiratory discomfort with an automatic control of the ventilation (the NéoGanesh system) than without it. Moreover, the diagnosis proposed by this system, concerning the capability of the patient to breathe without external assistance, is more efficient than with the usual manual procedure.

The extensive use of the system will contribute to its improvement (knowledge encoded, enhancement of the data interpretation methods) and will fully validate this new technique for automatic supervision of assisted ventilation. It will directly contribute to the definition and test of guidelines for the management of assisted ventilation.

Because of the extreme difficulty in defining and in validating appropriate physiologic models for patient's ventilation, NéoGanesh relies on the representation of expertise acquired over the 10 past years about breathing patterns during acute respiratory failure and about pressure support ventilation. It is useful in a restricted medical field: for example heart failure, oxygenation and neurologic problems, which may influence weaning, are not taken into account. The limitations of NéoGanesh are also its strength. Compared to other ventilator-management managers such as VentPlan [37] which incorporate mathematical models, it may be implemented in ICU. It works in closed-loop at the patient's bedside in contrast to Weanpro [45] or VentEx [39]. NéoGanesh constitutes a clinically validated module, the first part of a more complex patient monitoring system.

We are pursuing our research in several directions. We are exploring the integration of new measurements to better characterize the patient's state [6]. A new version of the current system used in our Hospital, relying on an asynchronous execution of the agents (implemented with a concurrent programming language), has to be extensively tested to measure its performance. Distribution of knowledge raises several specific problems such as synchronization of agents, resolution of conflicts between agents viewpoints and coordination of different temporal reasoning mechanisms.

Planning capabilities will also be integrated to our temporal model, to improve the quality of the system's predictions and its capability to dynamically adapt the therapeutic strategy. Temporal scenario recognition appears to be an interesting method to reason about time in dynamic process supervision when no mathematical model of the process is known to determine its behavior [34]. Planning typically requires *constraint satisfaction* techniques, in our case applied to complex object structures. Recent work on the integration of constraint satisfaction with object-oriented languages [36] are directly applicable for this purpose.

ACKNOWLEDGMENTS: This work has been partially financially supported by Hamilton AG company. We wish to thank Professor Jean-François Perrot for his stimulating advice and constant encouragement that significantly influenced this work.

8. REFERENCES

[1] Artificial Intelligence in Medicine journal Special Issue on Temporal Reasoning in Medicine, 8 (1996)

[2] J. P. Briot, Actalk: a testbed for classifying and designing actor languages in the Smalltalk-80 environment, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'89*) (1989), 109-130.

[3] L. Brochard, A. Harf, H. Lorino and F. Lemaire, Inspiratory pressure support prevents diaphragmatic fatigue during weaning from mechanical ventilation, *Am. Rev. Respir. Dis.* 139 (1989) 513-521.

[4] L. Brochard, F. Pluskwa and F. Lemaire, Improved efficacy of spontaneous breathing with inspiratory pressure support, *Am. Rev. Respir. Dis.* 136 (1987) 411-415.

[5] L. Brochard, A. Rauss, S. Benito, G. Conti, J. Mancebo, N. Rekik, A. Gasparetto and F. Lemaire, Comparison of three methods of gradual withdrawal from ventilatory support during weaning from mechanical ventilation, *Am. J. Respir. Crit. Care Med.* 150 (1994) 896-903.

[6] J. Carrive, B. Louis, A. Harf and M. Dojat, BioPhony: An open system to measure the airway area by acoustic reflexion, *Proceedings of the eighteenth conference IEEE-EMBS* (1996), 64-65.

[7] M.-C. Chambrin, C. Chopin and K. H. Mangalaboyi, Autoregulated inspiratory support system, *Proceedings of the fourteenth conference IEEE-EMBS* (1992), 2419-2420.

[8] L. Chittaro and M. Dojat, Using a general theory of time and change in patient monitoring: experiment and evaluation, *Comput. Biol. Med.* (1996) In Press.

[9] T. L. Dean and M. P. Wellman, *Planning and control* (Morgan Kaufmann, San Mateo (Ca), 1991).

[10] R. Dechter, I. Meiri and P. Judea, Temporal constraints networks, *Artif. Intell.*, 4 (1991) 61-95.

[11] T. Deutsch, E. Carson and E. Ludwig, *Dealing with medical knowledge. Computers in clinical decision making* (Plenum Press, New York, 1994).

[12] M. Dojat, L. Brochard, F. Lemaire and A. Harf, A knowledge-based system for assisted ventilation of patients in intensive care units, *Int. J. Clin. Monit. Comput.* 9 (1992) 239-250.

[13] M. Dojat, A. Harf, D. Touchard, M. Laforest, F. Lemaire and L. Brochard, Evaluation of a knowledge-based system providing ventilatory management and decision for extubation, *Am. J. Respir. Crit. Care Med.*, 153 (1996) 997-1004.

[14] M. Dojat and F. Pachet, Effective domain-dependent reuse in medical knowledge bases, *Comput. Biomed. Res.* 28 (1995) 403-432.

[15] M. Dojat and F. Pachet, An extendable knowledge-based system for the control of mechanical ventilation, *Proceedings of the fourteenth conference IEEE-EMBS* (1992), 920-921.

[16] M. Dojat and F. Pachet, Representation of a medical expertise using the Smalltalk environment: putting a prototype to work, in: G. Heeg, B. Magnusson and B. Meyer, eds., *TOOLS 7* (Prentice Hall, New York, 1992) 379-389.

[17] M. Dojat and C. Sayettat, A realistic model for temporal reasoning in real-time patient monitoring, *Appl. Artif. Intell.* 10 (1996) 121-143.

[18] P. Ershowsky and B. Krieger, Changes in breathing pattern during pressure support ventilation, *Respir. Care* 32 (1987) 1011-1016.

[19] J. F. Fiastro, M. P. Habib and S. F. Quan, Pressure support compensation for inspiratory work due to endotracheal tubes and demand continuous positive airway pressure, *Chest* 93 (1988) 499-505.

[20] Z. Guessoum, Un environnement opérationnel de conception et de réalisation de systèmes multi-agents, PhD Thesis in Computer Science, University Paris 6, 1996.

[21] Z. Guessoum and M. Dojat, A real-time agent model in an asynchronous object environment, in: W. Van de Velde and J. W. Perram, eds., *Agents Breaking Away* (Springer, Berlin, 1996) 190-203.

[22] B. Hayes-Roth, An architecture for adaptive intelligent systems, *Artif. Intell.* 72 (1995) 329-365.

[23] B. Hayes-Roth, R. Washington, D. Ash, R. Hewett, A. Collinot, A. Vina and A. Seiver, Guardian: a prototype intelligent agent for intensive care monitoring, *Artif. Intell. in Med.* 4 (1992) 165-185.

[24] E. T. Keravnou, Temporal diagnosis reasoning based on time-objects, *Artif. Intell. in Med.* 8 (1996) 235-265.

[25] R. A. Kowalski and M. J. Sergot, A logic-based calculus of events, *New generation computing*, 4 (1986) 67-95.

[26] P. Ladkin, Time representation: A taxonomy of interval relations, *Proceedings of the sixth National Conference on Artificial Intelligence* (1986), Philadelphia, PA, 360-366.

[27] G. Lanzola, S. Falasconi and M. Stefanelli, Cooperative software agents for patient management, *Proceedings of the fifth conference on Artificial Intelligence in Medicine Europe, AIME'95* (1995), Pavia (It), 25-28 June, 173-184.

[28] T. P. Laubscher, W. Heinrichs, N. Weiler, G. Hartmann and J. X. Brunner, An adaptive lung ventilation controller, *IEEE Trans. Bio. Eng.* 41 (1994) 51-58.

[29] D. M. Linton, P. D. Potgieter, S. Davis, A. T. J. Fourie, J. X. Brunner and T. P. Laubscher, Automatic weaning from mechanical ventilation using an adaptative lung ventilation controller, *Chest* 106 (1994) 1843-1850.

[30] N. R. MacIntyre, Respiratory function during pressure support ventilation, *Chest* 89 (1989) 677-683.

[31] D. J. Musliner, J. A. Hendler, A. K. Agrawala, E. H. Durfee, J. K. Strosnider and C. J. Paul, The Challenge of Real-Time AI, *Computer* (1995) 58-66.

[32] F. Pachet, On the embeddability of production rules in object-oriented languages, *JOOP* 8 (1995) 19-24.

[33] F. Pachet and J. F. Perrot, Rule firing with metarules, *Proceedings of the sixth conference on Software Engineering and Knowledge Engineering* (SEKE) (1994), Jurmalia (Letonie), 322-329.

[34] N. Ramaux and M. Dojat, Temporal scenario recognition for intelligent patient monitoring, *Proceedings of the sixth conference on Artificial Intelligence in Medicine* (1997), Grenoble (Fr), AIME'97. In press.

[35] M. Ramoni, M. Stefanelli, L. Magnani and G. Barosi, An epistemological framework for medical knowledge-based systems, *IEEE Trans Systems, Man, Cybernetics* 22 (1992) 1361-1375.

[36] P. Roy and F. Pachet, Reifying constraint satisfaction in Smalltalk, *JOOP* (1997). In press.

[37] G. W. Rutledge, G. E. Thomsen, B. R. Farr, M. A. Tovar, J. X. Polaschek, I. A. Beinlich, L. B. Sheiner and L. M. Fagan, The design and implementation of a ventilator-management advisor, *Artif. Intell. in Med.* 5 (1993) 67-82.

[38] B. Séroussi, V. Morice, F. Dreyfus and J. F. Boisvieux, Control theory as a conceptual framework for intensive care monitoring, *Artif. Intell. in Med.* 7 (1995) 155-177.

[39] N. Shahsavar, U. Ludwigs, H. Blomqvist, H. Gill, O. Wigertz and G. Matell, Evaluation of a knowledge-based decision support system for ventilator therapy management, *Artif. Intell. in Med.* 7 (1995) 37-52.

[40] Y. Shoham, Agent-oriented programming, *Artif. Intell.* 60 (1993) 139-159.

[41] Y. Shoham, Temporal logic in AI: semantical and onthological considerations, *Artif. Intell.* 33 (1987) 89-104.

[42] J. H. Strickland and J. H. Hasson, A computer-controlled ventilator weaning system, *Chest* 103 (1993) 1220-1226.

[43] T. I. Sukuvaara, M. E. Sydänmaa, H. O. Nieminen, A. Heikelä and E. M. J. Koski, Object-oriented implementation of an architecture for patient monitoring, *IEEE Eng. Med. Biol.* 12 (1993) 69-81.

[44] M. J. Tobin, W. Perez, S. H. Guenther, B. J. Semmens, M. J. Mador, S. J. Allen, R. F. Lodato and D. Dantzker, The pattern of breathing during successful and unsuccessful trials of weaning from mechanical ventilation, *Am. Rev. Respir. Dis.* 134 (1986) 111-118.

[45] D. A. Tong, Weaning patients from mechanical ventilation. A knowledge-based system approach, *Comput. Meth. Prog. Biomed.* 35 (1991) 267-278.

[46] P. Van Beek, Reasoning about qualitative temporal information, *Artif. Intell.* 58 (1992) 297-326.

[47] M. Younes, Proportional Assist Ventilation, a new approach to ventilatory support, *Am. Res. Respir. Dis.* 145 (1992) 114-120.