# A Distributed, Agent-Based Architecture for the Acquisition, Management, Archiving and Display of Real-Time Monitoring Data in the Intensive Care Unit

Dimitrios G. Katehakis[1], George Chalkiadakis[1, 2], Manolis Tsiknakis[1], Stelios C. Orphanoudakis[1, 2]

[1] Center of Medical Informatics and Health Telematics Applications (CMI-HTA), Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH), P.O. Box 1385, GR 711 10, Heraklion, Crete, Greece, Tel: +30 (81) 391600, fax: +30 (81) 391601, E-mail: {katehaki, gehalk, tsiknaki, orphanou}@ics.forth.gr, URL: http://www.ics.forth.gr.

[2] Department of Computer Science, University of Crete, P.O. Box 2208, GR 714 09, Heraklion, Crete, Greece.

## Abstract

The intensive care is a unit of vital importance within a hospital, where although a small number of patients are being treated, a great number of their vital functions have to be monitored very carefully. Monitoring within an Intensive Care Unit (ICU) is usually based on heterogeneous, dedicated hardware and software devices that embody their own time metrics for the collection of vital sign measurements provided at regular time intervals. This paper presents a distributed, agent-based architecture for the acquisition, management, archiving and display of real-time monitoring data in the ICU that makes efficient use of the Common Object Request Broker Architecture (CORBA). A prototype implementation is presented for the acquisition and communication of the continuously fed, vital sign information from the appropriate and distributed, real-time monitoring devices to single graphical user interfaces at the corresponding ICU monitoring workstations. Focus is paid on the design and development of individual, collaborating software agents to be used for data acquisition and monitoring, and on their communication through stable interfaces of the middleware infrastructure. Additional issues examined include the use of efficient mechanisms for handling sudden increase in workload, environment changes and monitoring device malfunctions.

**Keywords:** intensive care, CORBA, software agents, real-time ICU monitoring, distributed architectures, distributed systems.

## 1. Introduction

In intensive care, there is a need for the use of assisting information systems for on-line monitoring and recording of vital sign information acquired for bedside monitoring. This need stems from the existing demand for the exploitation of the continuously increasing volume of data produced, which makes patient supervising extremely tedious for medical personnel [Bellon94] [Metniz95] [Tonnesen97]. Monitoring patients in an ICU demands appropriate reactions to time-critical circumstances and complicated situations, since the distinction between small and large parameter changes or the presence of specific monitoring signals and the decision of when and what actions are required is not an easy task to model and follow.

Artificial Intelligence (AI) has in the past emphasized on the creation of centralized "expert" systems, which tried to specialize in dealing with difficult problems, by making use of complicated knowledge structures. Most of the published efforts so far deal with the development of an ICU medical workstation to be used for ventilation management, mostly with adult patients. Medical workstations belonging to this domain mostly contain a very complex architecture to manage all the different parameters. For example VentPlan's [Rutledge93] goal is to assist medical personnel in the treatment of patients who need respiration support, provided by a ventilator for postoperative patients in a Surgical ICU (SICU). On the other hand, VIE-VENT [Miksch95] is a knowledge-based monitoring and therapy-planning system for artificially ventilated newborn infants to optimize therapy planning and to support neonatologists in their daily routine. Several other approaches exist: PATRICIA [Moret-Bonillo93] has been designed to advise clinicians in the management of patients dependent on mechanical ventilation and incorporates a patient-oriented symbolic approach for representing knowledge and a symbolic-oriented temporal approach for the intelligent control of the monitoring process. EPILOG is a UNIX-based system that controls routine operations such as analogue-to-digital conversion, data storage, and media management [Collura92] [Collura93]. GUARDIAN [Larsson97] [Larsson98], the most advance R&D effort so far, is applied to respiratory and cardiovascular monitoring problems in an SICU. GUARDIAN's reasoning system interprets perceived information from the environment, performs all knowledge-based reasoning and problem solving (e.g. problem detection, diagnosis, prediction, planning, explanation), and decides what actions to perform. It also constructs and modifies dynamic global control plan to co-ordinate its perception, reasoning, and action. However, none of the above mentioned approaches emphasizes on the distributed nature of the architectures, and none of them discusses issues closely-related to the middleware infrastructure required in an ICU in order to handle the requirements imposed by the real-time nature of the continuously-fed information (i.e. availability, scalability, extensibility, and resource sharing).

This paper's objective is to elaborate on the architectural framework that has already been presented in [Katehakis97], regarding the ICU information system. As such it mainly focuses on the perception of the dynamic environment found within the ICU, and provides stable CORBA interfaces to achieve on-line acquisition and communication of continuously fed, multi-sensorial vital signs from the appropriate and distributed, real-time monitoring devices to a uniform graphical user interface. An important issue examined includes the use of efficient mechanisms for handling sudden increases in workload, environment changes and monitoring device malfunctions. Issues pertaining to cognition are left, for the moment, open for future work. Section 2 describes the real-world, daily situation that exists in the intensive care, as well as its requirements. Section 3 presents the architectural considerations for the design of any system that is to be used for collecting, validating, recording, recalling, processing and communicating multimedia information related to patients under medical observation and recovery in the intensive care. Section 4 focuses on the design characteristics of the appropriate software components used for data acquisition and monitoring, as well as their communication requirements. The implementation status of the developed software components that can acquire, manipulate, store and display continuously fed vital signs through uniform graphical user interfaces at the corresponding ICU monitoring workstations, are presented in Section 5. Section 6 exhibits the implemented system's behavior as well as the lessons learned, while Section 7 provides the reader with some conclusions and considerations regarding future work.

## 2. Requirements of the Intensive Care

The ICU is a hospital ward with a limited number of patients and a small group of users. Under *normal conditions* the ICU monitoring devices function properly, the patient's state is steady and the medical personnel's responsibility is to watch patients and treat them accordingly. When one or more monitoring device alarms are active, then there is an *emergency situation*. In such a situation, it is very important for the patients' record to be updated with information regarding the injected drugs, the medical acts, as well as the acceptance or rejection of certain data recordings during the alarm period. Another critical point for the proper health care delivery in an ICU is the seamless *shift change*. The major issue here is the critical patient information, associated to special treatment needs, that has to be passed along. What is important here is not what has to be done, rather than what has preceded. Much of detail in this data is not relevant to the patient record after the patient has passed a given point in time.

Vital sign parameters like Arterial Blood Pressure (ABP), Hearth Rate, Temperature, Respiratory Rate (RR), Fraction of Inspired Oxygen ($FiO_2$), Tidal Volume (VT) and Positive End-Expiratory Pressure (PEEP), are of the most common ones that an ICU practitioner deals with daily. A typical set of ICU monitoring devices (including physiological monitors, respirators, thermidometers, infusion pumps, ventilators, and capnographs) provides a wide variety of digital parameters (like e.g. cardiac rate, respiration rate, pulse oximetry, $FiO_2$, $VCO_2$, $VO_2$, RQ) with varying sampling rate requirements ranging from periods of milliseconds to seconds or even more. Consequently, there is a clear distinction between *simple measurements* (having a sampling period at the order of seconds or more) and *continuous recordings* (having a sampling period at the order of milliseconds – 500 Hz is the typical sampling rate for ECGs).

With all the above in mind, the requirements that any ICU-oriented information system should fulfill can be summarized as follows:

- **On-line data acquisition from various sources:** This, combined with the ability to extract additional information from primary sensed data, can lead to new data presentation methods and can provide the practitioners with new possibilities for efficient treatment.
- **Automatic data validation:** This way human errors can be minimized and equipment malfunctions can be isolated.
- **Integrated data presentation centered around the different data modalities:** Controlling data in a uniform manner can reduce significantly the time spent by hospital personnel switching to and from various data modalities, and thus improve the ICU productivity.
- **Fast data manipulation:** The amount of real-time data should be condensed to a more accurate representation of the patient's condition, without causing bottlenecks. The computer system must not distract the busy ICU clinician, while the data entry process and the data review process must be easy to understand and perform.
- **Reporting and charting:** By presenting information in an easily accessible, meaningful, and readable way, documentation quality can be improved significantly.
- **Adherence to international standards:** This is the only viable solution to the problem of integration.
- **Scalability:** It should be easy for the architecture to be expanded with the addition of more interoperable software components. It should be also easily incorporated within a larger scale on-line system.

## 3. Architectural Considerations

Most of the on-line monitoring devices found in the ICU today are proprietary, vendor dependent systems that have little or no access to other information sources such as hospital information systems. They are usually based on dedicated hardware and software that embody their own time metrics to collect data provided at regular time intervals. They process data acquired from *sensors*, and consequently send the corresponding results to proper *activators* or *displays*. Continuously fed data that need to be monitored uniformly so as to detect any changes promptly from a central monitoring workstation have to be either read periodically or passed together with their corresponding time stamp by the monitoring devices themselves.

Any architectural attempt to enable this data collection from the various monitoring devices, as well as the compression of vital sign information for the effective storage of huge volumes of data produced, should be modular and extensible. In order to achieve dynamic reallocation of devices, data acquisition should be performed by separate and reusable, software components connected to the appropriate ICU monitoring devices, which should be able to operate independently [Hayes-Roth96]. Each acquisition component should possess access to individual monitoring device profiles, in order to be able to extract the vital sign measurement packets provided by attached device(s) (e.g. via the use of multiple serial communication ports) with minimum reconfiguration effort each time acquisition is initiated. Therefore, knowledge of information regarding the appropriate communication parameters, monitoring device name, its individual characteristics, as well as the value range of all the relevant measurements is essential. This would allow for on-line reconfiguration [Stewart97]. In addition, storage management techniques should be efficient enough to allow for the tracking of any potential inter-working anomalies. The environment should be managed throughout its execution because of the varying workload environment changes and failures the system must continuously react to [Marzullo91]. Special care should be taken for enabling automatic relation and comparative evaluation of certain diagnostic parameters, and their changes, as well as the openness for future enhancements, enabling the use of smart alarms and controls. The user-interface, providing end-user access to visual information should be uniform and customizable [Thull93] so that it may substantially contribute to the extraction of useful conclusions. Functional integration of the ICU monitoring system with other information systems, through the development of a multimedia clinical workstation, should be supported too.

Any such non-centralized architecture composed of multiple autonomous processing entities, co-operating towards a common purpose, in a dynamic and distributed environment demands the use of distributed computing standards, specifying inter-object communication. In addition, for each individual entity to be able to achieve the predetermined goals, it needs to exhibit re-activeness to external stimuli, and to possess both learning and co-operation capabilities.

### 3.1 Agent-Based Computing

It is a fact that the concept of "agent hood" has not been strictly defined so far. However, it is possible to define a set of main characteristics that could identify an entity as an agent. Software agents could in general be regarded as dynamic and adaptable, computational entities or robots that try to satisfy a set of objectives and are able to exhibit autonomous, goal-oriented behavior in complicated, unpredictable, dynamic, and possibly heterogeneous environments [Maes94] [Hedberg95] [Russel95]. Jennings and Wooldridge mention in [Jennings96] *autonomy*, *pro-activeness, social ability* and *responsiveness* as "key hallmarks of agent hood". Autonomy refers to the entity's ability to have control over its own actions and pro-activeness to its capability to take initiative to act in such a way that will enhance the fulfillment of its intended goals. Social ability refers to the exhibition of social behavior while it interacts with other agents (software entities or humans). Finally, responsiveness can be thought of as the ability of an agent to respond promptly and properly to the changes that occur to its environment[1].

The goals of an agent could vary from being extremely simple to being extremely complicated: they could be intermediate goals to serve a final goal, or internal needs and motives that have to be held within certain limits [Wooldridge95]. The agent has to use *reasoning* in order to achieve its goals (probably by altering the environmental conditions), and has to exhibit behaviors that could also vary from stereotype to complex [Hayes-Roth95] [Sycara96]. Clearly, agents can be thought to perform three tasks: *perception* of the dynamically changing environment conditions, *reasoning* so as to interpret the data being perceived and solve problems and *action* to influence/ alter these conditions. In analogy to the human sensing and effecting organs, an agent has *sensors* to perceive its environment and *actuators* (or effectors) to act on it (Figure 1). These tasks are performed while interaction with other dynamic (human or non-human) entities is taking place.

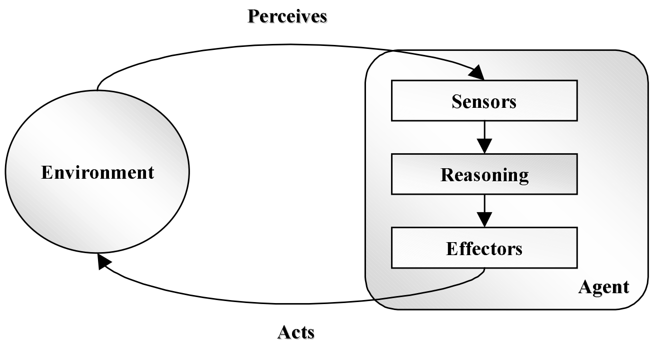


*Figure 1. Agents Interact with the External Environment through Sensors and Effectors.*

While Single Agent Systems (SAS) use a single agent that can communicate with a human user and local or remote resources (but not with other agents), Multi-Agent Systems (MAS) are occupied by many agents that are able to communicate with each other. According to Stone and Veloso [Stone97] the use of MAS possesses certain benefits:

- The overall system's speed is improved due to the parallel execution of a task's components by different agents.

---

[1] Except those main characteristics, secondary abilities the agents exhibit are: *sincerity*, which means that the agent will not consciously give false information; *good will,* which means that the agents does not have contradictory intentions; and *sense,* which means that the agent will only perform actions that seem to promote the achievement of its goals (or at least they don't seem to hazard those goals). In addition, another ability that often characterizes agents is *mobility*, that is the capability of agents to move from one system to another, in order to use remote resources or to co-operate with other agents.

- The overall system becomes more stable and secure due to workload sharing and no single point of system's failure exists.
- The overall system has the ability to extend and improve by just adding new agents in it.

The agents that occupy MAS may serve common system goals, or they may not have any common goals at all; in any case, they co-operate within the team to achieve their goals. When there are no common team goals, the team is called a *coalition of agents*, and it is common for members of the coalition to be *antagonistic* (i.e. they try to maximize their own profits from participating in the coalition). On the other hand, there are MAS architectures or architectures of agents that participate in MAS, which allow the temporary or once-and-for-all abandonment of secondary agent goals, in order to promote a common or a primary goal. Agents that participate in such a MAS, are usually called *co-operating agents* [Sandholm95]. Knowledge sharing between all the agent-members is decisive for the successful operation of a MAS, The existence of coherent understanding interaction protocols and the clear representation of global or private goals is necessary, in order to achieve successful knowledge sharing [Cohen91][Tambe96]. The existence of collisions-resolving protocols is also important, since collisions are possible to occur because of different private goals of the agents. The resolution of collisions is usually achieved through negotiations or even through auctions. Another important aspect is that agent communication is conducted either directly, or with the use of an intermediate or coordinator [Genesereth94].

Agents that participate in a multi-agent system share knowledge and abilities, and each of them is dedicated to achieve certain, relatively simple goals and to perform simple actions. What adds to the "intelligence" of the whole system is agent co-operation (or antagonism). Agents, not like traditional AI systems that are mostly "closed" applications without direct interaction with the environment, keep on interacting with the external environment directly and continuously by means of their sensors and effectors. An agent-based system, contrary to an AI system, may have to deal with more than one problem at the same time, possibly having to compromise different goals and action plans. In addition, cognitive structures of agent systems are adjustable and dynamically change as knowledge and experiences are acquired, in contrast to traditional expert systems.

### 3.2 Distributed Object Computing Middleware Infrastructure

Unlike client/ server architectures, where the client discovers and communicates directly with the server, when dealing with distributed object computing, communication middleware acts as an extra functional layer between clients and servers. This additional layer allows applications to be developed without knowledge of the location or any given logic implementation of all external functionalities. This way a client can transparently invoke a method on a server object by using an Object Request Broker (ORB) without needing to be aware of where the servicing object is located, the programming language implementation, the underlying Operating System (OS), or the any other system aspects that are not part of an object's interface (Figure 2).

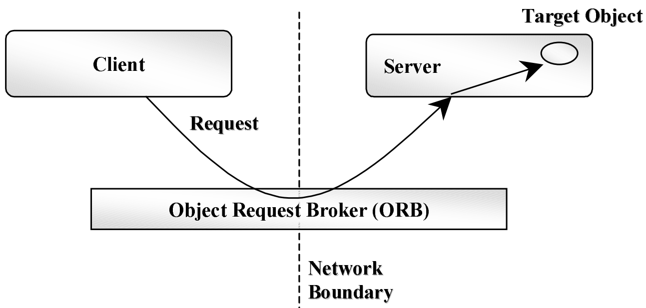


*Figure 2. A Client Invoking a Server Object Method.*

Three are the basic computing infrastructures that currently provide for transparent inter-process communication:

- The Object Management Group's (OMG) CORBA has been produced by a consortium of 760+ vendors as a platform-neutral infrastructure and as such, it can support object distribution across heterogeneous environments [Vinoski97] [OMG98]. CORBA allows applications to use a common interface, defined in an Interface Definition Language (IDL), across multiple platforms and development tools. As viewed by the client, a

CORBA object is entirely opaque, in that the object's implementation and location are unknown to the application that uses it.

- Sun's Java Remote Method Invocation (RMI) is based on the execution environment of Java to support network computers [Wollrath96] [Sun99]. Here, platform independence is accomplished by means of the Java Virtual Machine (JVM) that emulates a computing platform itself. The JVM is provided for each actual combination of hardware and OS upon which Java is to run.
- Microsoft's Distributed Component Object Model (DCOM) is an integration architecture that targets the homogeneous Windows environment (personal computer) and permits interaction between objects executing on separate hosts in a network [Box97] [Brown98]. As an extension rather than a separate architecture, DCOM inserts a stub interface between the calling application and the actual implementation of that interface, resembling an RPC-based model.

All three infrastructures automate common network programming tasks, such as object location and activation, parameter marshaling/ de-marshaling, socket and request de-multiplexing, fault recovery, and security by delivering requests to objects and by returning any responses to the clients making requests [Kuhns99] [Schmidt99]. Since clients see only the object's interface, and never any implementation detail, the existence of a distributed object computing middleware infrastructure guarantees substitutability of the implementation behind the interface. It also simplifies application development by providing a uniform view of heterogeneous networks, protocols, and OS layers. This facilitates the development of flexible distributed applications and reusable services in heterogeneous distributed environments. However, all three implementations are characterized by the lack of Quality of Service (QoS) specification and enforcement. CORBA provides no standard way for indicating the relative priorities of client requests, and there are no means for DCOM or RMI clients to inform an ORB how frequently to execute operations that must run periodically. In addition there are no guarantees that real-time applications will not to block indefinitely when ORB end system and network resources are temporarily unavailable, and in addition, they all incur both throughput and latency overhead [Pyarali96] [Gokhale98]. Nevertheless, there is a clear distinction, between what the needs of real-time systems are [Stankovic88], and what the architecture presented in this paper tries to achieve for the reasons described in Section 2.

One of the biggest factors in favor of Microsoft's DCOM solutions is its installed base. Most 32-bit versions of Windows support it, despite the fact that the programming model behind COM and DCOM is closely wedded to C++, making support for other programming languages such as COBOL and Java problematic. In a DCOM space the handler is usually an ActiveX component which although it can be packaged in several different forms, it is usually delivered as a Dynamic Link Library (DLL). As such, these components need to be installed, if only temporarily, into a user's system so that they can be executed. Using Java™ RMI is the easiest solution, since it only supports Java objects, and it fits directly into the Java model, with minimal impact on development resources for pure Java distributed systems, offering high portability. Although Java will undoubtedly have a large share of the market for new software development, Java™ RMI is impractical for use with objects or applications written in any other language, especially due to the huge number of existing legacy systems. CORBA provides the greatest flexibility of all three with its programming-language and platform neutrality, although there are some costs associated with this neutrality, both in deployment and in runtime overhead. Open systems and the sharing of resources increase portability, and cost effectiveness and all three architectures claim to allow applications to communicate with one another no matter where they are located or who has designed and implemented them. No matter what is the technology of choice, the adoption of a component approach in server applications, saves a lot of time in maintenance and increases the potential for future improvements. Baring in mind that there is no right or wrong choice, the following facts need to be recorded:

- CORBA has a very strong support from the industry and is not limited by either platform (like e.g. DCOM), or programming language (like e.g. Java™ RMI).
- The COM+ enhancement of DCOM (to appear in 2000) adds software layers to emulate CORBA's strengths [Kirtland97] [Box99].
- The Java Developers Kit (JDK) version 1.2 is now shipping bundled with a CORBA implementation providing support for the Java™ RMI over the Internet Inter-ORB Protocol (IIOP).

Therefore, and after taking into account all the above considerations, the Center of Medical Informatics and Health Telematics Applications (CMI-HTA), of the Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH), has designed and implemented a component-based architecture for the acquisition, management, archiving and display of real-time monitoring data in the ICU by means of CORBA. This architecture is described in the sections that follow.

## 4. System Architecture

The software entities that inhabit the presented architectural environment have a number of agent characteristics and exhibit, to a certain extent, agent-like behavior, and therefore, the term agent is used from this point on to refer to these entities.

Two such software entities, namely the *acquisition* agent and the *monitoring* agent, collaborate within the presented ICU monitoring architecture and their names are representative of their functionalities. In general, acquisition agents perform data acquisition and feed data to monitoring agents, who in turn facilitate data-visualization and storage (Figure 3). Since providing the agents that inhabit the proposed architectural environment with sophisticated cognition algorithms or knowledge bases is beyond the scope of this work, the reasoning tasks performed are quite simple, and no complicated symbolic representations have to be used. As a result, agent communication is being achieved without the use of an Agent Communication Language [Finin97]. However, extensions of the proposed architecture that would enhance both the agents' cognitive abilities and the entire system's intelligence are under consideration.

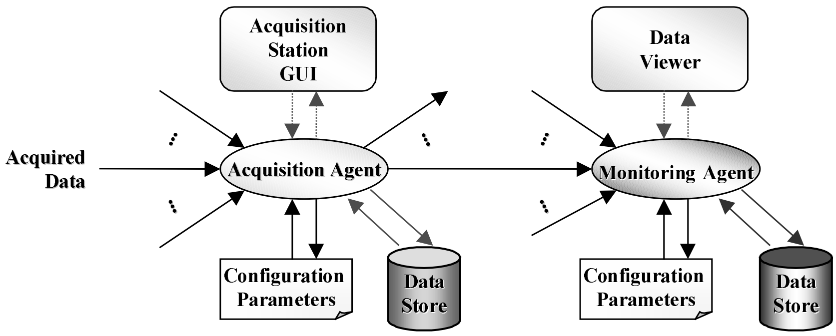


*Figure 3. Inputs and Outputs of the Acquisition and Monitoring Agents.*

The **acquisition agent** collects data from special hardware devices (i.e. the bedside monitoring devices) and/ or appropriate clinical information systems (e.g. the biochemical laboratory). Acquired data are temporarily kept at a local data store (i.e. RAM or disk), until they are transmitted to the appropriate monitoring agents. An acquisition agent may have a number of input and output channels, each of which can be dedicated to a different monitoring agent (Figure 4). The acquisition agent is therefore communicating with several monitoring agents simultaneously. Each outgoing channel possesses a set of individual characteristics that identifies the transmitted signal (i.e. destination, name, priority level, transmission rate, normalization factor, and units), and is serviced by the monitoring agent accordingly. The **monitoring agent** resides on a host and receives data, which are stored temporarily in a data repository and are visualized through a Graphical User Interface (GUI).

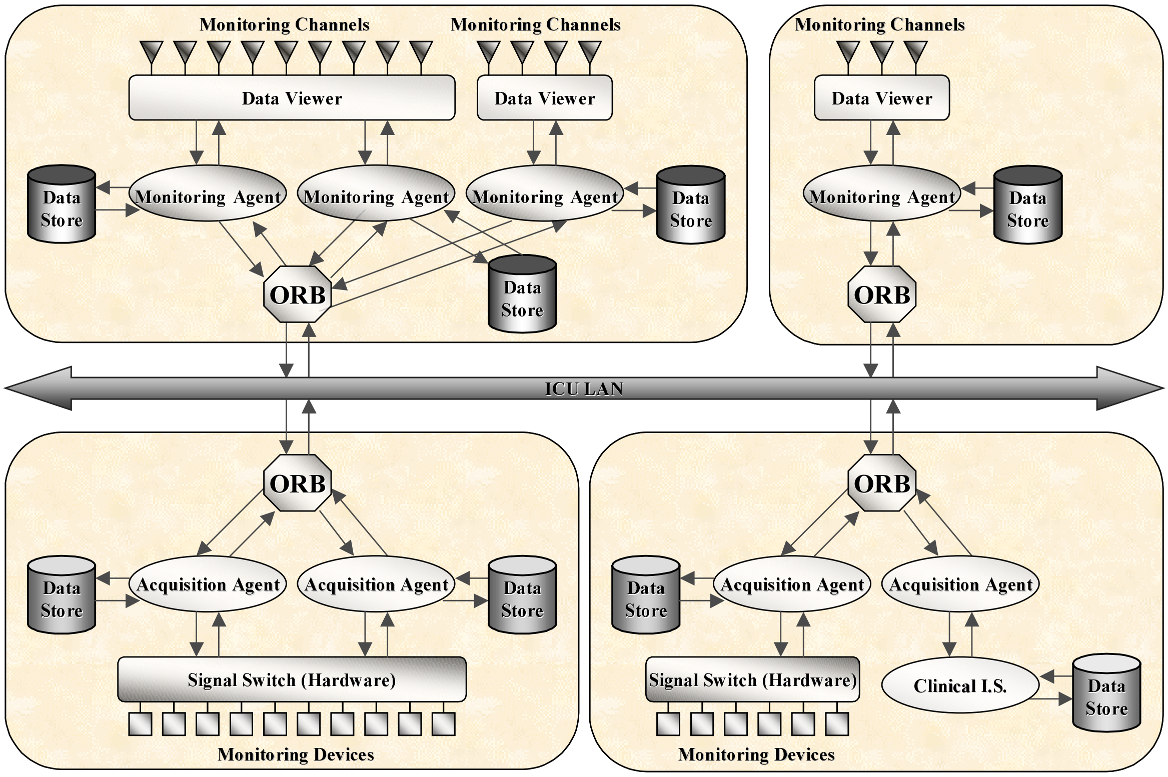


*Figure 4. The Distributed Agent-Based System Architecture.*

### 4.1 Dynamic Behavior

The monitoring agent, once initiated remains active, as it continuously checks for the existence of data to monitor. The agent-based architecture can handle the complexity of an ICU monitoring system by forcing the individual software components to collectively respond to situations where the timing requirements cannot be met. This is because sudden workload environment changes, monitoring device malfunctions, and/ or any other type of dynamic system changes may occur at any time.

Continuously fed data that need to be monitored from a central ICU monitoring workstation can either be read periodically or are passed together with their corresponding time stamp by the monitoring devices themselves. As soon as the acquisition agent initializes a certain input channel to prepare it for data collection, it reserves a certain amount of data storage space with the "*no signal*" indication. Each channel carries its own time stamp, which is defined either by the monitoring device itself, or by the acquisition agent's sampling period.

In the event of a monitoring device malfunction the acquisition agent informs immediately (responsiveness & pro-activeness) the corresponding monitoring agents by sending a special "*equipment malfunction"* signal. Every acquisition agent performs its main tasks (data acquisition and communication) autonomously, while a monitoring agent can intervene to the work of an acquisition agent, in case the former has decided (after reasoning) to enforce the latter to disconnect from its host. The acquisition agent could be responsive to this action, by immediately trying to connect to another monitoring agent in order to serve its disconnected channels.

In several cases, a human user can demand that a certain acquisition agent channel services a different monitoring device. This is done through the acquisition station's GUI, and is facilitated by means of the appropriate monitoring device drivers. The choice for device change is a choice of major importance, and results to several state transitions for the various open channels of the acquisition agent. All related data acquisition procedures have to be stopped and all related open acquisition agent channels (both incoming and outgoing) have to be closed and then restored to their original state. Proper care is taken for substituting all missing information, for all the re-initialization moments, with the indication "*unavailable*".

The record keeping of both continuous recordings and simple medical measurements is done in such a way that allows backward tracking up to a certain period. This guarantees error recovery, as far as data propagation from acquisition agents to monitoring agents is concerned, when network resources are temporarily unavailable. This means that when timing constraints are not met, then lower quality of on-line vital sign monitoring results.

The monitoring agent can autonomously reach decisions concerning the intelligent allocation of its resources (i.e. its outgoing monitored data channels). This can be done using reasoning/ cognitive procedures. Although the current monitoring agent implementation does not include any complicated cognition algorithms, when the monitoring agent runs out of resources a static algorithm is implemented that enforces the simple priority scheme of Figure 5. A monitoring agent resource can be a data-viewing channel, a smart alarm channel or a control channel.

*IF there is no monitoring agent resource available AND*
*there is a pending resource allocation request:*
*IF there is a monitoring agent resource allocated to an incoming "equipment malfunction" signal for over a time period of t*
*THEN release this monitoring agent resource*
*ELSIF there is a monitoring agent resource allocated to an incoming signal with priority level lower than the priority level of the pending resource allocation request*
*THEN release this monitoring agent resource*
*ELSE reject pending request*

*Note: The pending request is not guaranteed resource allocation since another pending request of higher priority may acquire the monitoring agent resource.*

*Figure 5. Simple Resource De-allocation Scheme for the Monitoring Agent.*

### 4.2 Communication Protocol

The CORBA technology is used to facilitate the communication between involved software components. CORBA IDL interfaces, where the functions that need to be exported to the network are defined, are built by each agent (either acquisition or monitoring). Any acquisition agent, required to service a certain outgoing data channel, gets a reference to the corresponding monitoring agent, by binding to the appropriate host. Subsequently, the acquisition agent then sends synchronization information (individual channel characteristics) to apply for an available monitoring agent resource (i.e. a monitoring channel) to serve its requesting outgoing data channel. If an available resource exists, the monitoring agent locks the resource and returns its *id* to the requesting acquisition agent. At the same time, the monitoring agent binds to the acquisition agent host to get prepared to invoke certain exception handling acquisition agent methods. If no monitoring agent resource is available, then the monitoring agent executes its resource release algorithm and the acquisition agent resubmits its application after a predetermined period of time. After a certain number of unsuccessful requests, the acquisition agent disconnects. Once a monitoring agent resource is allocated to the acquisition agent, the acquisition agent can start transmitting data to the monitoring agent through its connected data channel (Figure 6).

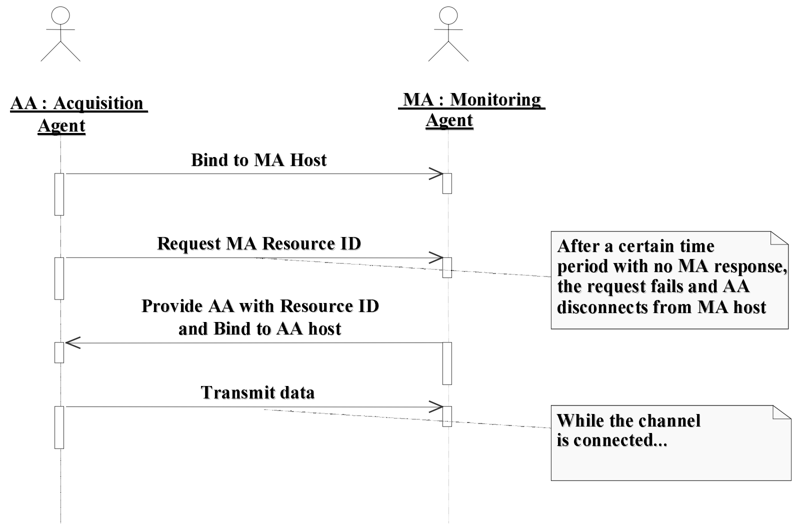


*Figure 6. Communication Establishment between an Acquisition Agent and a Monitoring Agent.*

A number of state transitions may occur, when a single acquisition agent outgoing channel is connected to the appropriate monitoring agent. The acquisition outgoing channel may be *idle*, *fully functional* (transmitting data properly), or *partly functional* (expecting certain actions to occur). The state transition diagram of Figure 7 shows that the partly functional state can be further decomposed into six more states (*AS*, *BtH*, *AS-BtH*, *BtH-S*, *AS-BtH-S*, *BtH-S-SNSD*). When the acquisition agent channel is in the *Initial State,* then it has not been bound to the monitoring agent's host, and hence it is non-functional (it is idle). The acquisition agent channel transmits data effectively only in the fully functional state (*AS-BtH-S-SD*).

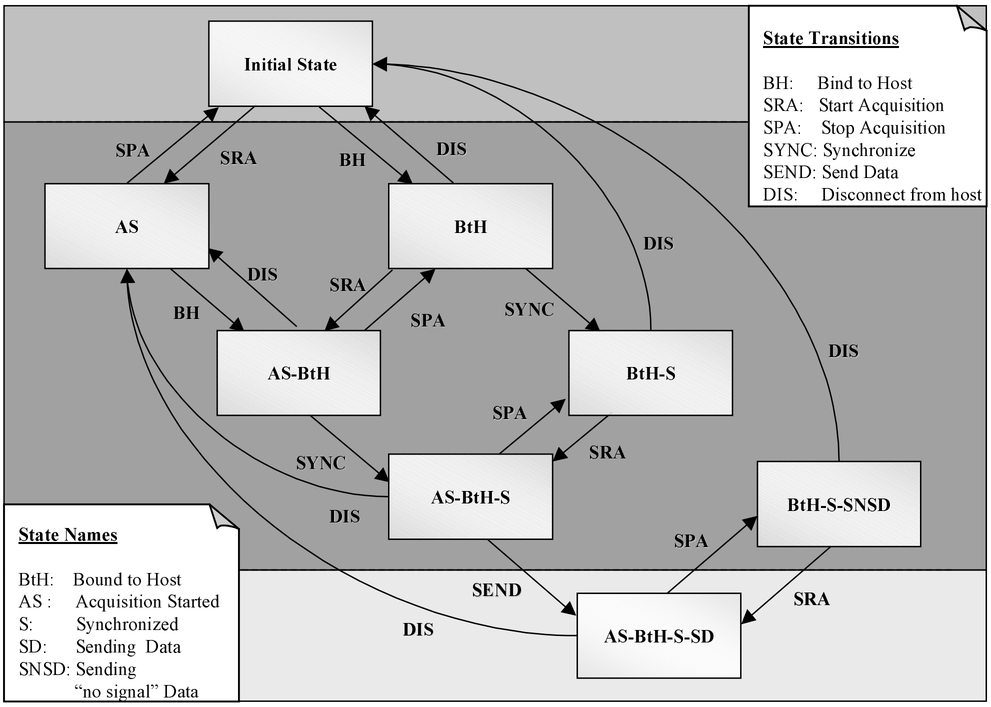


*Figure 7. State Transition Diagram for an Acquisition Agent Channel during the Communication with a Monitoring Agent.*

The acquisition agent's channel may release the resource if either the acquisition agent requests to disconnect its channel from the monitoring agent, or the monitoring agent decides to allocate the monitoring agent resource to the input from another acquisition channel. As Figure 8 displays, when either the acquisition or the monitoring agent request for disconnection, both agents release their references to each other.

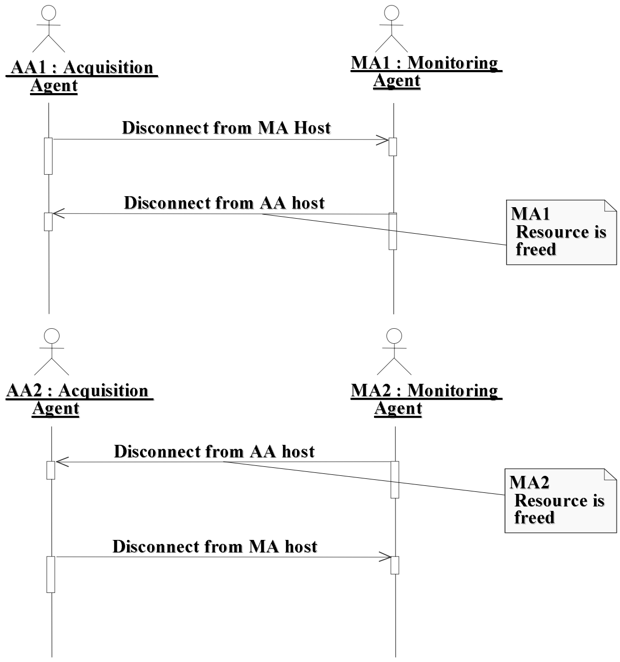


*Figure 8. Acquisition- Monitoring Agent Disconnection Scenarios.*

## 5. System Implementation

Four applications have been developed so far, which comprise the ICU monitoring system. The first of these, named "Continuous Recordings Viewer" (*CR Viewer*) handles the monitoring and storage of high frequency continuous recordings and embodies a monitoring agent. The second application, named "Simple Measurements Viewer" (*SM Viewer*) monitors and stores simple numeric patient measurements (simple physiological measurements), and also embodies a monitoring agent. An "*ICU Demographic Data Viewer*" is the database interface for patient demographics. Finally, a distributed "*Acquisition Station*" with one embodied acquisition agent has also been developed, and performs the tasks of medical data acquisition and data transmission to the SM and/or CR Viewers. The topology of the architecture implemented is depicted in Figure 9.

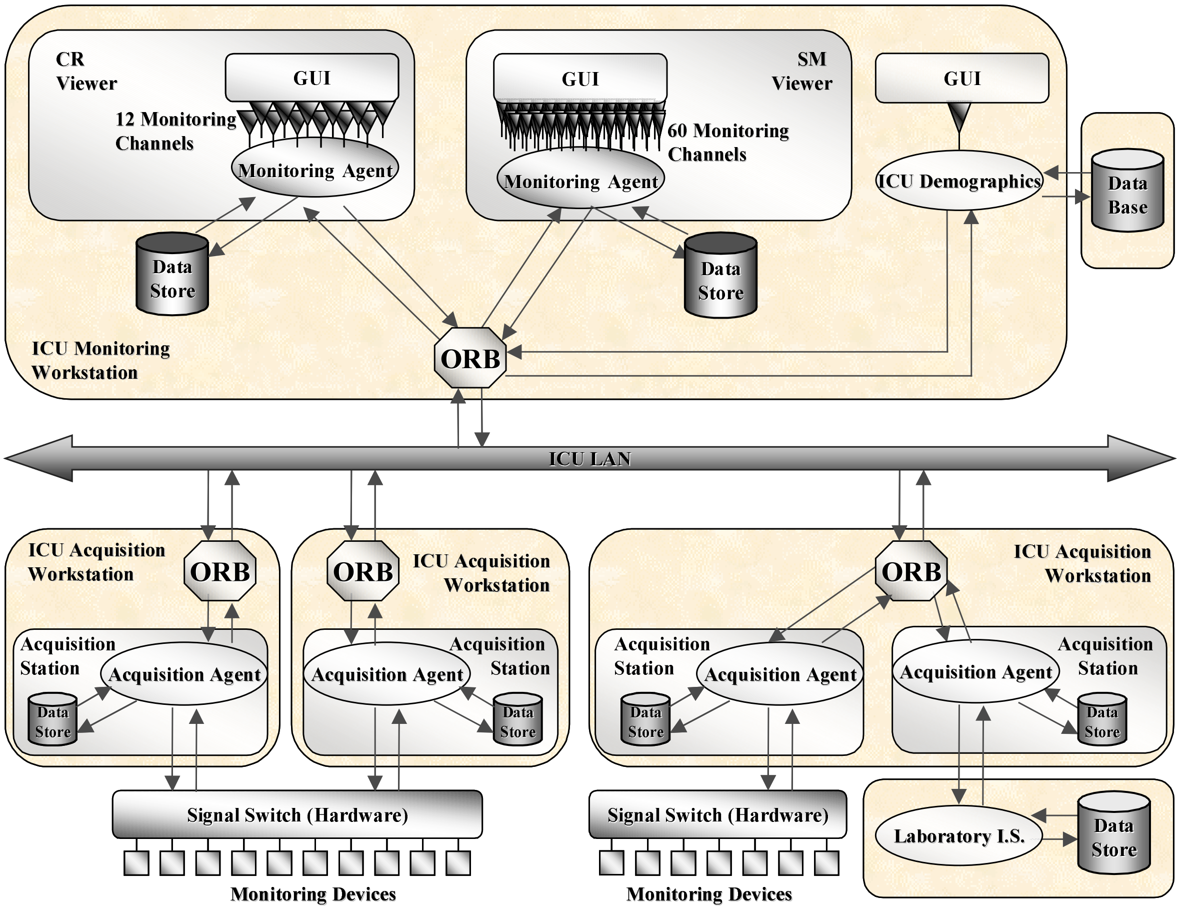


*Figure 9. The Architecture of the System Implemented by the CMI-HTA.*

The **CR Viewer** (Figure 10) has the following features:

- up to twelve input data channels can be served
- waveforms are presented to the user on-line
- each waveform carries its individual time stamp
- any number of 1-12 waveforms can be dynamically positioned on the presentation screen in various ways
- consequent presentation of the waveforms (in a form of a "slide show") is offered
- users can "freeze" any waveform in order to capture its view at a specific time
- daily overview of the waveforms is possible
- users can zoom in and out of each waveform
- import/ export functionalities are supported


*Figure 10. Typical CR Viewer Screen.*

The **SM Viewer** has the following features:

- up to sixty input data channels can be served
- real-time display of all changing measurements, their names and units of measurement is supported
- current hour's measurements' values are presented in a spreadsheet-like table
- current hour's measurements' trends can be drawn
- daily overview of all measurements via spreadsheet-like tables and diagrams is supported
- on-line data acquisition supported from CORBA enabled information systems

The **ICU Demographic Data Viewer** features the insertion/ update/ viewing of patient demographics and medical actions/ diagnoses concerning the specific patient, as well as of the medicines prescribed during the patient's stay at the ward.

The **Acquisition Station** can be used for the specification of a medical device to be used for the actual data acquisition (monitoring device), as well as for the specification of the appropriate settings for the communication with the device. An acquisition station can serve up to twelve input channels by means of its embodied acquisition agent. Its functionality also includes the connection and communication of each input channel with one monitoring agent. In other words, one acquisition station may communicate simultaneously with twelve monitoring agents, possibly each of whom is positioned on a different viewing application (CR and/ or SM). Figure 11 shows some typical acquisition station screens.

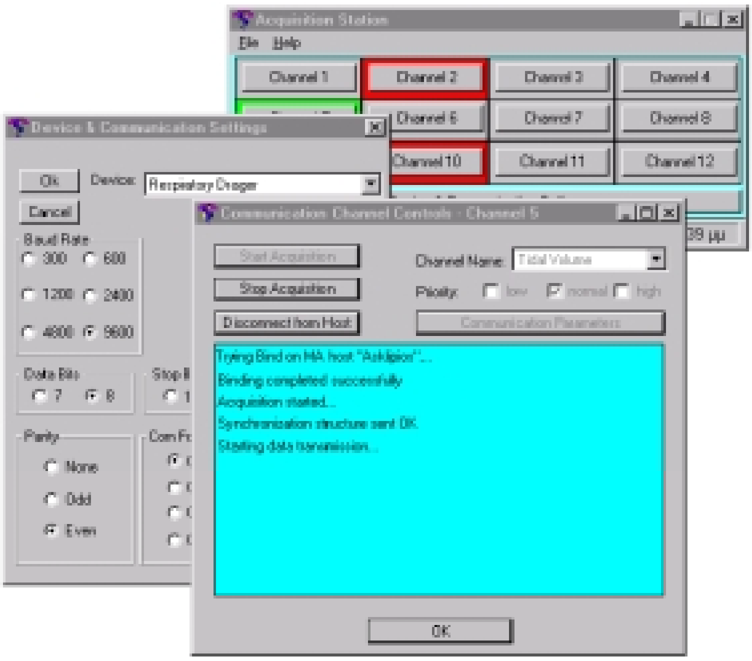


*Figure 11. Typical Acquisition Station Screens.*

## 6. Conclusions – Lessons Learned

Any ICU related monitoring information system is by nature, patient-centered and as such, it needs to be able to provide patient-related vital sign information as close as possible to the rate it is produced. Due to the complexity of the data modalities involved, and its data-integration, communication and presentation requirements, the ICU is exactly the place where location-independent access to patient data, multi-platform support, and the use of standard communication protocols makes more sense. The distributed object computing approach adopted takes care of packaging parameters appropriately for transmission over the communication link and syntactically allows components to be viewed as if they were local objects. The implemented system's response times within a local network testing area were not much higher compared to the response times observed when the communicating components were acting within the same address space. However, as it has already been mentioned in the Section 3.2 (Distributed Object Computing Middleware Infrastructure), no end-to-end Quality of Service (QoS) guarantees is provided at this time.

The presented approach enables the acquisition and communication of continuously fed, vital sign information from appropriate, distributed, real-time monitoring devices to corresponding ICU monitoring workstations. Most of the effort has been paid on describing the appropriate IDL interfaces, as well as the efficient management and handling of the vast amounts of the produced vital data. This allows for an integrated and uniform presentation of information that frees medical personnel from having to switch to and from various data modalities. The architecture is both extensible and able to be integrated with other clinical information systems, since it consists of collaborating components built with the combined use of existing standards and emerging technology.

The agent-based approach adopted ensures for the effective collaboration of the autonomous components and is capable of handling the complexity of the ICU environment. This is done by forcing individual software components to collectively respond to situations where timing requirements cannot be met, increasing thus predictability. The system dynamically adapts to the changes in environment and undesired events such as physical failures or malfunctions of the monitoring devices. This is done by performing continuous data validation and communicating the operation status to the monitoring agents. Because of its distributed nature, the system is able to continue operation even if certain components are out of reach (e.g. due to power failure) for a certain time because individual components perform data buffering continuously. During the congested failure-recovery periods, this leads to the need for communication channels load balancing, because of the bulk amounts of data that need to be propagated at that time.

The future of CORBA is very promising, particularly for high-performance real-time systems, since real-time system development strategies seem to migrate towards those used for "mainstream" real-time systems to achieve lower development cost and faster time to market [Schmidt98]. OMG on July 20, 1999 completed a vote to adopt the Real-time CORBA 1.0 specification in order to correct this deficiency by allowing clients to specify QoS policies at the ORB level. Nevertheless, no real-world products are expected to be ready earlier than mid-2000. CORBA 3, the first major addition to CORBA from the OMG since the Internet Inter-ORB Protocol (IIOP), is also expected to be released very soon. [Vinoski98] [Siegel99]. In this sense, CORBA has an advantage over DCOM and Java™ RMI since it can be integrated into a wider range of platforms, and programming languages. The flexibility and adaptability offered by CORBA make it very attractive for use in real-time systems, and the progress reported in this paper indicates that the real-time challenges can be overcome.

Other aspects that should be mentioned at this point is the fact that specialized personnel is required for configuring the acquisition agents to work properly with the various proprietary monitoring systems that are used in an ICU ward. All experiments required this configuration, which proved to be a hard task, because the hardware devices present in the ICU ward, use a plethora of different data transmission protocols. The fact that once the system is up and running, keystroking is kept minimal, has been really appreciated. On the other hand, the real costs of the ICU information systems are usually high. The most cost-effective way to make the ICU staff to utilize computers and include them in their daily practice is by providing all the functionalities they require. This is usually done at the cost of certain architectural compromises, but still the development of health informatics and services when based on definitions and requirements in close cooperation with healthcare professionals sometimes adapt technology standards to application requirements.

## 7. Discussion – Future Work

The presented work comes as complementary to the continuous effort of CMI-HTA to deliver seamless access to healthcare information over HYGEIAnet, the regional health-telematics network of Crete. According to this approach, although the ICU ward, is autonomous and devoted to the delivery of a particular set of services, the desirable continuity of care requires that different medical centers, offering complementary services and different levels of expertise, exchange relevant patient data and operate in a co-operative working environment [Tsiknakis97]. The ICU in particular, is an example where timely access to a patient's Virtual Electronic Health Care Record (EHCR) by the authorized personnel ensures that diagnoses and treatment decisions are made with a full knowledge of the patient's history [Katehaki97]. Therefore it is very crucial for patient monitoring systems to integrate data generated in ICUs with other data sources such as a hospital information system (HIS).

Another issue currently under consideration deals with the examination of the appropriate strategies to be adopted for selecting and processing data to be recorded into the patient's EHCR. Although at the moment import/ export functionality for recording captured views of the patient's history is supported, only verified and annotated data by the care provider can be transmitted to the patient's EHCR. This procedure can be automated, enabling automatic recording of ICU information. This can be done by archiving median values for definable periods, standard deviation as measure of variability, or even the complete data set when in emergency.

Work under progress at the CMI-HTA includes also the design of one more agent type, called *Cognition Agent*. This type of agent will be used to facilitate data fusion. Cognition agents will apply proper sets of association rules to data received by various acquisition agents, and will be responsible for cognitive tasks. Under normal conditions, such an agent could be used for the extraction of composite information and/ or the management of certain control devices, by means of a co-operative monitoring agent. In emergency cases, the cognition agent will inform the proper monitoring agent, which in turn will notify ICU personnel, using special alarm activation hardware (Figure 12).

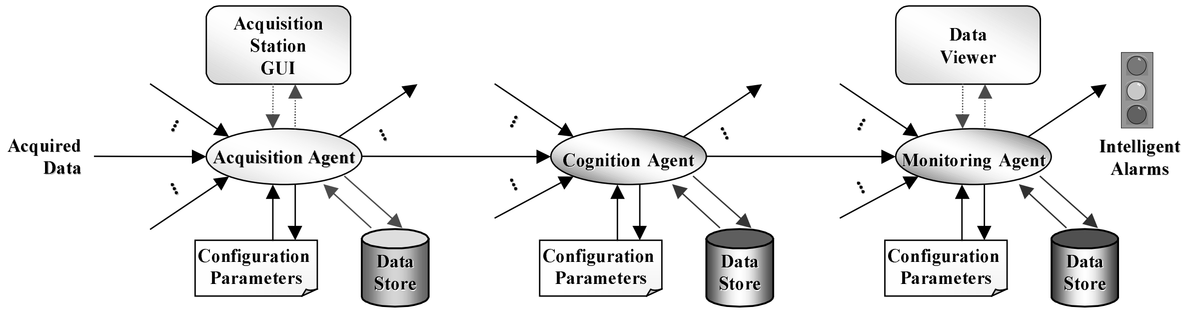


*Figure 12. Introducing Cognition Agent.*

Future work also includes the design of sophisticated cognition algorithms to enhance the agents' reasoning skills. The development of agents' knowledge bases is also in CMI-HTA's plans; in that case, the use of an Agent Communication Language (possibly KQML) to communicate knowledge between the agents will be considered.

## 8. Acknowledgements

The work presented has been funded in part by the General Secretariat for Research and Technology of the Hellenic Ministry for Development under project IHIS - Integrated Hospital Information System. The authors would also like to thank the ICU staff of the Evangelismos Hospital in Athens, Greece, for their contribution during the requirement analysis phase.

## 9. References

[Bellon94] Bellon E., Feron M., Marchal G., Suetens P., Van Den Bosch B., Bogaert J., Verschakelen J., Schetz M., Lauwers P., Oosterlinck A. and Baert A. L., "*Design for user efficiency in a dedicated ICU viewing station*", Medical Informatics, March 1994, 19(2), pp. 161-170.

[Box97] Box D., "*Essential COM*", Addison-Wesley, Reading, MA, 1997.

[Box99] Box D., "*Windows 2000 Brings Significant Refinements to the COM(+) Programming Model*", Microsoft Systems Journal, May 1999, 14(5), pp. 17-36.

[Brown98] Brown N., Kindel C., "*Distributed Component Object Model*", Microsoft Corporation, January 1998.

[Cohen91] Cohen P.R., and Levesque H.J., "*Teamwork*", Nous, 1991, 25(4), pp.487-512.

[Collura92] Collura T. F., Jacobs E. C., Burgess R. C. and Turnbull J. P., "*The Epilog System-Automated Long Term EEG Monitoring for Epilepsy*", IEEE Computer Magazine, September 1992, 25(9), pp.5-14.

[Collura93] Collura T. F., Jacobs E. C., Braun D. S., Burgess R. C., "*EView - A Workstation-Based Viewer for Intensive Clinical Electroencephalography*", IEEE Transactions in Biomedical Engineering, August 1993, 40 (8), pp. 736-744.

[Finin97] Finin T., Labrou Y., and Mayfield J., "*KQML as an agent communication language*", in Bradshaw J.(Ed.),"Software Agents", AAAI Press/MIT Press, 1997.

[Genesereth94] Genesereth M.R., and Katchpel S.P., "*Software Agents*", Communications of the ACM, 1994, 37(7), 48-53,147.

[Gokhale98] Gokhale A., and Schmidt D. C., "*Measuring and Optimizing CORBA Latency and Scalability Over High-speed Networks*", Transactions on Computing, 1998, 47(4).

[Hayes-Roth95] Hayes-Roth B., "*An Architecture for Adaptive Intelligent Systems*", Artificial Intelligence, 1995, 72(1-2), pp. 329-365.

[Hayes-Roth96] Hayes-Roth B., Larsson J. E., "*A domain-specific software architecture for a class of intelligent patient monitoring systems*", Journal of Experimental and Theoretical Artificial Intelligence, 1996, 8(2), pp. 149-171.

[Hedberg95] Hedberg S. R.: "*The first harvest of softbots looks promising*", IEEE Expert, August 1995, pp.6-9.

[Jennings96] Jennings N. and Wooldridge M., "*Software Agents*", IEE Review, January 1996, pp. 17-20.

[Katehakis97] Katehakis D. G., Tsiknakis M., Armaganidis A., Orphanoudakis S. C., "*Functional and Control Integration of an ICU, LIS and PACS Information System*", Proceedings of MIE'97, IOS Press, Porto Carras, Greece, May 25-29, 1997, pp. 15-19.

[Kirtland97] Kirtland M., "*The COM+ Programming Model Makes it Easy to Write Components in Any Language*", Microsoft Systems Journal, December 1997, 12(12), pp. 19-30.

[Kuhns99] Kuhns F., O'Ryan C., Schmidt D. C., Othman O., and Parsons J., "*The Design and Performance of a Pluggable Protocols Framework for Object Request Broker Middleware*", Proceedings of the IFIP 6th International Workshop on Protocols For High-Speed Networks (PfHSN'99)", Salem, MA, August 1999.

[Larsson97] Larsson J. E., Hayes-Roth B., Gaba D. M. and Smith B. E., "*Evaluation of a medical diagnosis system using simulator test scenarios*", Artificial Intelligence in Medicine, October 1997, 11(2), pp. 119-140.

[Larsson98] Larsson J. E., Hayes-Roth B., "*Guardian: An Intelligent Autonomous Agent for Medical Monitoring and Diagnosis*", IEEE Intelligent Systems, Jan-Feb. 1998, 13(1), pp. 58-64.

[Maes94] Maes P., "*Modeling Adaptive Autonomous Agents*", Artificial Life Journal, 1994, 1(1-2), MIT Press.

[Marzullo91] Marzullo K., Cooper R., Wood M. D., Birman K. P., "*Tools for Distributed Application Management*", IEEE Computer Magazine, August 1991, 24(8), pp. 42-51.

[Metniz95] Metnitz P. G. H., Lenz K., "*Patient data management systems in intensive care - the situation in Europe*", Intensive Care Medicine, Springer-Verlag 1995, 21(7), pp. 703-710.

[Miksch95] Miksch S., Horn W., Popow C., Paky F., "*Monitoring and Therapy Planning as a Real-World Problem: VIE-VENT*", International Journal of Clinical Monitoring and Computing, 1995, 12(2), pp. 118-119.

[Moret-Bonillo93] Moret-Bonillo V., Alonso-Betanzos A., García-Martín E., Cabrero-Canosa M., Guijarro-Berdiñas: "*The PATRICIA Project*", IEEE Engineering in Medicine and Biology, 1993, 12 (4), pp. 59-68.

[OMG98] Object Management Group, "*The Common Object Request Broker: Architecture and Specification (CORBA™)*", 2.2 ed., Object Management Group, Inc. Publications, February 1998.

[Pyarali96] Pyarali I., Harrison T.H. and Schmidt D.C., "*Design and Performance of an Object-Oriented Framework for High Performance Electronic Medical Imaging*", USENIX Conference on Object Oriented Technologies and Systems, Toronto, Canada, June 1996, pp. 191-208.

[Russel95] Russel S. & Norvig P.: "*Artificial Intelligence: A Modern Approach*", Prentice Hall Inc., 1995.

[Rutledge93] Rutledge G. W., Thomsen G. E., Farr B. R., Tovar M. A., Polaschek J. X., Beinlich I. A., Sheiner L. B., Fagan L. M., "*The design and implementation of a ventilator-management advisor*", Artificial Intelligence in Medicine, 1993, 5(1), pp. 67-82.

[Sandholm95] Sandholm T. W., and Lesser V.R.: "*Coalition Formation among Bounded Rational Agents*", Proceedings of IJCAI-95, Montreal 1995, pp. 662-669.

[Schmidt98] Schmidt D. C., Levine D. L., and Mungee S., "*The Design and Performance of Real-Time Object Request Brokers*", Computer Communications, April 1998, 21, pp. 294-324.

[Schmidt99] Schmidt D. C., Levine D. L., and Cleeland C., "*Architectures and Patterns for Developing High-performance, Real-time ORB Endsystems*", in the book series "Advances in Computers", by Zelkowitz M. (editor), Academic Press, to appear in 1999.

[Siegel99] Siegel J., "*A Preview of CORBA 3*", IEEE Computer Magazine, May 1999, 32(5), pp. 114-116.

[Stankovic88] Stankovic J. A., "*Misconceptions About Real-Time Computing. A Serious Problem for Next Generation Systems*", IEEE Computer Magazine, October 1988, 21(10), pp. 10-19.

[Stewart97] Stewart D.B., Volpe R.A. and Khosla P.K., "*Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects*", IEEE Transactions on Software Engineering, December 1997, 23(12), pp.759-776.

[Stone97] Stone P., and Veloso M., "*Multiagent Systems: A Survey from a Machine Learning Perspective*", Carnegie Mellon University CS Technical Report CMU-CS-97-193, December, 1997.

[Sun99] Sun Microsystems Inc., "*Java Remote Method Invocation - Distributed Computing For Java*", White Paper, May 1999 (http://java.sun.com/marketing/collateral/javarmi.html).

[Sycara96] Sycara K. and Zeng D., "*Coordination of multiple intelligent software agents*", International Journal of Cooperative Information Systems, 1996, 5(2-3), pp.181-211.

[Tambe96] Tambe M., "*Teamwork in Real-World, Dynamic Environments*", Proceedings of the International Conference on Mutli-Agent Systems (ICMAS), December 1996.

[Thull93] Thull B., Popp H.-J. and Rau G., "*Man-Machine Interaction in Critical Care Settings*", IEEE Engineering in Medicine and Biology, December 1993, 12(4), pp.42-49.

[Tonnesen97] Tonnesen A. S., "Critical Care Handbook", Departments of Anesthesiology & Surgery, University of Texas Medical School at Houston, ICU Information Systems Chapter, Copyright © 1995, 1996, 1997 (http://anes1.med.uth.tmc.edu/cc/is/infosys.html).

[Tsiknakis97] Tsiknakis M., Chronaki C. E., Kapidakis S., Nikolaou C., and Orphanoudakis S. C., "*An Integrated Architecture for the Provision of Health Telematic Services based on Digital Library Technologies*", International Journal on Digital Libraries, Special Issue on "Digital Libraries in Medicine", 1997, vol. 1(3), 257-277.

[Vinoski97] Vinoski S., "*CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*", IEEE Communications Magazine, February 1997, 35(2), pp. 46-55.

[Vinoski98] Vinoski S., "*New Features for CORBA 3.0*", Communications of the ACM, October 1998, 41(10), pp.44-52.

[Wooldridge95] Wooldridge M., and Jennings N. R.: "*Intelligent Agents: Theory and Practice*", Knowledge Engineering Review, 1995, 10(12), pp.115-152.

[Wollrath96] Wollrath A., Riggs R., and Waldo J., "*A Distributed Object Model for the Java System*", USENIX Computing Systems, vol. 9, November/ December 1996.