# The SIMON Architecture: Distributing Data, Tasks, and Knowledge in Intelligent ICU Monitoring Systems

Patrick R. Norris[1] , Karlkim Suwanmongkol[2], Antoine Geissbuhler[3]
and Benoit M. Dawant[2]

Vanderbilt University, [1] Department of Biomedical Engineering, [2] Department of Electrical and Computer Engineering, [3] Division of Biomedical Informatics
Box 1610, Station B, Nashville, Tennessee 37235
simon@vuse.vanderbilt.edu
http://www.vuse.vanderbilt.edu/~simon

**Abstract.** Otherwise successful research efforts in intelligent critical care monitoring often fail to achieve the same level of success when applied to actual clinical systems, or when additional tasks or knowledge, possibly outside the original problem domain, are added. A variety of factors contribute to this lack of success, including issues related to system reliability, scalability, and inherent design restrictions imposed by the underlying knowledge model of the system. These issues can and, in part, have been addressed by distributing data and monitoring tasks to separate modules. However, effective knowledge distribution remains a significant challenge. This paper reviews research efforts to distribute intelligent monitoring systems, and describes the SIMON (Signal Interpretation and MONitoring) architecture, which currently supports distributed tasks and data. Advantages and disadvantages of this distribution are detailed, and preliminary results of clinical testing are presented.

## 1 Introduction

Monitoring and caring for critically ill patients requires complex, timely analysis of many types of information. *Intelligent monitoring* is used to describe the task of knowledge-intensive integration of data from multiple sources where real-time constraints must be considered, and much work has been done to apply these techniques to the problems of intensive care unit monitoring. However, building and implementing these systems presents many difficulties. Some of the challenges can be addressed by partitioning a system into multiple components, and distributing data, tasks, and/or knowledge among these components. Typically, distribution is used to address issues including: 1) Performance: Distributing tasks over multiple processors and/or platforms allows work to be shared; 2) Reliability: Systems with multiple, redundant components may be less prone to failure; 3) Reuse: Components might be copied for use in other systems, or modified with minimal effects on the system as a whole; 4) Scalability: Properly designed systems may be scaled up by adding additional modules; and 5) Expandability: New functionality, such as interfaces to other systems, can be tested and added with minimal impact. Although much work remains, recent advances in technology and wide availability of low-cost PC systems facilitate efforts to solve these problems via distributed data, tasks, and knowledge in

intelligent critical care monitoring systems. The following brief review highlights research efforts, standards, and tools that enhance distribution. It is not exhaustive as space constraints prevent a complete listing of systems and research projects.

### 1.1 Distributing Data

Communicating data among system components is often the first challenge to be addressed, since tasks and knowledge cannot be distributed in practice without such communication. Even the most centralized monitoring systems typically require interfaces to "distributed" bedside medical devices. In most cases these interfaces are proprietary, and researchers have expended considerable work to simply collect and integrate data from distributed bedside devices [1]. Efforts to provide a standard protocol for communicating data from these devices is the focus of the IEEE 1073 standard (Medical Information Bus, or MIB) [2]. Increasingly, intelligent monitoring systems must share information with other hospital systems such as the clinical lab. Efforts to standardize this communication are also underway, in the form of the HL7 protocol [3].

In addition to collecting data from distributed sources, information must also be communicated between internal system components and user interfaces. Various approaches have been attempted including use of interprocess communication (IPC) [4] and network file sharing [5]. Recently, the world wide web (WWW) and hypertext transfer protocol (HTTP) have been used to provide users with distributed interfaces to medical monitoring systems [6], and to collect data from clinical users using web-based programming languages such as Java [7]. These and other communication standards have simplified the task of distributing data, although implementing standards often requires additional programming tools or resources, i.e., a web server for distributing WWW interfaces.

Overall, a variety of standards combined with past research success promise to make data distribution less problematic. Vendors of physiologic monitors and other bedside devices are beginning to incorporate standard interfaces that provide relatively easy access to some or all data [8]. There are still no universal solutions, though, so individual design and resource constraints are the main factors in choosing methods to communicate data between distributed components.

### 1.2 Distributing Tasks

Distributing data in intelligent monitoring systems is motivated by the need to support distributed tasks. Tasks are often classified by the level of abstraction at which they operate. Low level tasks are those devoted to acquisition of raw data, while higher level tasks identify features and perform complex reasoning [9]. Through the early 90's efforts to distribute tasks along these lines were motivated mainly by performance concerns. For example, the Intelligent Cardiovascular Monitor [10] used the process trellis architecture to ensure real-time performance of monitoring tasks, given sufficient numbers of processing units. Currently, the performance/price increase in readily available computing technology seems to be outpacing processing demands of most intelligent monitoring systems, making performance less of a

concern. Instead, researchers are distributing tasks to support system reliability, scalability, expandability, and component reuse. The NeoGanesh system for closed-loop control of mechanical ventilation is composed of distributed modules, or agents, in part "to facilitate future extensions and refinements, and to allow reuse of knowledge bases" [11].

A variety of tools and standards to support task distribution are becoming widely available. Proper use of object-oriented (OO) programming methods and languages such as C++ and Java facilitates development of individual system modules. Standards such as CORBA [12] and *de facto* standards such as Microsoft's DCOM [13] provide for distribution of object functionality across different processes, possibly across networked computer systems. Groups including the Andover Working Group [14] and CORBAmed [15] are developing information models to support component-based OO development in health care. From a theoretical perspective, substantial work in distributed artificial intelligence and multi-agent systems continues to advance methods for effectively dividing tasks in complex knowledge-based systems [16]. There is no universal consensus as to how to best distribute tasks in intelligent monitoring systems, and individual system requirements and available resources are still the main factors in choosing a solution.

### 1.3 Distributing Knowledge

The previous section described effective research efforts, tools, and standards to distribute tasks in intelligent monitoring systems, including tasks related to knowledge representation and processing. However, distributing knowledge-oriented tasks usually does not result in the expected benefits of modularization, from a knowledge engineering perspective. In fact, it is usually more difficult to modify, reuse, and augment the knowledge of a medical decision support system when the knowledge is divided among multiple components. Task-oriented modules usually compete for system resources and data, which are relatively easy to conceptualize, model, and allocate. Knowledge modules, in comparison, compete for something much less tangible: the right to govern a particular aspect of system behavior, given a particular system state and inputs. Typically this is not a problem in well-defined problem domains, as developers can ensure consistency among knowledge modules to eliminate potential inconsistencies and conflict. However, as systems are generalized and augmented to expand their usefulness, potential inconsistencies in the knowledge base (KB) rapidly become problematic [17]. Efforts to standardize and modularize medical knowledge representation such as the Arden syntax [18] are useful for building systems and encoding knowledge, but do not deal with the core problems related to medical knowledge distribution and augmentation.

Researchers are beginning to address these difficult challenges. Designing systems with reusable, domain-independent problem solving methods that operate on reusable domain ontologies (descriptions of domain concepts and relationships) holds promise for increasing reusability of problem-solving knowledge in medical informatics applications [17]. The MAITA (Monitoring, Analysis, and Interpretation Tool Arsenal) project is developing tools to allow rapid, distributed construction of ontologies and knowledge bases, to enhance the development and incremental augmentation of knowledge-based monitoring systems in non-medical domains [19].

In medical domains implementation of clinical guidelines poses similar knowledge representation and distribution problems, which is the focus of the ASGAARD project [20]. Issues of maintaining consistency in large, distributed KBs are also a concern in controlled medical terminology systems. These are being addressed, in part, by incorporating intelligent system modules responsible for maintaining KB consistency when new knowledge is added [21]. Finally, related work in cooperative multi-agent systems is being applied to patient care systems [22]. It is uncertain if current lines of research will adequately address the significant problems of medical knowledge distribution in intelligent monitoring systems, or if new methods will be needed.

To summarize this brief review, despite a great deal of research and attempts to distribute data, tasks, and knowledge in intelligent monitoring systems, there are still few universally accepted tools, standards, or methods. In most cases the individual implementation is the single most important factor in determining the best approach. A variety of choices regarding component distribution must be made during design, implementation, and testing phases. Some of the choices and factors influencing distribution and clinical use of the SIMON system are described in the next section.

## 2 Methods: The SIMON Architecture

The SIMON system [4] was re-designed to provide a significantly more distributed architecture, as shown in Figure 1. The overall motivation for distribution was to better support intelligent monitoring research in a clinical environment, and minimize system maintenance requirements.

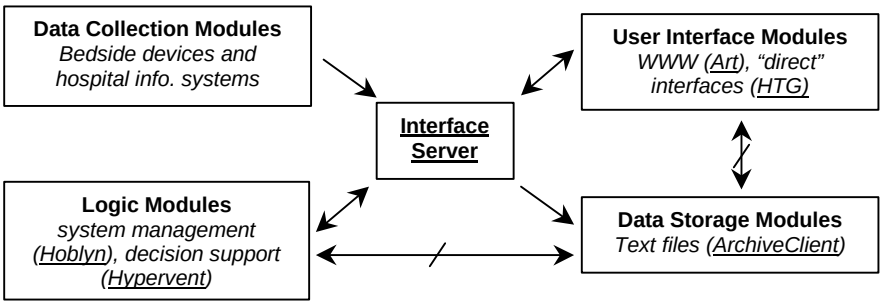


**Fig. 1.** Current SIMON architecture, organized by component function. Arrows show data flow, with plain arrows corresponding to TCP/IP sockets communication. Hashed arrows indicate *ad hoc* data sharing to support components that do not yet implement socket interfaces. Specific module names, described in the text, are underlined.

As such, the new system incorporates several significant changes. First, a central point of access to all system data was included to explicitly provide information translation and message routing functionality. Second, use of platform-neutral TCP/IP communication protocols was introduced. Finally, publish-subscribe methods were incorporated. In comparison, the original system components extensively used Unix

shared memory and remote procedure calls for communication between modules [4], which was difficult to extend and support in the current environment. The various component modules, and rationale for implementing them, are described below.

### 2.1 Interface Server

The Interface Server is the key to system distribution, as it is responsible for message routing and translation. Making these tasks explicit and dedicating them to a single module allows for a great deal of flexibility in designing and implementing other system components. Additional modules can be tested and added as needed to augment system functionality, and existing modules can be modified with minimal impact on existing components. In terms of software engineering, the SIMON Interface Server enables a degree of binary encapsulation between components. From a functional standpoint, monitoring systems research often requires relatively high data transfer rates, especially when physiologic waveforms are involved. For that reason the SIMON Interface Server incorporates a publish-subscribe system, in which new data can be continuously routed (published) to system modules as data is collected, given a single request (subscription). This enables higher efficiency than traditional approaches, where components issue explicit requests for new data periodically. The server also has mechanisms for queuing data, in the event that a network connection to a subscriber is interrupted, and will feed the queued data to the module if/when the connection is resumed. All data is time-stamped at the point of collection to ensure accurate interpretation by system components regardless of transit delays. Another important aspect of the Interface Server is that all communication with it must be via TCP/IP sockets. Although translation modules must be built to support data transfer for non-TCP/IP sources, this makes data routing much simpler. Routing becomes a matter of associating two ports, and any intermediate translation need only be concerned with the data format itself, not the underlying communication mechanisms. While this approach might be somewhat limiting in the long term (see Discussion below), it allows rapid addition of new system modules and easy modification of routing schemes to support the dynamic nature of research-based systems.

The Interface Server, and most other system components except where noted below, are currently implemented in C++ and run as Windows NT (Microsoft, Redmond, Washington) services. Running applications as services enables automatic startup when the system is booted, secure remote control though the Windows NT Server remote service manager, and operation independent of user sessions.

### 2.2 Data Collection Modules

The data collection modules interface with various bedside devices and hospital information systems such as the clinical lab, and relay this data to the Interface Server. We have implemented a flexible approach to collecting data from bedside medical equipment with RS-232 serial interfaces, using RS-232 to Ethernet bridges. These compact devices are easily mounted near the bed, and allow a remote computer to transparently access up to eight RS-232 serial ports via a 10BT Ethernet network.

Furthermore, multiple devices can be accessed from a single computer, and only performance concerns and limits of the operating system restrict the total number of serial ports. Thus, a single computer at a remote location can connect directly to large numbers of RS-232 bedside medical equipment at multiple beds. Conversely, multiple computers can interface to different devices at a single bedside, if high-performance data collection from multiple devices is required. This provides great flexibility in implementing bedside data collection modules.

In practice, several concerns must be addressed when interfacing bedside medical devices via distributed networks. In addition to bandwidth issues there are security concerns, as an individual's medical data is being transmitted and there is the remote potential to disrupt operation of certain bedside devices via serial interfaces. Also, reliability may be lower than direct serial connections, especially if network load is high. The current SIMON implementation addresses these issues by using a dedicated, isolated Ethernet network for all device communication. In addition, use of separate data collection modules for each medical device improves reliability, reducing the potential for failures to affect multiple devices. The current implementation supports data collection from physiologic monitors, IV pumps, ventilators, and cardiac output machines. Figure 2 shows a sample of actual data collected by the system.

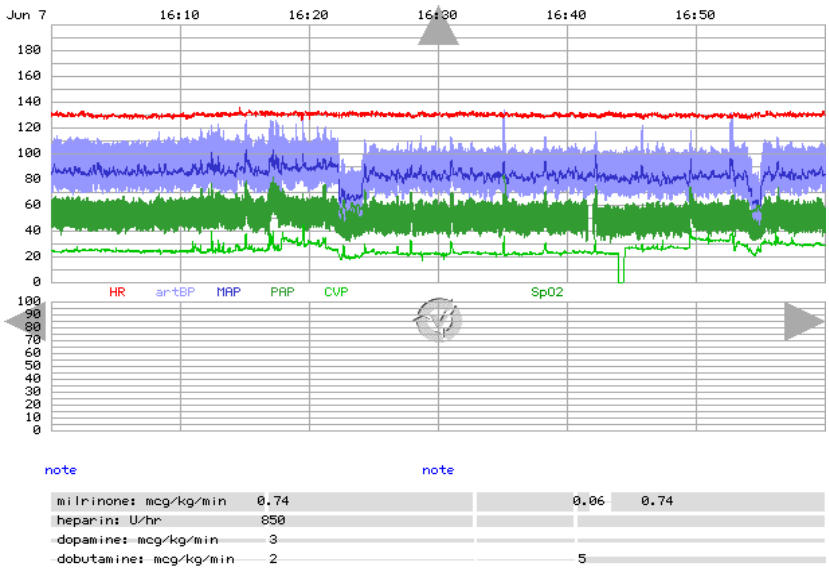


**Fig. 2.** Bedside device data collected once per second, plotted over one hour. The upper plots show cardiovascular data, including heart rate and catheter pressure readings. Respiratory data including pulse oximetry and ventilator data would appear in the middle plot if this patient had required such monitoring. The lower bar plots show IV drug infusion rates, with the bar thickness corresponding to normalized dosage.

## 2.3 User Interfaces

The World Wide Web is often used to provide user interfaces to data. Web technology provides a means of rapidly developing distributed clients and is highly desirable in most cases. However, direct web access to the SIMON Interface Server is undesirable for several reasons. First, presentation of data via the web is resource intensive, both in terms of hypertext markup language (html) formatting and implementing http protocols. For example, the graph shown in Fig. 2 is used in a current web interface, and must be generated from substantial amounts of monitor data. If current displays are needed these graphs must be re-created often. Furthermore, incorporating a web server sacrifices the flexibility to move or reconfigure Interface Server(s), since distributed clients (which may not be under the project's direct control) would rely on the server being at a given address. Finally, both commercial and public domain web servers are widely available, and have additional features such as support for secure protocols and more efficient handling of client requests. As such, the current implementation incorporates an additional module called Art to format data and provide it to a web server. Art does not yet interface directly to the Interface Server, but accesses data by reading text files generated by another module, as indicated by the hashed arrow in Figure 1.

In some cases user interfaces do not need to include formatted data or be widely distributed, and it is usually faster to develop them for the native target platform. For example, Figure 3 shows the prototype HTG (Head Trauma GUI), a SIMON interface for use on a dedicated workstation adjacent to a bed near the bedside of patients with traumatic head injuries. Since it has been designed for use at only a few beds simultaneously and extensive data formatting is not required, it is implemented in C++ and accesses the Interface Server directly.

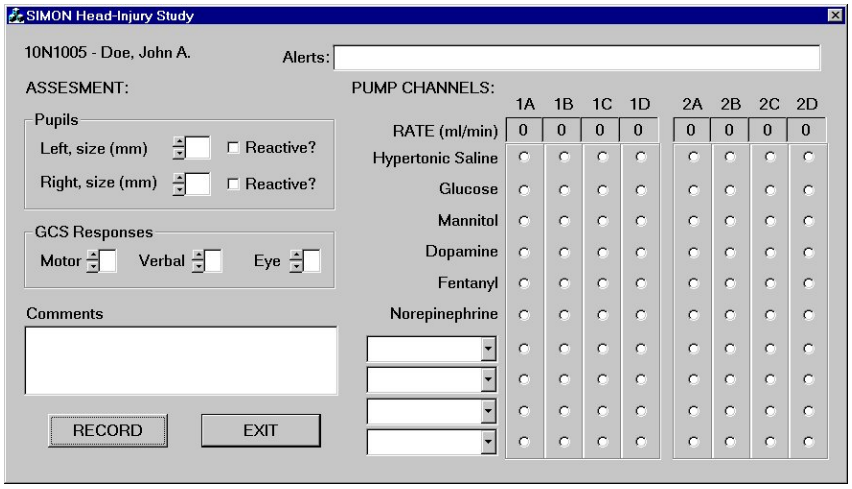


**Fig. 3.** Prototype of Head Trauma GUI, a bedside user interface. Users can enter periodic patient assessments and identify the drugs being given through IV pumps.

### 2.4 Other Modules

Additional system modules are responsible for archiving and accessing data, and for implementing logic to support system management and decision support tasks. Most of these components are under development or are being ported from the previous SIMON architecture, so their implementation is not entirely consistent with the reference architecture. Hash marks on arrows in Figure 1 indicate these inconsistencies, by showing data transfers that should be handled via the Interface Server.

In terms of data storage and system management, the ArchiveClient module subscribes to data from the Interface Server, and stores it in ASCII text files as it is received. ArchiveClient was designed primarily to support system components that could not be quickly adapted to the new architecture, such as the Art and Hoblyn modules, and to provide an easy way to store data. Hoblyn performs system and data management tasks, such as maintaining patient demographic information and archiving datasets as bed occupancy changes. Hoblyn is currently implemented as a finite state machine in PERL, and will eventually access data via the Interface Server as opposed to ASCII text files.

In addition to logic for supporting patient identification and associated data management, logic related to medical decision support is being implemented in distributed modules. The Hypervent module is the first attempt to encode medical knowledge in the system, and will provide basic alerts related to treatment of head injury patients on ventilators. Currently being prototyped as a PERL script, hypervent identifies a hyperventilated state based on ventilator settings, and then monitors arterial blood gas results to ensure the desired pCO2 is obtained. Alerts generated by hypervent will be passed to the user interface shown in Figure 3 via the Interface Server. While the architecture supports easy addition of modules to incorporate decision support knowledge into the system, it does not adequately address the issues of effective knowledge distribution mentioned in section 1.3. Overall, though, the current SIMON implementation provides effective distribution of data and some tasks, and has been successfully tested in the clinical environment. Results of this preliminary testing follow.

## 3 Results

SIMON has been operational on a single test bed in the VUMC trauma unit since September, 1998, and an additional test bed was added in May, 1999. As of that date, data had been collected for 60 patients over most of their stay in the test beds. Reliability was generally high and the system required attention less than once per week. Most failures occurred during initial testing, and the specific causes for almost all failures were identified and corrected. Some of this success in the initial field test may be attributed to the distributed architecture, which allowed for easier reliability testing during development, and "on-the-fly" modification of system components. In addition, the component-based architecture allowed for easy addition of the second bed. All hardware installation and software configuration was completed in under two hours, and data from the original bed was interrupted for under ten minutes.

In April 1999, physicians and nurses began reviewing the data graphs shown in Figure 2, at a bedside WWW interface during the course of patient care. After three weeks one physician reported modifying patient treatment based on observed trends in intracranial pressure (ICP) and cerebral perfusion pressure. The nurse was also instructed to use the graphs as an aid in determining when to administer drugs to reduce ICP.

## 4 Discussion

The current architecture supports distributed tasks and data in a critical care intelligent monitoring system intended for research and development use. Its success can be evaluated in terms of the general factors that motivate system distribution, as outlined in the introduction. First, SIMON achieves a degree of scalability, in that individual component modules can be added to support additional tasks for new devices, over multiple beds. Currently, resources of the Interface Server are the main limiting factor in how many modules can be added to the system. Although the system has not approached limits of computation power or available ports, it has only handled 1 Hz parametric data on a routine basis. If high frequency waveform data is added scalability could be limited. This relates to issues of performance, and it is important to be aware of additional computational resources needed to coordinate distribution. In SIMON's case, these additional resources were well worth the overall benefits of modularizing the system, since SIMON does not incorporate many tasks which are data-transfer intensive. Similarly, reliability of distributed systems might not be higher if the shared resources to coordinate communication are not robust. Again, the individual project environment dictates benefits of distribution: SIMON has a dedicated network for device communication and a reliable platform to support system modules, and distribution increased reliability. Finally, the system is expandable, as the sockets-based interface to the Interface Server provides connectivity to and from a wide variety of computing platforms and languages. Programs written in PERL, Java, and C++, running on Windows NT or Solaris (Sun Microsystems, Palo Alto, California) platforms have successfully communicated with the Interface Server. Overall, the improvements in expandability and reliability are most important in achieving an architecture to support intelligent monitoring research in a clinical environment. Some improvements are needed, though, to completely achieve this objective.

In the short term, existing system modules will be upgraded to eliminate the *ad hoc* data communication occurring outside the Interface Server. Another short-term objective is to develop modules for relational database support, to allow querying historical data. However, since current data is available through the Interface Server, additional research efforts to provide clinical decision support based on short-term information have already begun, in the form of elementary knowledge modules such as Hypervent. In the long term, development of extensive modular knowledge bases will require addressing the challenging issues of knowledge distribution previously described. The SIMON architecture, by enabling flexible, reliable implementation of distributed components in a working clinical monitoring system, significantly enhances these research efforts.

*Acknowledgements: This work was supported in part by grants NIH LM-00053-01A1, NSF CDA-9617519, and the VUMC Divisions of Biomedical Informatics and Trauma.*

## References

1. East, T.D., Wallace, C.J., Morris, A.H., Gardner, R.M., and Westenskow, D.R.: Computers in Critical Care. Critical Care Nursing Clinics of North America **7** (1995) 203-216.
2. IEEE 1073 MIB Standards Committee Home Page: http://grouper.ieee.org/groups/mib/
3. Health Level 7 Home Page: http://www.hl7.org
4. Dawant, B.M., Uckun, S., Manders, E.J., Lindstrom, D.P.: The SIMON Project: Model-Based Signal Analysis and Interpretation in Intelligent Patient Monitoring. IEEE Engineering in Medicine and Biology **12:4** (1993) 82-91
5. Krieger, D., Burk, G., Sclabassi, R.J.: Neuronet: A Distributed Real-Time System for Monitoring Neurophysiologic Function in the Medical Environment. IEEE Computer **24:3** (1991) 45-55
6. Nenov, V.I., Klopp, J.: Remote Analysis of Physiologic Data From Neurosurgical ICU Patients. Journal of the American Medical Informatics Association **2** (1995) 273-284
7. Norris, P.R., Dawant, B.M., Geissbuhler, A.: Web-Based Data Integration and Annotation in the Intensive Care Unit. In: Masys, D. (ed.): Proceedings of the 1997 Annual Fall Symposium. Hanley and Belfus, Philadelphia (1997) 794-798
8. Metnitz, P.G.H.: Patient Data Management Systems in Intensive Care – the Situation in Europe. Intensive Care Medicine **21** (1995) 703-715
9. Coiera, E.W.: Intelligent Monitoring and Control of Dynamic Systems. Artificial Intelligence in Medicine **5** (1993) 1-8
10. Factor, M., Gelernter, D.H., Kolb, C.E., Sittig,D.F.: Real-Time Data Fusion in the Intensive Care Unit. IEEE Computer **24:11** (1991) 45-54
11. Dojat, M., Pachet, F., Guessoum, Z., Touchard, D., Harf, A., Brochard, L.: NeoGanesh: A Working System for the Automated Control of Assisted Ventilation in ICUs. Artificial Intelligence in Medicine **11** (1997) 97-117
12. Object Management Group Home Page: http://www.omg.org
13. Distributed Component Object Model (DCOM) – Downloads, Specifications, Samples, Papers, and Resources: http://www.microsoft.com/com/dcom.asp
14. HP Healthcare Andover Working Group Services Page: http://interactive.medical.hp.com/mpgawg/
15. CORBAmed Home Page: http://www.omg.org/homepages/corbamed/corbamed.htm
16. Nwana, H.S. Software Agents: An Overview. Knowledge Engineering Review **11:3** (1996) 1-40
17. Musen, M.A. Modern Architectures for Intelligent Systems: Reusable Ontologies and Problem Solving Methods. In: Chute, C. (ed.): Proceedings of the 1998 Annual Fall Symposium. Hanley and Belfus, Philadelphia (1997)
18. Pryor, T.A., Hripcsak, G.: The ARDEN Syntax for Medical Logic Modules. International Journal of Clinical Monitoring and Computing **10** (1993) 215-224
19. The MAITA Project Home Page: http://www.medg.lcs.mit.edu/projects/maita/
20. The Asgaard Project Home Page: http://smi-web.stanford.edu/projects/asgaard/
21. Cimino, J.J.: Distributed Cognition and Knowledge-Based Controlled Medical Terminologies. Artificial Intelligence in Medicine **12** (1998) 153-168
22. Lanzola, G., Falasconi, S., Stefanelli, M. Cooperative Software Agents for Patient Management. In: Barahona, P., Stefanelli, M., and Wyatt, J. (eds.): Artificial Intelligence In Medicine – AIME '95 Proceedings. Springer-Verlag, Berlin Heidelberg New York (1995) 173-184