

Service-oriented Subscription Management of Medical Decision Data in the Intensive Care Unit

S. Van Hoecke¹, J. Decruyenaere², C. Danneels², K. Taveirne¹, K. Colpaert², E. Hoste², B. Dhoedt¹, F. De Turck¹

¹Department of Information Technology, Ghent University, Ghent, Belgium

²Department of Intensive Care, Ghent University Hospital, Ghent, Belgium

Summary

Objectives: This paper addresses the design of a platform for the management of medical decision data in the ICU. Whenever new medical data from laboratories or monitors is available or at fixed times, the appropriate medical support services are activated and generate a medical alert or suggestion to the bedside terminal, the physician's PDA, smart phone or mailbox. Since future ICU systems will rely ever more on medical decision support, a generic and flexible subscription platform is of high importance.

Methods: Our platform is designed based on the principles of service-oriented architectures, and is fundamental for service deployment since the medical support services only need to implement their algorithm and can rely on the platform for general functionalities. A secure communication and execution environment are also provided.

Results: A prototype, where medical support services can be easily plugged in, has been implemented using Web service technology and is currently being evaluated by the Department of Intensive Care of the Ghent University Hospital. To illustrate the platform operation and performance, two prototype medical support services are used, showing that the extra response time introduced by the platform is less than 150 ms.

Conclusions: The platform allows for easy integration with hospital information systems. The platform is generic and offers user-friendly patient/service subscription, transparent data and service resource management and priority-based filtering of messages. The performance has been evaluated and it was shown that the response time of platform components is negligible compared to the execution time of the medical support services.

Keywords

Computer-assisted, decision-making, intensive care, information systems, web services

Methods Inf Med 2008; 47: 364–380

doi:10.3414/ME0480

1. Introduction

A computerized intensive care unit (ICU) is an extremely data-intensive environment, resulting in enormous databases. It is generally assumed that every patient generates around 16,000 different data values on a daily base. Morris [1] reports over 236 different variable categories in a medical ICU record and concludes that this far exceeds human intellectual capability. It is estimated that humans are capable of adequately managing only five to nine variables. Not only the amount of data, but also the heterogeneity calls for automated data processing in the ICU.

According to Weed [2], the road ahead requires the development of a framework in which the medical knowledge and workflows of physicians are embedded in tools and requiring the routine use of those tools in all decision-making by both providers and patients. Information technology can facilitate the abstraction of relevant information and support the physician through software services for medical decision support. A software component qualifies as a service when its business logic is protocol-independent, location-agnostic and contains no state so that the service cannot remember information or keep state from one invocation to another. Services perform their logic and return the result in a single call. Services do not contain presentation logic, so they may be reused across diverse applications. Typical examples of medical support services for the ICU can be an infection detection service, a service supporting the ventilatory strategy for patients, multi-service systems for prescribing antibiotics in the ICU or services predicting the outcome of critically ill patients. Within the ICU of Ghent University Hospital, several

implemented medical support services already exist, namely a sepsis detection service, a service detecting kidney dysfunction based on the RIFLE criteria [3], a service calculating the CPIS score [4], multi-service systems for prescribing antibiotics in the ICU and services calculating the SOFA score [5].

It is expected that in future ICU information systems, hundreds of medical support services will be active simultaneously in order to optimize the care of critically ill patients. However, providing a large number of medical support services requires a management platform for subscription, data handling as well as for the notification and presentation of results. Therefore, the only solution is to incorporate a service management platform which processes medical decision data from laboratories and bedside monitors.

In this paper, we present such a service-oriented platform for the management of medical decision data in the ICU. The presented platform allows physicians to subscribe to a patient/service combination using a simple graphical user interface, and classifies medical support messages according to priorities. High-priority messages and alarms are delivered directly to the physician's smart phone or PDA, while lower-priority message can be presented on the patient's bedside terminal or sent by e-mail. The platform is fundamental for service deployment since it simplifies medical support services by taking over the general functionalities such as subscription management, medical data retrieving, medical support service localization, delivery of the decision support and alarm messages, and service resource management. This way, services only need to implement the medical decision algorithms and are not over-

loaded with these general functionalities. The services only process the medical data, sent from laboratories or monitors, and generate a result set or medical suggestion. As a result, medical support services can easily be outsourced to external partners or even created by physicians. The platform decision support and alarm messages are processed by the platform and delivered to the patient's bedside terminal, or the treating physician's PDA, smart phone or in his mailbox.

Service-oriented architectures are an important current technology for distributed software systems. They allow for easy integration since the services are built on open standards and can work together without custom coding, even though they do not share the same language or application platform. Service-oriented architectures also allow easy reuse of components across applications. The services can be used and reused to support different medical decision algorithms by composing multiple services. This way, redundancies are eliminated and the platform can react on-demand to events from the environment. This makes service-oriented platforms particularly suitable for ICU automation and is an interesting alternative approach for agent-based platforms, presented in other work by the authors. In [6] the authors presented a five-layer software architecture based on component and middleware technologies, providing inherent support for flexible plug-in of medical decision support agents, whereas in [7], the authors presented an architecture for easy distribution of the agents along multiple workstations to execute them simultaneously using grid technology. However, due to the opportunities and growing importance of service-oriented architectures, the service-oriented approach has been the subject of recent research.

In [8] Lenz reported that the advantage of platform-independent distribution of information has led to euphoric visions of future integrated systems, but requires transparent access to heterogeneous information sources and needs to deal with essential problems related to integration of autonomous systems. The status of available clinical decision-support systems continues to change and implementations are making

strides [9-11]. However they are mostly two-layer architectures, containing a front layer and database layer, this way providing no means to efficiently integrate developed algorithms, scores and tools for medical decision support [11-14]. The addition of new medical decision rules and alerts within these systems is slow and difficult. Moreover, they i) lack general functionalities, such as subscription management, easy user interfaces and alert handling, or ii) use the same medical decision rules for all patients [11], resulting in difficulties and limitations for physicians to interact with the electronic alerting systems. The decision support should return a large added value with minimum effort required by the users. However, to the authors' knowledge, no such medical support service subscription and data management platform fulfilling these requirements has been reported upon yet.

The remainder of this paper is structured as follows: Section 2 details the objectives of our platform, whereas Section 3 deals with the methods, starting with a general concept, followed by a detailed platform component description and component interaction as well as some internal design details. Section 4 deals with the evaluation results and presents two prototype medical support services. A critical discussion of the platform and its expected benefits is presented in Section 5. Finally, in Section 6 the main conclusions are highlighted.

2. Objectives

The aim of this research is to design a platform that allows for the subscription management of the medical decision support data. The platform, called the Intensive Care Subscription Platform (ICSP), should offer advanced features listed below:

- 1) User-friendly patient/service subscription: Human-platform interaction for subscribing to patient/service combinations should be straightforward by using user-friendly user interfaces so that application training for the physicians can be minimized.
- 2) Transparent data retrieving: The platform should make data about a patient

apparent to a physician wherever and whenever he needs it and assist physicians in decision making, problem solving and undertaking actions. When new medical data is available, results should be instantly delivered to the physicians, while during ward rounds physicians should be able to actively query services for an overview of historical service outputs and results.

- 3) Medical support service localization: Medical support messages should be delivered in real time. Patient alerts should be sent to the appropriate physician as soon as possible. Therefore the platform should localize and select the best server for processing the medical data.
- 4) Priority-based filtering of medical support messages: Alerts should be clear, specific, and easy to interpret and limited in amount in order to improve acceptance rate. A distinction should be made between messages based on their importance. Alarm messages should be sent to the physician's smart phone or PDA, and the patient's bedside terminal, while e-mail can be used for sending reports for medical decision support and overviews of historical medical results should be presented at the patient's bedside terminal.
- 5) Dynamic runtime updating: In order to deliver the medical decision support messages as soon as possible, the platform needs to distribute the load. The platform should be scalable for future adding of new medical support services in a dynamical way so that the platform and current services can continue working.

Besides these functional requirements, the Intensive Care Subscription Platform should also fulfill the following requirements stated as follows:

- 1) A single point-of-failure should be avoided in the platform in order to ensure reliability. The system has to work correctly at all times and no messages should get lost, especially not alarm messages.
- 2) Several successful information systems have been introduced and implemented in hospitals. The platform should allow

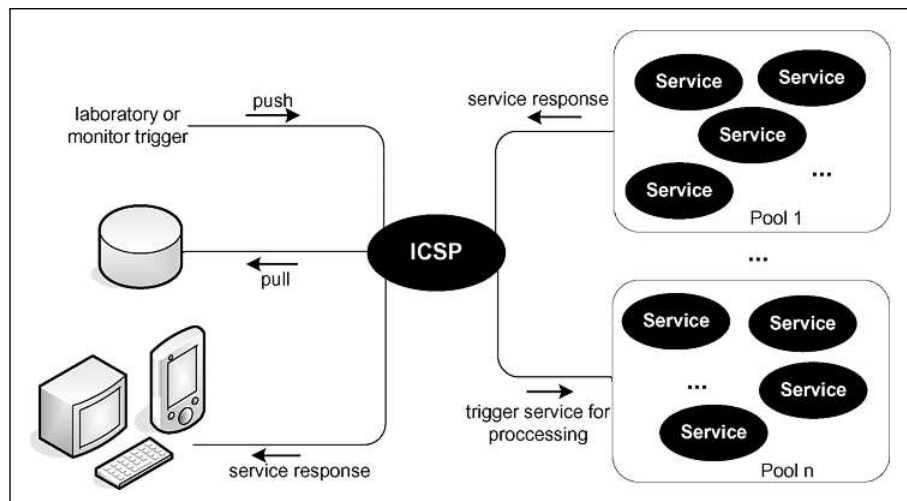


Fig. 1 General concept of the Intensive Care Subscription Platform (ICSP)

for easy integration with these different hospital information systems.

- 3) The platform should be generic and it should be possible to plug-in new components at runtime, independent of implementation languages, operating systems and hardware.
- 4) Security is a very important aspect of an ICU platform: the platform should provide a secure communication environment and a secure execution environment.
- 5) The platform performance should be such that the extra response time imposed by the platform and security mechanisms is small enough to be negligible compared to the execution time of the medical decision support services, while the platform scalability should be large enough to handle the execution of multiple simultaneous medical decision support services.

Within the Intensive Care Subscription Platform, laboratory, monitor or time-based triggers need to be able to activate one or more medical support services. The results from these services are not sent back to the initiators, but forwarded to the subscribers. As a consequence traditional request brokers cannot fulfill the requirements for our Intensive Care Subscription Platform; therefore a new platform is designed and presented in this manuscript.

3. Methods

The ICSP platform offers a generic environment, coupling all actors of the medical decision process and offering advanced features for subscribing to a patient/service combination, transparent data retrieving, medical support service localization and management, and delivery of medical support messages.

Section 3.1 details the general concept of the platform architecture, whereas Section 3.2 describes the platform components. Section 3.3 deals with the component interactions and Section 3.4 describes the internal design details.

3.1 General Concept

The main functionality of the Intensive Care Subscription Platform provides efficient management and data subscription for medical support services. Figure 1 illustrates the general concept.

As can be seen in this figure, laboratories or monitors provide medical data to the platform. This data is processed by the medical support services and, depending on the priority, results are sent to the patient's bedside terminal, the physician's PDA or smart phone, or to an e-mail address. There is also the possibility to activate services at regular

times (e.g. every night, every hour, etc.) by creating a cron trigger that will pull the required data from the HIS and activate the according medical support service for processing it and sending the results to the physician's e-mail address. During ward rounds or when needed, medical support services can also be invoked on request to query for overviews of historical decision outputs and results.

Suppose a physician wants to be notified immediately when there are changes in the kidney function of his patient. Therefore the physician has to run the client application and authenticate himself. Afterwards he can subscribe to the preferred patient/service combinations. From then on, whenever a new creatinine value is available from the laboratory, or a new urine measurement is entered in the intensive care patient database, the ICSP locates the appropriate service type for processing these data and selects the best or least loaded server running such a medical support service. The selected kidney service is then activated and calculates the new kidney status. If there are any changes in the status compared to the previous one, an alarm is sent to both the bedside monitor of the patient and to the physician's smart phone. The ICSP facilitates the abstraction of relevant information and supports the physician in medical decision-making. Due to the improved reaction times, care can be provided faster which could potentially lower patient's morbidity or mortality.

Our platform is designed based on the principles of service-oriented architectures, wherein all components are implemented as Web services. In view of the broad support for Web services and common XML-based standards, Web services are a promising concept for the integration of heterogeneous software components since applications can easily be distributed and expose well-defined functionality as a Web service. As will be motivated in Section 3.4.1, the Web service technology enables the required integration for the subscription and management platform.

By using the generic concepts of service-oriented computing, the platform presented in this paper is not restricted to appliance in intensive care. Due to its generic, dynamic

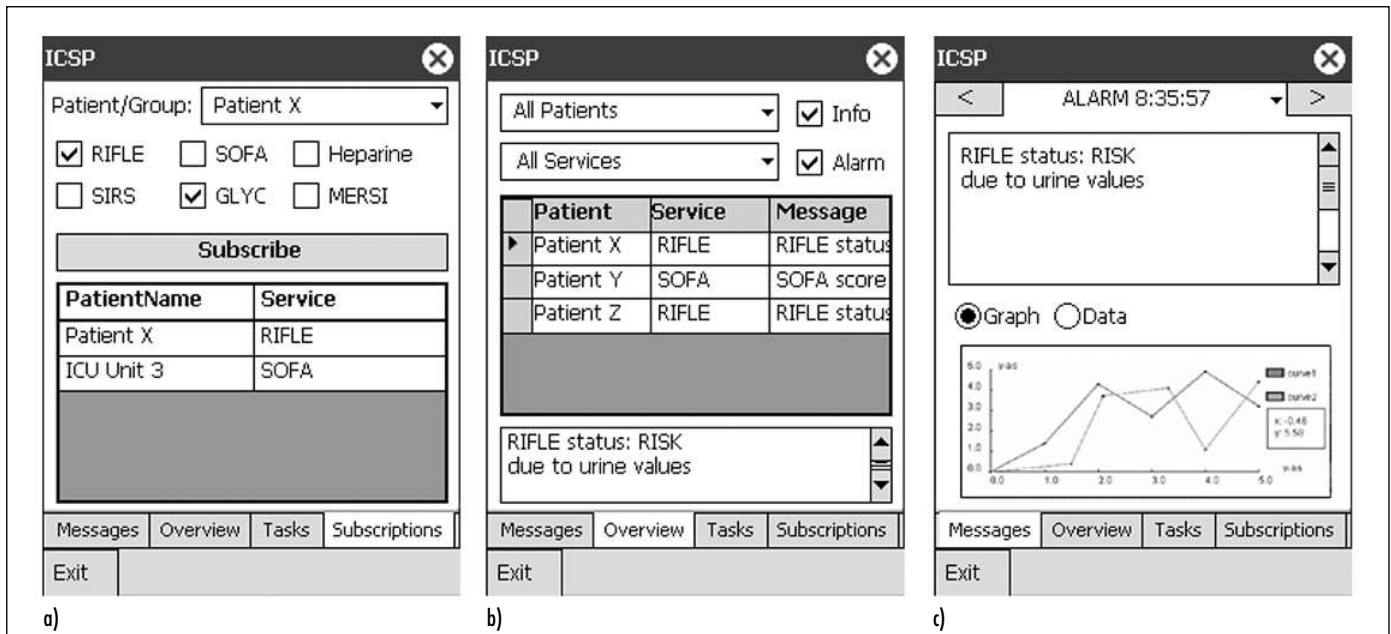


Fig. 2 a) Screenshot of the PDA subscription form for physicians; b) screenshot of the overview tab showing the alarm and info messages and filter options; c) screenshot of the messages tab where details (graphs, tables) of a single message can be viewed

and scalable design, it can be used within a wide range of applications. The platform acts as a generic communication system in which services can easily be plugged. The service functionality can cover areas other than medical decision-making.

As already stated, the medical support services can rely on the platform for the general functionalities detailed below.

3.1.1 Subscription Management

Physicians using the platform have to run the client application and authenticate themselves by using their electronic ID or login and password. Afterwards they can subscribe to the patient/service combinations for which they would like to receive support and alarm messages. This does not have to be a one on one relation and a subscription can be made for more than one patient at the time, such as ICU unit groups or all patients of the physician, respectively more than one service at the time. Besides subscriptions of their choice, each physician will also have some automatic subscriptions to obliged medical services based on their specialty in order not to miss important medical suggestions or alerts. Figure 2a presents a screenshot of this PDA

subscription form for the physicians. The subscription tab gives an overview of all active subscriptions for the current physician. Creating new subscriptions is done by filling in the form on the subscription tab page. After selecting a patient or patient group in this form, the already active subscriptions for this patient or patient group are marked and subscriptions can be deactivated or new ones can be added. Once this is done and subscribe is pressed, the subscriptions are updated in the subscription list. Patients discharged from the ICU are automatically unsubscribed and will no longer appear in the physicians' subscription list.

In order to improve the PDA's user interface, feedback of the end-users has been taken into account during development and implementation [15]. The PDA interface is tested by the physicians and has been judged simple and efficient. The abbreviations denoting services are not technical but are medical abbreviations carefully selected by the physicians and already used in paper workflows and algorithms.

3.1.2 Retrieving Medical Data

The platform supports push as well as pull models for data retrieving. Whenever new

data is available from laboratories or monitors, or at regular times, a trigger needs to activate the according medical support services to process the data. A trigger can be either a message (XML, SOAP or even plain text) from the laboratories or monitors, directly intercepted by the platform using the push model, or a cron job to poll for new medical data in the HIS database using the pull model. Since all monitor and laboratory data is added to the HIS database as well, one might think it should suffice to only support the pull model querying for new data in the HIS. However, due to the delays before this data is available in the HIS, a push model from the laboratory or monitors directly to the platform is needed as well.

3.1.3 Medical Support Service Localization for Processing the Medical Data

When new medical data is retrieved from a laboratory, database or monitor trigger, the ICSP locates the appropriate medical support services for processing this data and selects the best or least loaded server running such a service. Processing the data from both models, services can generate a result set, a medical suggestion or an

alert, tagged with a priority indicating the importance.

3.1.4 Delivery of Decision Support and Alarm Messages

The generated result set or medical suggestion is sent to the subscribed users. This way, users are notified by the ICSP whenever new medical data is available or alarms are generated. In order not to overload physicians with unimportant messages, the medical decision support messages are classified according to priorities. This way alarm messages are tagged as high priority, and delivered to the physician's PDA or smart phone, and at the patient's bedside terminal, while the decision support messages and reports can be received by e-mail. Figures 2b and 2c present screenshots of the PDA client application for the physicians.

As can be seen on Figure 2b, the overview tab shows all the alarm messages and info messages with additional filter options for only showing info messages or alarm messages respectively, or filtering on a specific patient or medical support service. Below the list, the text of the selected message is presented more clearly. If the physician wants to see more than just the text message, he can view the tables and graphs of the selected message in the messages tab, presented in Figure 2c.

The PDA client application also provides a tasks tab where physicians can actively query medical support services for an overview of historical service outputs and results.

3.1.5 Service Resource Management

In future ICU information systems, hundreds of medical support services will be active simultaneously in order to optimize the care of critically ill patients. Due to the large amount of input data to these services and the inherently complex algorithms, these services can impose an unacceptable high processing load when all executed on a single workstation, even when it is a powerful server. Therefore, medical support services are distributed along multiple pools. The service subsystem (consisting of service manager and service manager co-

ordinator) handles service activation and allows dynamically adding new services. Since the subsystem dynamically builds its capability list, the platform can continue working when new components are added.

3.1.6 Service Realization and Deployment

Medical support services can be easily plugged into the ICSP. The service functionality and choice for implementation language is completely free as long as the generic service interface presented in Section 3.4.3 is implemented. Whenever a new service is plugged into the platform, the service has to be registered to the Service Manager Coordinator and – if the service needs medical data for which triggers do not yet exist – a new adapter needs to be added to the Trigger Forwarder, handling the triggers of this new service. Both actions can be done using the Platform Manager. The platform can continue working whenever new services or adapters are added.

3.2 Platform Component Description

Figure 3 depicts the main software components of the platform. Below is a description of each individual component:

Trigger Forwarder (TF): The Trigger Forwarder is the top level component of the platform and handles the incoming triggers on one hand pushed from laboratories or monitors and on the other hand cron triggers pulling from databases. This component receives the triggers and transforms them, using an appropriate adapter from the adapter pool, to the internal generic trigger format based on XML. Afterwards the triggers are forwarded to the medical support services via the Event Handler (EH). By adding new adapters to the pool, compatibility with diverse laboratories, monitors and databases can be assured.

Communication Component (CC): The Communication Component handles the communication between the doctors' PDAs, bedside monitors and the platform. Through the graphical front-end of this component, care providers can manage their subscriptions. The CC also contains a data-

base, storing all the messages so that they do not get lost. In this database, messages are queued until the receiver comes online.

Subscription Component (SC): The Subscription Component receives new subscriptions from the Communication Component and manages these subscriptions in an internal database. This way the Trigger Forwarder can find out where to deliver the trigger data, while the Communication Component discovers which medical support services are processing data.

Database Manager (DBM): The Database Manager handles communication with external databases, containing patient records and monitoring results. The DBM provides a single interface for access by the services, independent of the background database or query language. Therefore the Database Manager has to use appropriate adapters to map the key values requested by the medical support services onto the correct queries. The DBM is also responsible for executing these queries in order to provide the requested data to the services.

Event Handler (EH): The Event Handler takes care of the communication between the medical support services and the rest of the platform. On one hand the EH offers the services a single interface to communicate with the platform, on the other hand services are abstracted as a single interface, simplifying complexity for the other platform components.

Service Manager Coordinator (SMC): The Service Manager Coordinator is responsible for the coordination of all messages from the platform to the services. In the Service Manager Coordinator all medical support services are registered and described in detail. When a new Application Server is installed in the system, the Service Manager of the pool it belongs to registers that Application Server and its services to the SMC. Medical support services can however also be discovered dynamically using Web service discovery standards. Since this incorporates some overhead, a trade-off has to be made between performance and dynamic discovery. Therefore the SMC uses a combination of registration mechanisms and dynamic discovery.

Based on the load information provided by the Pool Management Server Coordinator, the SMC can select the optimal Application Servers for executing medical support services by using advanced load-balancing algorithms.

Service Manager (SM): The Service Manager manages a pool of Application Servers. When a new Application Server (AS) is started, it has to register itself to the SM of that pool. The Service Manager then registers that Application Server and its medical support services to the Service Manager Coordinator. When new services have to be started on specific Application Servers, the Service Manager Coordinator contacts the Service Manager of the pools these ASs belong to. The SM then forwards the start commands to the correct AS. However, result sets from these Application Servers are not sent back to the Service Manager, but directly forwarded to the Event Handler.

Application Server (AS): Application Servers execute the actual medical support services. Each Application Server can implement one or more services, can have its own capabilities and can provide services using different programming languages. When a new AS is installed in the system, it must register itself to the Service Manager of the pool it belongs to.

Pool Management Server Coordinator (PMSC): The Pool Management Server Coordinator stores an overview of each server running an AS/LM pair. This information is updated at regular intervals by the Pool Management Server of each pool. At regular time intervals, an overview of the load of the platform is pushed from the PMSC to the Service Manager Coordinator. When selecting the required medical support services, the Service Manager Coordinator can request more detailed and up-to-date information about each server running an AS/LM pair from the PMSC.

Pool Management Server (PMS): Each pool has its own Pool Management Server. It gathers load information from the monitors in the pool. At regular intervals, the load information of the whole pool is reported to the Pool Management Server Coordinator. When a new Application Server is started, the load monitor it is paired with has to

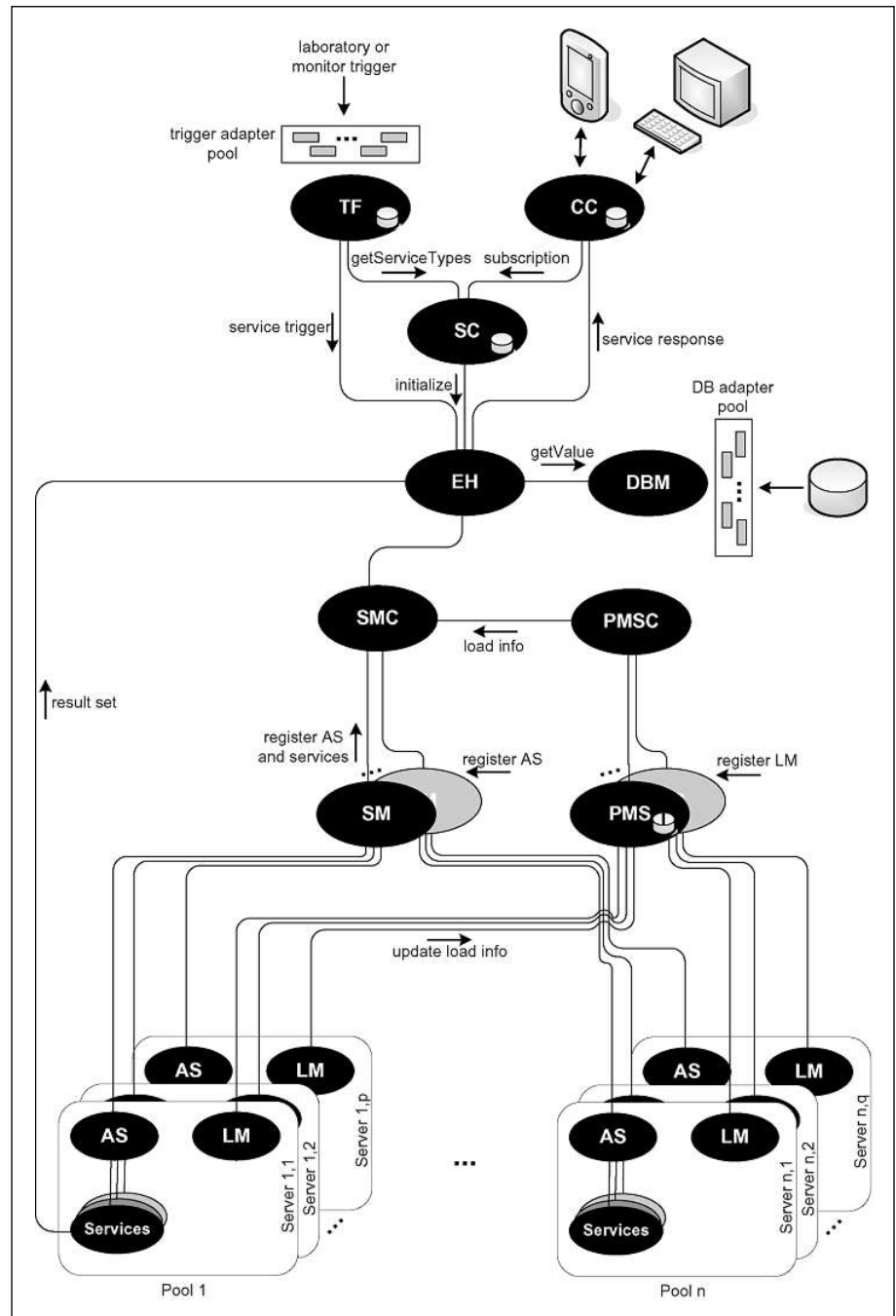


Fig. 3 Architecture of the management platform for medical decision data in ICU. The following components are shown: Trigger Forwarder (TF), Communication Component (CC), Subscription Component (SC), Event Handler (EH), Database Manager (DBM), Service Manager Coordinator (SMC), Service Manager (SM), Application Server (AS), Pool Management Server Coordinator (PMSC), Pool Management Server (PMS) and Load Monitor (LM).

register itself to the PMS of the pool it belongs to.

Load Monitor (LM): Each Application Server is paired with a Load Monitor that monitors the load characteristics of the machine it is running on.

At regular intervals the LM reports the load of its machine to the Pool Management Server of the pool it belongs to. When a new AS is started, the LM it is paired with has to register to the PMS of its pool.

3.3 Component Interaction

Two kinds of events can activate the medical support services in the platform, initiating new score calculations, suggestions, outcome predictions or history overviews. On one hand a physician can activate the medical support service by executing one of the commands of the service in his graphical client. On the other hand, as already stated, triggers can activate the according services. After transforming the trigger into the internal generic trigger format, the patient ID is retrieved from the trigger in order to find out from the Subscription Component which services are running for that patient. The Trigger Forwarder keeps a mapping of the different kind of triggers and their interested service types. Cross-cutting the results from the Subscription Component with the results from its own mapping, the Trigger Forwarder is aware of all interested medical support services running for this patient. This way, it can deliver the trigger correctly by activating the Event Handler for each service and each patient. The Event Handler forwards the trigger to the Service Subsystem, where the Service Manager Coordinator selects the correct Service Manager and Application Server in order to reach the running service.

When a service is activated, either by a command request or a trigger, the medical support service may need additional medical data from databases. Retrieving medical data from databases is supported by the pull system. The Database Manager handles this communication with external databases and provides a single interface for access by the medical support services, independent of the background database or query language. Therefore the Database Manager selects the appropriate adapter to map the key values requested by the services onto the correct queries. The DBM is also responsible for executing these queries in order to provide the requested data to the medical support services.

Processing these data from commands, triggers and databases, services generate a result set or medical suggestion and send it to the Event Handler, who delivers the messages to the subscribed users over the platform.

3.4 Internal Design Details

An ICSP platform prototype as well as some prototype medical support services have been implemented and are currently evaluated by the Department of Intensive Care of the Ghent University Hospital. Below is an overview of the main platform design details.

3.4.1 Web Services

The Intensive Care Subscription Platform is composed of multiple Web services as independent building blocks. The platform components are autonomous, independent of other services and only responsible for their own functionality. The services are loosely coupled through agreed interfaces. A Web service is defined by the W3C [16] as a software system designed to support interoperable machine-to-machine interaction over a network and is built up from standard internet protocols. A Web service has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards [16]. SOAP is a standardized packaging protocol for exchanging structured information between applications. A SOAP message consists of an envelope containing a header and a body. The header contains blocks of information relevant to how the message is to be processed, including routing and delivery settings, authentication or authorization assertions, and transaction contexts. The body contains the actual message to be delivered and processed. Based on the exchange of structured text messages, the interaction makes abstraction of the underlying technologies.

The open, XML-based standards, WSDL and SOAP, are widely accepted by the industry and guarantee platform-independency and a more flexible integration of applications. Many times in the past, the goal of perfect interoperability has been promised, although it has rarely been achieved. Since many major vendors are pushing Web services and have agreed to common XML-

based standards, Web services are a successful new concept of integration middleware.

The platform components and medical support services were implemented as Web services in the Microsoft .NET environment, using Visual Studio 2003. As an advantage of service-oriented platforms, a platform-independent communication mechanism is used and integration can be easily enabled with Java or J2EE services.

3.4.2 Generic Message Schema

When a medical support service has processed the incoming trigger, the results are parsed in an XML document, following the generic message schema presented in Figure 4.

Medical suggestion services can output messages, tables, and charts. Therefore the generic message schema contains three parts. The first part specifies the patient and medical support service, and gives a description of the message. That description can for example contain a textual advice. The second part provides functionality for sending chart data. It is up to the presentation component how to present the chart data. If the chart presentation tool provides line charts, angles and zooming, physicians will be able to zoom into the line chart created from the message's chart data, while other tools for example will present the data as a bar chart. In order to specify the chart, the service has to specify a list of coordinates. The third and final part of the message allows sending table items, this way presenting tables in the graphical client.

By using a generic message schema, generic client applications can be developed able of presenting all types of messages coming from the medical support services. It is up to the presentation component of the client application how to handle the data and how to present the messages, tables and charts. Since all messages are following the generic message schema, the client application can present the results from all the different types of medical support services.

3.4.3 Generic Service Interface

The platform acts as a generic communication system in which medical support ser-

```

<xs:element name="genericMsg">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="message" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="patientId" type="xs:string"/>
            <xs:element name="serviceType" type="xs:string"/>
            <xs:element name="message" type="xs:string"/>
            <xs:element name="description" type="xs:string"/>
            <xs:element name="chart" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>...</xs:complexType>
            </xs:element>
            <xs:element name="table" minOccurs="0">
              <xs:complexType>...</xs:complexType>
            </xs:element>
            <xs:element name="messageType" type="xs:string"/>
            <xs:element name="dateTime" type="xs:dateTime"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Fig. 4 Generic message schema for the output of the services. An output message can contain a patient id, service type, message, description, chart, table, message type and a date/time.

vices can easily be plugged. Therefore the platform offers a generic interface for the services. The service functionality and implementation language is completely free as long as they implement the generic service interface.

As can be seen in Figure 5, the generic WSDL interface consists of four methods:

- 1) *init*: actions to be taken when a medical support service is initialized for a new subscribed patient;
- 2) *trigger*: actions to be taken when a trigger arrives at the service. This method contains the main functionality of the service;
- 3) *getCommands*: returns the commands of the medical support service;
- 4) *executeCommands*: execution of a command of the service.

3.4.4 Platform Management

In order to ease platform management, an additional component is implemented in the

platform, namely the Platform Manager (PM). The Platform Manager can interact directly with the Trigger Forwarder, the Event Handler and the Database Manager for configuration purposes. The Platform Manager can configure the Trigger Forwarder and can add new adapters to the trigger adapter pool so that the Trigger Forwarder is able to intercept a new trigger in future. The medical support services can also be configured by adjusting the Event Handler properties through the Platform Manager. When new databases are added to the platform and the appropriate DB adapter in the pool does not yet exist, an adapter can also be added through this Platform Manager. Therefore, the PM provides an easy interface to set the SQL mapping for the key values used in the platform.

3.4.5 Security

Within traditional channel security, plain-text messages are sent into a secure channel.

The Web service technology however introduces message-level security. By using the Web service technology for the platform implementation, additional security mechanisms are possible. Channel security and message security correspond to a different security context. Since all data is sent through a protected tunnel, channel security is well suited for protection of a point-to-point communication. Once the data leaves the tunnel, protection is no longer applied and thus messages cannot stay encrypted in databases or over multiple hops. Channel security is typically achieved using the well performing Secure Socket Layer (SSL) or the Transport Layer Security (TLS) protocol. On the other hand, message level security allows establishing an end-to-end security context across multiple point-to-point connections so that the data remains secure until the message reaches the ultimate receiver. In addition, message security provides a way to protect only one part of the message in order to limit the overhead for

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions ...>
  <wsdl:types>
    <s:schema ...>
      <s:import namespace="http://icsp.intec.ugent.be/triggers.xsd" />
      <s:import namespace="http://icsp.intec.ugent.be/commandsSchema.xsd" />
      <s:import namespace="http://icsp.intec.ugent.be/genericMsgSchema.xsd" />

      <s:element name="init">...</s:element>
      <s:element name="initResponse">...</s:element>
      <s:element name="trigger">...</s:element>
      <s:element name="triggerResponse">...</s:element>
      <s:element name="getCommands">...</s:element>
      <s:element name="getCommandsResponse">...</s:element>
      <s:element name="executeCommand">...</s:element>
      <s:element name="executeCommandResponse">...</s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message ...>...</wsdl:message>
  <wsdl:portType ...>...</wsdl:portType>
  <wsdl:binding ...>...</wsdl:binding>
</wsdl:definitions>
```

Fig. 5 Generic WSDL interface for the medical support services. This interface consists of four methods: `init`, `trigger`, `getCommands` and `executeCommands`, and their responses.

encryption and decryption. Message security ensures complete independence of the underlying (insecure) transport layer.

Since our subscription platform model operates on logical endpoints that abstract the physical components and application infrastructure, it incorporates multi-hop technology with intermediate actors. These intermediaries need to scan portions of the SOAP message header for routing and management purposes but do not need to be able to read the complete content.

As will be motivated in Section 4, secure Web services perform better using SSL, thus transport layer security, instead of WSS message level encryption. Since SSL is more mature, tools are already optimized to provide this security technology. After an asymmetric handshake, SSL uses symmetric encryption while WSS keeps using the more calculating-intensive asymmetric encryption.

Although less efficient, WSS has its own advantages such as independence of the transport layer, the possibility to secure data

over multiple SOAP hops and the protection against repudiation.

By using SSL for point-to-point interactions and WSS for multi-hop interactions, a trade-off is made between performance (SSL) and security functionality (WSS) in order to secure the subscription platform to the best. Result sets from the medical support services are not sent back to the Service Manager and Service Manager Coordinator, but are directly forwarded to the Event Handler, eliminating redundant intermediaries. This way service result sets only have to cross the Event Handler and the Communication Component before reaching their destination. These intermediary components only scan portions of the message header for routing, and should thus not be able to read the message body content. By encrypting the result set messages using Web service security at message layer, it is assured that the result sets cannot be intercepted or altered by malicious users, independent of the transport layer to be secure or not. Only the sender (i.e. medical support

service) and the ultimate receiver (e.g. doctor's PDA or bedside monitor) are able to read the complete message content.

Component configuration by the Platform Manager requires only point-to-point interactions without intermediaries. In this case, secure communication is enabled by using the transport layer security SSL, since this performs better than WSS and no intermediaries have to be crossed.

3.4.6 Reliability

Since the ICSP platform has to work correctly at all times, the platform is decomposed into multiple functional components that can be coupled dynamically. Dynamically replacing one component with another one, in case of failure or overloading, avoids single point-of-failures and ensures the required platform reliability.

Besides platform reliability, no messages should get lost and alert escalation is required for high-priority messages. Therefore ICSP, more specifically the Communi-

ication Component, uses WS-ReliableMessaging for sending the medical suggestions, results and alerts to the subscribed users. WS-ReliableMessaging describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures. The protocol supports different delivery assurances such as *AtLeastOnce*, *AtMostOnce*, *ExactlyOnce* and *InOrder*. If the Communication Component cannot transmit the message to the recipient for some reason, for example, if the physician is on vacation and consequently his PDA cannot be reached, an exception is raised so that the Communication Component knows the message was not transmitted. As a consequence, the messages are queued in the Communication Component until the receiver comes online. For high-priority messages, alert escalation is provided, so the Communication Component will re-send alarms and high-priority messages to a colleague physician with the same specialty/role when the subscribed receiver is not online. The roles and specialties of all physicians within ICU are therefore stored at the Communication Component.

By using WS-ReliableMessaging, guaranteed message delivery of the medical suggestions, results or alarms is ensured.

3.4.7 Service Discovery

When a new Application Server is installed in the system, the Service Manager of the pool it belongs to registers that Application Server and its services to the SMC. Services can however also dynamically be discovered using Web service discovery standards. Since this incorporates some overhead, a trade-off has to be made between performance and dynamic discovery. Therefore a combination of registration mechanisms and dynamic discovery is used. An Application Server uses a service registry or ontology-enhanced search engine to locate the services automatically. The Application Server itself proactively advertises itself with the service registry of the Service Manager so that requesters can find its services when they query the registry. The Service Manager then registers an overview of the Application Servers and their services to the

Service Manager Coordinator so that the SMC has an overview of all available medical support services in the platform.

3.4.8 Service Selection

In order to process the trigger or command data, the SMC component implements load balancing algorithms for dynamically selecting medical support services, based on the functionality of the services and the load of the Application Servers. Several load balancing algorithms can be used, ranging from simple random, round robin or least connected algorithms, to more sophisticated ones such as weighted load balancing, minimum load, or other custom-tailored algorithms. The SMC currently implements the minimum load algorithm for load balancing, presented by the authors in [7], since this algorithm outperforms the other load balancing algorithms.

Supplied with the service interfaces and corresponding server load information, the SMC can select the best services for processing the data. When one of the selected services suddenly slows down or becomes unavailable, the SMC chooses an alternative medical support service. This selection process is repeated until the data is processed.

3.4.9 Service Databases

Some medical support services require a history per patient of measurements or calculated scores. Therefore each service can save this content in a dedicated database. Since the platform dynamically selects the services based on the load on the workstations, different requests for one medical support service can be handled by different instances of that service. Therefore all instances of one service use a common database to store all derived abstractions, calculated scores, and service states or history overviews of measurements.

Moreover missing information regarding physiological parameters, laboratory parameters, microbial results, antibiotic use and ICU scores are extracted daily from the HIS to complete the database and allow the development of auxiliary management services to monitor this database in order to check acceptance or rejection of the sug-

gested decisions and detect alarming trends. Figure 6 presents the different tables of this database.

In order to alleviate the medical support services from locating this database, an additional component is implemented in the platform, called the Service Resource Manager (SRM). This component can be used by the medical support services in order to save their state, history, calculated scores, etc. Simplified, one can think of the SRM as a front-end service for a database. The SRM provides a Web service interface for editing a database through the SRM methods *createTable*, *dropTable*, *deleteColumn*, *deletePatient*, *getAllFromTable*, *tableExists*, *getTableDescription* and *insert*. If for scalability reasons, more than one database would be needed, the SRM can transparently handle this.

3.4.10 Platform Availability

The platform has been designed to be resilient to component failures and bottlenecks by replicating components and Web services. Multiple instances of the components can exist, clustered and interacting with the system simultaneously. This platform ensures also availability in the service domains by organizing the Application Servers into pools. In order to be resilient against database failures, high-availability databases are used, replicating data from a primary source database to a standby target database, providing transparency to the platform regardless of the failure type from hardware, network, or software issues.

4. Evaluation Results

As already stated the components of the described ICSP platform, as well as some prototype medical support services, have been implemented and are currently being evaluated by the Department of Intensive Care of the Ghent University Hospital.

4.1 Evaluation Setup

The evaluation setup is shown in Figure 7. The platform components (APP) were

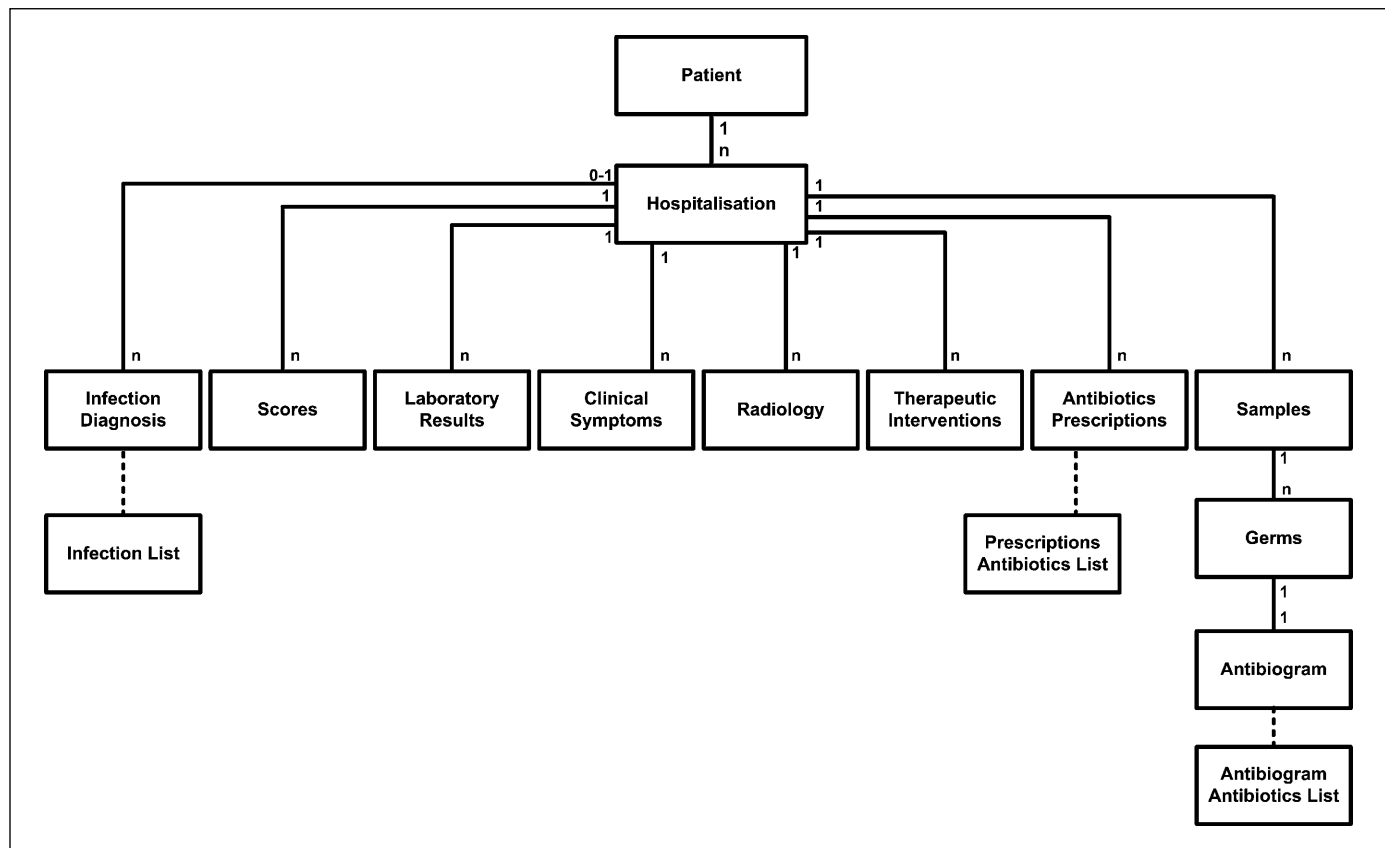


Fig. 6 ICSP database structure

deployed on a dedicated computer (INTEC_APPSERVER), while the medical support services (MSS) also run on a dedicated computer (INTEC_MSSSERVER). Both computers have an AMD Athlon 3000+ processor, 1GB RAM and Microsoft Windows 2003 Server installed. High-availability databases are used within IZ_DBSERVER, replicating data from a primary source database to a standby target database, providing transparency to the platform regardless of the failure type from hardware, network, or software issues. This database is then partially replicated to a dedicated computer for the ICSP platform, called the INTEC_DBSERVER. This way ICSP can only access anonymous patient data, exclusively containing the data from IZ_DBSERVER needed for the medical support services in ICSP. A firewall is strictly configured in between to limit communication to specific ports. The computers are connected via a fast Ethernet LAN.

The messages processed by the platform are delivered to the patient's bedside terminals via Ethernet LAN, to the physician's PDAs via WiFi 802.11b and to the smart phones or mailboxes through an SMTP-server.

4.2 Results Description

The overhead of both channel security and message security has been analyzed through a performance study for .NET Web services using SSL as transport layer security and the Web Services Enhancements (WSE) 2.0 filters and classes for Web service security (WSS) at the message layer. The major test results are presented in Figure 8. For more security test results, the reader is referred to [17].

As can be seen in this figure, secure Web services perform better using transport layer security (SSL), instead of message level security (WSS). SSL is also less in-

fluenced by the increasing amount of data than WSS. This is logical since SSL uses symmetric encryption and WSS the more calculation-intensive asymmetric encryption. Note that within SSL, the key needed for symmetric encryption is exchanged by using asymmetric encryption. Comparing default WSS encryption and partial WSS encryption, the figure shows that partial encryption generates less overhead for increasing data size since only one parameter needs to be encrypted instead of the total amount of data.

Although less efficient, WSS has its own advantages such as independence of the transport layer, the possibility to secure data over multiple SOAP hops and the protection against repudiation. Therefore, as stated in Section 3.3.5, a trade-off between performance (SSL) and security functionality (WSS) is made in order to secure the subscription platform to the best.

In order to illustrate the platform operation, two prototype medical support ser-

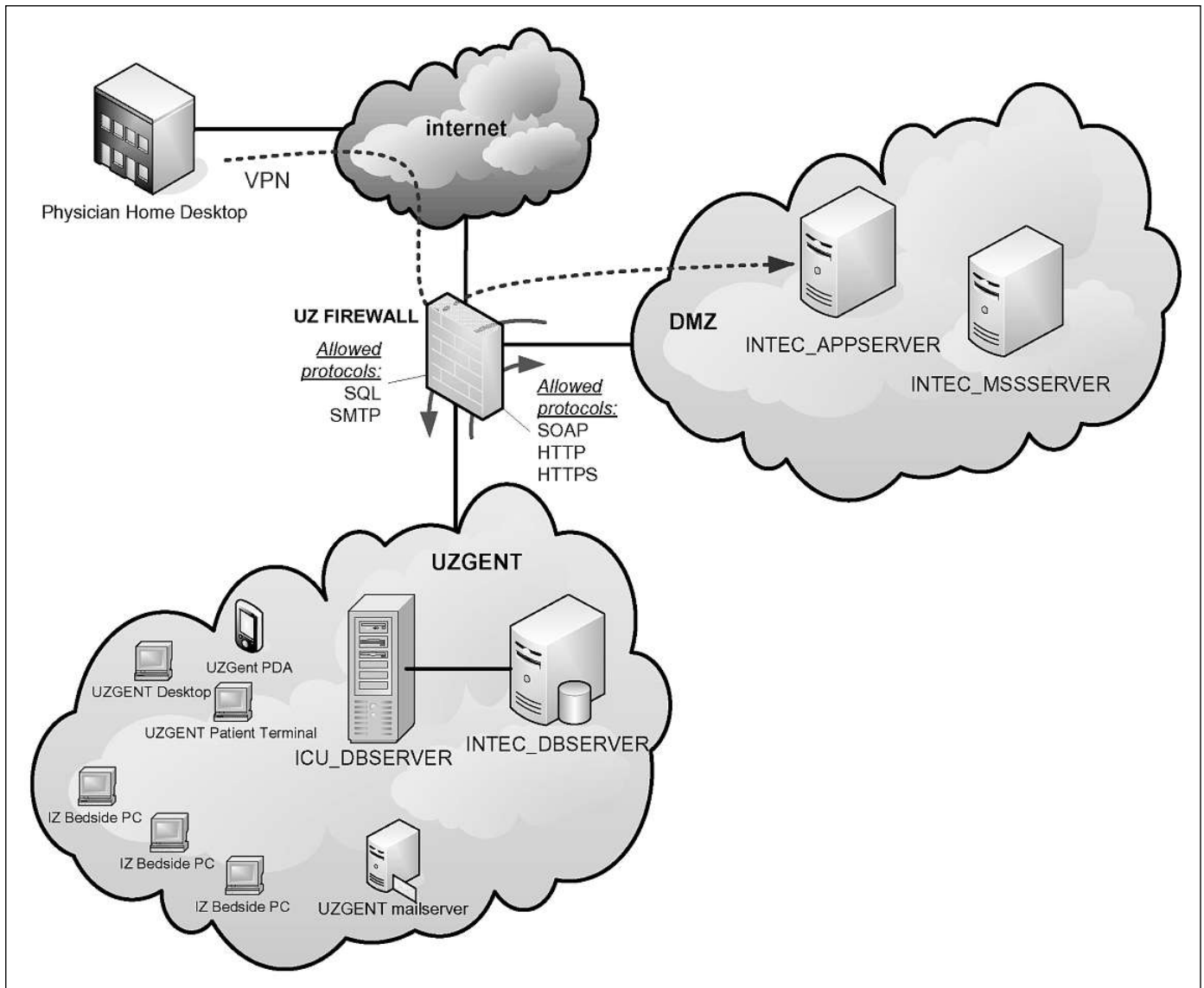


Fig. 7 Overview of ICU deployment scenario for obtaining the evaluation results

vices have been implemented that are activated at regular times by a cron trigger. First, the SOFA medical service calculates the Sequential Organ Failure Assessment (SOFA) score for all patients. This score is used to determine organ dysfunction and organ failure of critically ill patients and is calculated daily for all patients staying at ICU. The second service is a multi-service for prescribing antibiotics and consists of an ABSwitch service identifying patients at the ICU who are candidate for an early switch from intravenous antibiotic therapy to oral antibiotic therapy [18], an AB24 service mailing an overview of which antibiotics are started

and stopped during the day, an AB7/14 service checking which antibiotics are given to a patient more than 7, respectively 14, days in order to prevent resistance and finally an ABDose service giving an advice when the dosage of the recorded medication should be adjusted. Each of these services are scheduled every day and activated based on a time-based trigger.

In order to illustrate the platform performance, two prototype medical support services are presented in the next section where results should be delivered instantly to the physicians whenever new medical data is available.

4.3 Prototype Medical Support Services: RIFLE and GLYC

A group of experts proposed the RIFLE criteria to acute kidney dysfunction. RIFLE stands for the increasing severity classes Risk, Injury and Failure, and the two outcome classes Loss and End-Stage Kidney Disease. The three severity grades are defined on the basis of the changes in serum creatinine or urine output where the worst of each criterion is used. The two outcome criteria, Loss and End-Stage Kidney Disease, are defined by the duration of loss of kidney function [19].

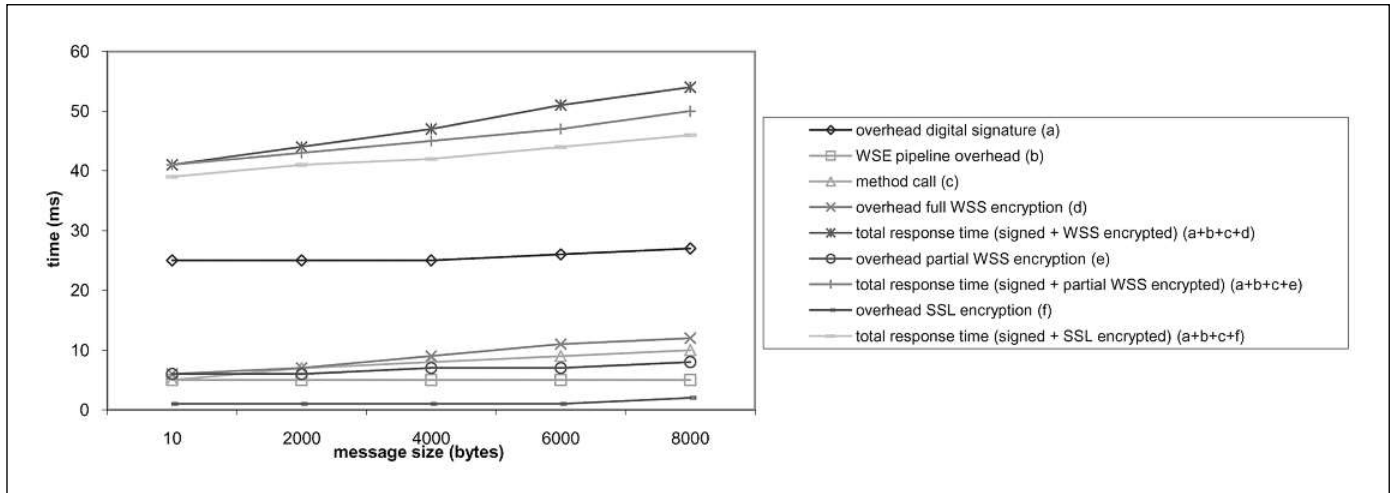


Fig. 8 Overview of response times for increasing message size and different security mechanisms

Since small changes in kidney function in hospitalized patients are important and impact the outcome, a RIFLE service was developed which calculates potential risk for kidney failure and uses the severity grading criteria Risk, Injury and Failure as presented in Figure 9. The medical support service calculates the baseline creatinine level, calculates the proportional failure of the kidney function, and generates an alarm

when patients have an increased failure of the kidney function.

Since optimal glyceic control in ICU patients decreases morbidity and mortality [20, 21], another medical support service, the GLYC service, gives an overview of the glyceic values of a patient, and makes suggestions about the insulin infusion rate in order to keep the glyceic between 80 and 140 mg/dl. Acceptance or rejection of the

suggested decision can easily be monitored by checking whether the given insulin infusion rate corresponds to the suggested decision. The GLYC service also provides graphical feedback on the efficiency of the glyceic regulation.

Figure 10 presents the time spent in the platform in between activating the trigger and presenting the result set in the client for both the GLYC and RIFLE service. The time is an average of hundred measurements and includes transforming the trigger, processing the trigger, selecting the appropriate medical support service and delivering the result set to the platform.

The figure shows that the platform processing time between firing the trigger and viewing the result set or medical suggestion, is around 135 ms, with a standard deviation of 7 ms, averaged over 100 values. The SM and SMC take negligible time since these components are implemented as session bean Web services calling an entity bean for service endpoint discovery. Since the service endpoints are fairly static and easily indexable data that is read far more often than being written, the SM and SMC read these service endpoints from an in-memory object graph via a session bean. This is much faster than continually reloading it from a database with JDBC, and results in negligible execution times.

As can be seen from Figure 10, 23% of the time is spent in the Trigger Forwarder for transforming the trigger into the internal

RIFLE-protocol	
Potential Risk ?	UO < 0,5 mL/kg/h x 2 h
Risk	SCreat increased > x 1,5 or increase > 0,3 mg/dL UO < 0,5 mL/kg/h x 6 h
Injury	SCreat increased > x 2 UO < 0,5 mL/kg/h x 12 h
Failure	SCreat increased > x 3 or (absolute > 4 mg/dL and increase > 0.5 mg/dL) UO < 0,3 mL/kg/h x 24 h or anuria x 12 h

Fig. 9 RIFLE criteria, using serum creatinine (SCreat) and urine output (UO) as variables

generic XML format, 25% in the Communication Component which requires database accesses to queue the messages to be sent, and 51% in the Subscription Component. It is not surprising that the Subscription Component is the heaviest loaded component, since it manages all subscriptions in an internal database. The Subscription Component is used twice, once to find out where to deliver the trigger data, and which medical support services are processing data, and once to find out the devices to deliver the service results to.

Therefore multiple instances of the components can exist, clustered and interacting with the system simultaneously for ensuring reliability and better performance.

On average, the time spent in the GLYC service is 1750 ms and 2401 ms in the RIFLE service, with a standard deviation of 20 ms and 126 ms respectively. The higher standard deviation for the RIFLE service can be explained by the fact that the RIFLE service follows, depending on the trigger, a different path in the algorithm resulting in more or less database transactions. The service also takes into account the RIFLE history of a patient, which influences the standard deviation as well. It is hereby shown that, since the time spent in the platform components is less than 150 ms, this is small enough to be negligible compared to the execution time of the ICU medical support services.

4.4 Platform Scalability Evaluation

The performance estimates for ICSP were obtained from measurements on the prototype implementation in the Department of Intensive Care of the Ghent University Hospital, where all the platform components were deployed on a single, dedicated computer, while the medical support services also run on a dedicated computer.

Figure 11 shows the total response time for an increasing handled trigger rate. The response time shown in the figure is defined as the time that passes between firing the trigger and presenting the results. A service, returning a single data value, is used with a negligible execution time in order not to hide inaccurate platform performance by

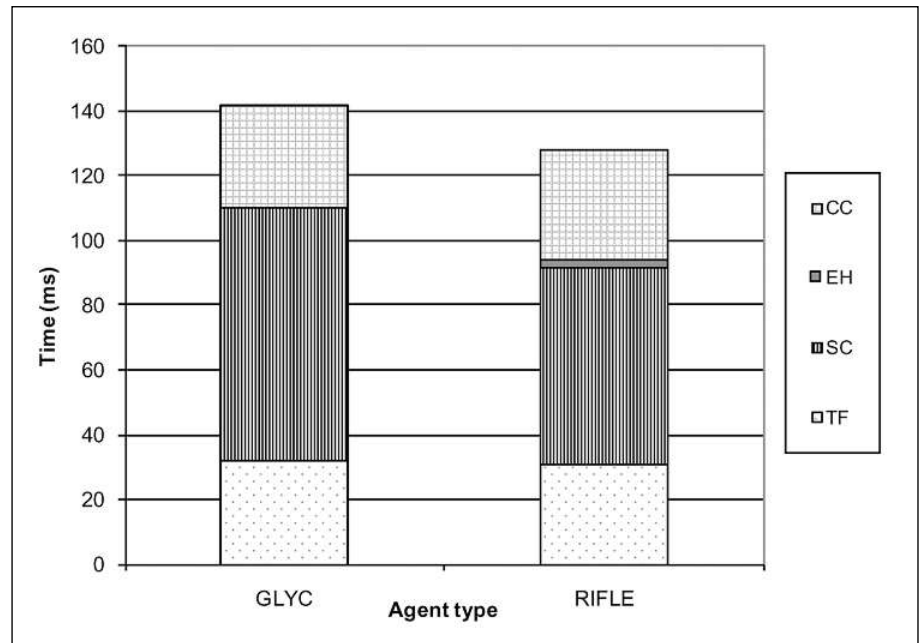


Fig. 10 Average time spent in ICSP platform components. Times of the following components are shown: Communication Component (CC), Event Handler (EH), Subscription Component (SC) and Trigger Forwarder (TF).

large service execution times. This way the response time incurred by the ICSP platform is presented for an increasing trigger rate.

As can be seen on the figure, the ICSP platform can handle 23.75 calls per second

on the current single-server installation. In the current platform implementation at Ghent University Hospital, the RIFLE service is running for 56 patients and generates around 605 triggers per day, resulting in an average of about 11 triggers per patient per

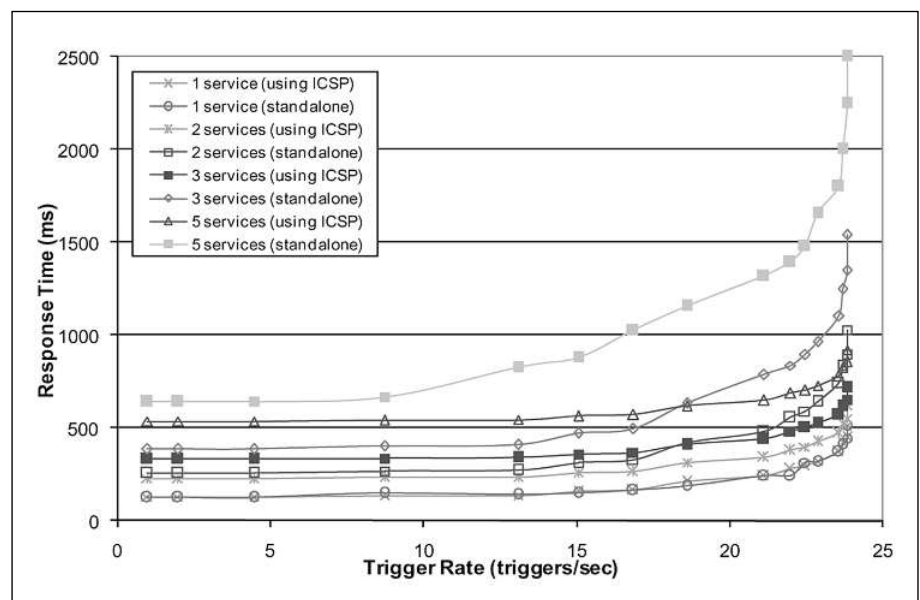


Fig. 11 Response time spent in the ICSP components in function of the throughput, i.e. number of handled triggers per second. As can be seen, ICSP can handle 23.75 triggers per second on the current single server installation.

day, ranging from patients with 4 triggers per day to patients with 27 triggers per day. Other medical support services have comparable trigger frequency or higher.

The ICU of Ghent University is the second largest ICU in Belgium and currently holds 56 beds. In September 2007 they will migrate 16 cardiac intensive care beds to the platform, bringing the total to 74 beds. Next year, they will also integrate Midcare, an additional 20 beds, resulting in a total of 94 beds in ICU. If a trigger per patient is thrown every 15 minutes, the current platform implementation scales up to more than 200 medical support services for 94 patients.

Since every patient generates around 16,000 different data values on a daily base, each of these data values can be configured as a trigger in the future when more medical support services are deployed in the platform, resulting in at most 16,000 triggers per patient per day. Data from devices that are continuously monitoring particular data values can also be used as a trigger. Since not all of these data will be interesting triggers, one can choose to only trigger the validated data or data wherefore the difference between the last trigger and current value is more than the configured bounds, instead of letting every single monitoring datum be a trigger. Figure 11 shows that the throughput increase is no longer linear for values of $x > 13$. Considering this transition point and thus allowing a trigger rate of 13 triggers per second, our single-server installation scales up to 70 patients, each generating at most 16,000 data values on a daily base. In order to be able to process the triggers for 94 patients, the platform components and Web services are replicated over two server installations. These two instances of the components are then clustered and interact simultaneously with the system in order to optimize throughput and performance. The manageable load on the HIS replication databases is independent from the ICSP and depends on the HIS database type and server it is running on. The ICSP platform scalability design allows however these triggers to be run on multiple replications of the HIS databases in order to divide the load if necessary.

Since medical parameters are used by multiple medical support services, one

trigger generally results in activating more than one service. As can be seen in the figure, the platform introduces benefits in response time if at least two medical support services are activated by a trigger.

5. Discussion

ICU medicine is expensive and there is a shortage of intensivists. In the USA, the different societies working in the critical care medicine field recently published recommendations about the future organization of ICU medicine [22-24]. They mention the introduction of computerized decision support as one of the most important recommendations to improve quality of patient care for the near future. However, currently only 10% to 15% of intensive care units are computerized [25]. The Intensive Care Unit of the Ghent University has started computerizing their department in 2003 in order to result in a paperless ICU by capturing and storing all data from monitors, ventilators and pumps in databases.

In our opinion, service-based software architectures are the best choice for implementing decision support systems in the ICU.

As stated before, it is expected that hundreds of medical support services will be active simultaneously in order to optimize the care of critically ill patients. The ICSP is fundamental for medical support service deployment since it simplifies medical support service development by taking over the general functionalities. This way, services only need to implement medical decision algorithms processing the received medical data from laboratories or monitors. The generated medical support message will be delivered to the patient's bedside terminal, or the treating physician's PDA, smart phone or mailbox.

Delays in treatment actions, especially in critical care medicine, can have important negative impacts on outcome. By using our platform it is possible to move from discovery of information to anticipation through delivering support and alarm messages.

The advanced features for the Intensive Care Subscription Platform were mentioned

in Section 2. These functional requirements were addressed in the design process in the following way:

Since physicians, and more specifically intensivists, do not have a lot of time for training, the user interface is very intuitive and uses medical abbreviations and well-known service names. Designing the interface is done in participation with the physicians in order to present the personalized support and alarm messages in an intuitive way.

By ensuring that the timeliest information is always delivered where and when it is needed by dynamically selecting the best service, the intensivists can focus on patient care. Solutions are also provided for mobile devices so that the platform and messages are conveniently accessible by the health-care professionals at the bedside or other point of care when needed. The generated alerts are easy to interpret and limited in amount in order to improve acceptance rate. High-priority messages are delivered directly to the physician's smart phone or PDA as well as the patient's bedside terminal, e-mail is used for sending reports for medical decision support and overviews of historical medical results are presented at the patient's bedside terminal.

Besides the functional requirements, Section 2 also mentioned the main requirements for the Intensive Care Subscription Platform. These requirements were addressed in the design process in the following way:

Single point-of-failures are avoided in the platform in order to ensure reliability. Since the system has to work correctly at all times and no messages should get lost, the platform is decomposed into multiple functional components that can be coupled dynamically and uses WS-ReliableMessaging to deliver the medical suggestions, results or alerts to the subscribed users. Dynamically replacing one component with another one, in case of failure or overloading, ensures the required platform reliability. By using WS-ReliableMessaging, guaranteed message delivery of the medical suggestions, results or alarms is ensured. If a message cannot be delivered, the message is queued until the receiver comes online. For high-priority messages, alert escalation is provided.

Since hospital healthcare involves very high level of diversity in hardware, software, and other elements, the interoperability of hospital information systems poses some difficult problems. In order to cope with this heterogeneity, Web services and service-oriented architectures provide the required easy integration with hospital information systems.

By making use of the Web service technology, generic definition of the interface and the exchanged messages, the platform is largely generic, i.e. as independent as possible of i) the implementation language of the services, ii) the operating system and hardware of the workstations, and iii) the particular type of the services. New Web service components can be added independent of implementation languages, operating systems and hardware.

Consequently, our platform is interoperable and interconnected, uses open standards and has a highly scalable distributed model to deliver information to physicians.

Since security is a very important aspect of an ICU platform, a secure communication environment and a secure execution environment are provided. A secure communication environment means that the information sent between the medical support services and users cannot be intercepted or even altered by malicious users. This has been taken care of by applying encryption techniques to data that is transported over the network. A secure execution environment on the other hand is a secure runtime environment designed to protect against deviant applications. This is easy to obtain when none of the involved servers has an Internet connection, by restricting login access to the application servers to the authorized personnel.

And finally, the platform performance has been evaluated and it was shown that the extra response time imposed by the platform is small enough to be negligible compared to the execution time of the ICU medical support services. By means of the described division of application servers in pools, the platform allows for the division of the whole set of servers in an ICU department in several sub-clusters, each with their own access rights and set of target applications.

We are currently developing auxiliary management services to monitor the ICSP database, containing all derived abstractions, calculated scores by the medical support services, and service states or history overviews of measurements, completed with additional physiological parameters, laboratory parameters, microbial results, antibiotic use and ICU scores, in order to check acceptance or rejection of the suggested decisions and detect alarming trends. These management services can check the acceptance of the GLYC insulin rate, check if the antibiotic switches are done as suggested in order to decrease the cost, check if the antibiotic dosage is followed so that the given antibiotics would not remain in the body and have toxic effects when the renal function is low, or even detect alarming trends for example in antibiotic prescription behavior. Future work also includes the development of a graphical tool for automatically generating services. This way, physicians will be able to design medical support services themselves in a simple way and add these services to the framework. An important advantage of the service-oriented approach allows for extending the functionality of components through emerging Web service standards. This way, the Event Handler functionality will be extended for dynamic selection and composition of services through standards on one hand for flow specification, such as WS-BPEL [26], and on the other hand for semantic markup, such as OWL-S [27], resulting in the deployment of multiple advanced medical support services.

6. Conclusion

It is generally expected that within the near future, intensive care unit computerization including advanced real time and bed-side decision-making capabilities, will become essential to guarantee the highest quality of care for every ICU patient. In this paper, we motivated the need for an ICU management and subscription platform allowing for the subscription of the medical decision support data and offering advanced features such as user-friendly patient/service subscription,

transparent data retrieving and migration, medical support service selection, delivery and priority-based filtering of decision support messages and service resource management.

The architecture has been implemented and the resulting platform is referred to as the Intensive Care Subscription Platform (ICSP). The platform is currently being evaluated in the real-life setting of the Department of Intensive Care of the Ghent University Hospital, showing that the extra response time introduced by the platform is less than 150 ms and negligible compared to the execution time of the ICU medical support services.

Acknowledgments

Sofie Van Hoecke would like to thank the IWT (Institute for the Promotion of Innovation through Science and Technology in Flanders) for financial support through her Ph.D. grant.

Filip De Turck acknowledges the F.W.O.-V. (Fund for Scientific Research-Flanders) for their support through a postdoctoral fellowship.

References

1. Morris AH, Gardner RM. Principles of Critical Care, chapter on Computer applications. In: Hall J, Schmidt G, Wood L (eds). New York: Mc Graw-Hill; 1992. pp 500-514.
2. Weed LL. Clinical Judgment Revisited. *Methods Inf Med* 1999; 38 (4): 279-286.
3. Hoste EA, Clermont G, Kersten A, Venkataraman R, Angus DC, De Bacquer D, Kellum JA. RIFLE criteria for acute kidney injury are associated with hospital mortality in critically ill patients: a cohort analysis. *Critical Care* 2006; 10: R73.
4. Georgieva M, Milanov S, Milanov M, Gyurov E. Clinical pulmonary infection score (CPIS) dynamics in polytrauma patients with ventilator-associated pneumonia. *Critical Care* 2004; 8 (Suppl 1): p 212.
5. Kajdacsy-Balla Amara AC, Andrade FM, Moreno R, Artigas A, Cantraine F, Vincent J. Use of the Sequential Organ Failure Assessment score as a severity score. *Intensive Care Med* 2005; 31: 243-249.
6. Decruyenaere J, De Turck F, Vanhastel S, Vandermeulen F, Demeester P, De Moor G. On the Design of a Generic and Scalable Multilayer Software Architecture for Data Flow Management in the Intensive Care Unit. *Methods Inf Med* 2003; 42: 79-88.
7. De Turck F, Decruyenaere J, Thysebaert P, Van Hoecke S, Volckaert B, Danneels C, Colpaert K, De Moor G. Design of a flexible platform for execution of medical decision support agents in the Intensive Care Unit. *Elsevier Journal of Com-*

- puters in Biology and Medicine 2007; 37 (1): 97-112.
8. Lenz R, Kuhn KA. Intranet Meets Hospital Information Systems: The Solution to the Integration Problem? *Methods Inf Med* 2001; 40 (2): 99-105.
 9. Fieschi M, Dufour JC, Staccini P, Gourvernet J, Bouhaddou O. Medical Decision Support Systems: Old Dilemmas and new Paradigms? *Methods Inf Med* 2003; 42: 190-198.
 10. Greenes RA. *Clinical Decision Support. The Road Ahead.* Elsevier Inc., 2007.
 11. Chen HT, Ma WC, Liou DM. Design and Implementation of a Real-Time Clinical Alerting System for Intensive Care Unit. *Proc AMIA Symp* 2002. pp 131-135.
 12. Wakai K, Kawamura T, Endoh M, Kojima M, Tomino Y, Tamakoshi A, Ohno Y, Inaba Y, Sakai H. A scoring system to predict renal outcome in IgA nephropathy: from a nationwide prospective study. *Nephrol Dial Transplant* 2006; 21 (10): 2800-2808.
 13. Adrie C, Cariou A, Mourvillier B, Laurent I, Dabane H, Hantala F, Rhaoui A, Thuong M, Monchi M. Predicting survival with good neurological recovery at hospital admission after successful resuscitation of out-of-hospital cardiac arrest : the OHCA score. *Eur Heart J* 2006; 27 (23): 2480-2485.
 14. Chen J, Chung J, Wong T, Fan KL, Pun CO. Early detection of pulmonary hypertension with heart sounds analysis pilot study. *Stud Health Technol Inform* 2006; 122: 112-116.
 15. Trivedi MH, Kern JK, Marcee A, Grannemann B, Kleiber B, Bettinger T, Altshuler KZ, McClelland A. Development and Implementation of Computerized Clinical Guidelines: Barriers and Solutions. *Methods Inf Med* 2002; 41 (5): 435-442.
 16. The World Wide Web Consortium (W3C). <http://www.w3c.org>
 17. Van Hoecke S, Haerick W, De Jans G, De Turck F, Laermans E, Dhoedt B, Demeester P. Design and Implementation of a Secure Media Content Delivery Broker Architecture, The 2005 International Symposium on Web Services and Applications (ISWS'05). Las Vegas, USA; 2005.
 18. Steurbaut K, Van Hoecke S, Colpaert K, Danneels C, Decruyenaere J, De Turck F. Granularity of Medical Software Agents in ICU – Trade-off Performance versus Flexibility. *Proceedings of the First International Symposium on Intelligent and Distributed Computing (IDC'07)*. Craiova, Romania; 2007.
 19. Hoste EA, Kellum J. Acute Kidney Injury: Epidemiology and Diagnostic Criteria. *Current Opinion in Critical Care* 2006; 12 (6): 531-537.
 20. Van den Berghe G, Wouters P, Weekers F, Verwaest C, Bruyninckx F, Schetz M, Vlasselaers D, Ferdinande P, Lauwers P, Bouillon R. Intensive insulin therapy in the critically ill patients. *N Engl J Med* 2001; 345 (19): 1359-1367.
 21. Rady MY, Johnson DJ, Patel BM, Larson JS, Helmers RA. Influence of individual characteristics on outcome of glycemic control in intensive care unit patients with or without diabetes mellitus. *Mayo Clin Proc* 2005; 80 (12): 1558-1567.
 22. Ewart GW, Marcus L, Gaba MM, Bradner RH, Medina JL, Chandler EB. The critical care medicine crisis: a call for federal action: a white paper from the critical care professional societies. *Chest* 2004; 125 (4): 1518-1521.
 23. Kelley MA, Angus D, Chalfin DB, Crandall ED, Ingbar D, Johanson W, Medina J, Sessler CN, Vender JS. The critical care crisis in the United States: a report from the profession. *Chest* 2004; 125 (4): 1514-1517.
 24. Irwin RS, Marcus L, Lever A. The critical care professional societies address the critical care crisis in the United States. *Chest* 2004; 125 (4): 1512-1513.
 25. Levy M. Computers in the intensive care unit. *Journal of Critical Care* 2004; 19 (4): 199-201.
 26. Andrews T, Cubera F, Dolakia H, Golland J, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, Trickovic I, Weeravarana S. *Business Process Execution Language for Web Services*, 2003.
 27. The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical White paper (OWL-S version 1.1), 2004.

Correspondence to:

Sofie Van Hoecke, senior PhD student
Ghent University
Department of Information Technology
Gaston Crommenlaan 8 bus 201
9050 Gent
Belgium
E-mail: sofie.vanhoecke@intec.ugent.be