An agent-based platform for efficient telemonitoring data processing

Sofie Van Hoecke, Kristof Taveirne, Koen De Proft, Filip De Turck, Johan Decruyenaere*, Bart Dhoedt

Abstract— **Telemonitoring devices allow healthcare providers to remotely monitor patients in their homes. These telemonitoring devices transmit their measurement values at regular times to a storage system. However, the amount of data in these storage systems and the heterogeneity imposes special attention for telemonitoring data processing. Information technology can facilitate the abstraction of relevant information and support the physician through software agents for medical data processing. Therefore, we present a platform that allows for the intelligent subscription of the medical agents' data, offering advanced features such as dynamic agent selection; user-friendly patient/agent subscription and profile based message filtering.**

Index Terms— **Telemonitoring, medical data processing, Web services, agents.**

I. Introduction

The ageing population and a shift in the burden of illness from acute (e.g. infections and injury) to chronic conditions (e.g. heart disease, asthma, epilepsy, cancer, mental health problems, etc.) drive up health costs. Information and communication technologies have considerable potential for improving delivery and quality of care and create opportunities to manage and provide care faster, at lower cost and higher levels of convenience for their patients.

Patients are being discharged from hospitals earlier. It is however sometimes difficult to permit them to go home, unless the home is fitted with the appropriate technology and offer of care. Often additional healthcare services and monitoring of their health status is required. Telemonitoring devices allow disease management firms and healthcare providers to remotely monitor patients in their homes, resulting in an improved quality of life for chronically ill and elderly patients. Ehealth applications empower providers with a broad range of tools to collaborate on a patient's condition, manage referrals and appointments, and provide consumers with more timely, accessible and convenient care. This includes patient diagnosis and follow-up, distance monitoring, and providing treatment in healthcare facilities or in homes. With tools like an Interactive Patient Terminal, or a TV-based healthcare platform, care managers can remotely monitor patients' vital signs data. Every day, patients take their own vital signs measurements as prescribed by their doctor: weight, glucose level, blood pressure, blood oxygen level, pulse and/or ECG rhythm. This information is then transmitted through an ordinary phone line or via modem to a storage system.

Clinicians, general practitioners and nurses can track daily patient measurements, store and retrieve historical data, analyze the monitoring data and generate reports. However, current systems suffer from flexibility and efficient data processing. Our view is that application of agent technology is the best way to allow for flexible and efficient processing. Software agents can support the physician through medical data processing. A software component qualifies as an agent when its behavior is autonomous (i.e. controls his internal state and takes his own decisions), flexible (i.e. adapts dynamically to changes in its environment) and proactive (may take the initiative when appropriate and perform actions not explicitly required by the user). Typical examples of such agents are an infection detection agent, an agent supporting the insulin strategy for patients with diabetics, or agents predicting the outcome of ill patients.

It is expected that in future healthcare information systems, tens of agents will be active simultaneously in order to optimize the care of patients. Consequently a management platform is required for subscription, data handling as well as for the notification and presentation of results. Therefore, the only solution is to incorporate a middleware platform coupling all actors of the medical data processing (laboratories, monitors, physicians, nurses, etc.), offering simple graphical user interfaces to physicians for subscribing to a patient/agent combination and restricting medical support messages to a minimum in order not to overload physicians with unimportant messages. In this paper, we detail the design of such a platform that allows for the intelligent subscription of the agents for medical data processing. The remainder of this paper is structured as follows: Section II describes the Web service technology, while in section III our platform for agent subscription is presented. Finally in section IV, we will highlight the main conclusions and identify some future work.

II. Web service technology

The platform has been designed and implemented based on the Web service technology. 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. By means of this technology, applications can easily be distributed and expose well-defined functionality as a Web service, which consumes and produces

Department of Information Technology
Gaston Crommenlaan 8 bus 201, B-9050 Gent
Ghent University/IBBT/IMEC, Ghent, Belgium
fax: +32 9 3314 899 - tel: + 32 9 3314 940
Email: sofie.vanhoecke@intec.ugent.be

*Department of Intensive Care
De Pintelaan 185, B-9000 Ghent, Belgium
Ghent University Hospital, Ghent, Belgium
Email: johan.decruyenaere@ugent.be

XML-messages over HTTP. Based on the exchange of structured text messages, the interaction abstracts the underlying technologies. The Web service technology enables thus the required integration for our intelligent subscription and management platform.

By using the Web service technology, our framework is designed based on the principles of service-oriented architectures, wherein all components are implemented as services. The total framework is composed out of multiple services as independent building blocks. These services are autonomous, independent of other services and only responsible for their own functionality. The services are loosely coupled through agreed interfaces and communicate using an Internet protocol like HTTP by sending XML messages using SOAP. By using the generic concepts of service-oriented computing, the framework presented in this paper isn't restricted to appliance in ehealth. Due to its generic, dynamic and scalable design, it can be used within a wide range of applications. The framework acts as a generic communication system in which agents can easily be plugged. The agent functionality is however completely free and can cover other areas than medical data processing.

III. Platform for agent subscription

Figure 2 depicts the main software components of the platform.

A. Requirements

The main requirements for the intelligent subscription platform of the medical data processing can be stated as follows:

- 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 alarming messages. We decomposed the framework into multiple functional components that can be coupled dynamically. Dynamically replacing one component with another one, in case of failure or overloading, ensures the required reliability.
- The platform should allow for easy integration
- The platform should be generic and independent of implementation languages, operating systems and hardware.
- The platform should support priority, such that high priority messages get priority over less critical messages and are presented first on the device monitor.
- The platform overhead must be small enough to be negligible compared to the execution time of the agents.
- Security is a very important aspect: the platform should provide a secure communication environment.

B. Functional description

The healthcare providers interact with the platform through a graphical front-end (see figure 1) and subscribe to their preferred agent/patient combination. An incoming trigger from laboratories or monitoring devices activates the agent selection. Based on semantic querying methods, a lookup for agents is performed using a specification of characteristics and properties. When the agents have processed the telemonitoring data, the platform checks the subscriptions in order to find out where to deliver the result message from the medical data processing. Since healthcare providers can choose patient/agent combinations, as well as the type of messages they want to receive (alarm, summary, info, etc.) and agents restrict the medical support messages to a minimum, physicians are not overloaded with unimportant messages.

C. Component description

The architecture consists of the following components:

- **Trigger Forwarder (TF):** The Trigger Forwarder is the top level component of the platform and handles the incoming triggers from laboratories or monitors. This component will receive the triggers and transform 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 agents via the Event Handler (EH).
- **Communication Component (CC):** The Communication Component handles the communication between doctors' PDAs, monitors and the platform. Through the graphical front-end of this component, care providers can manage their subscriptions. The CC also contains a database, storing all the messages so that they don't 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, and the Communication Component which agents are processing data.
- **Database Manager (DBM):** The Database Manager handles communication with external storage systems and databases, containing patient records and monitoring results. The DBM provides a single interface for access by the agents, 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 agents onto the correct queries. The DBM is also responsible for executing these queries in order to provide the requested data to the agents.
- **Event Handler (EH):** The Event Handler takes care of the communication between the agents and the rest of the framework. On one hand the EH offers the agents a single interface to communicate with the platform, on the other hand agents are abstracted this way as a single interface as well, simplifying complexity for the other platform components.
- **Agent Manager Coordinator (AMC):** The Agent Manager Coordinator is responsible for the coordination of all messages from the platform to the agents. In the Agent Manager Coordinator all agents are registered and described in detail. When a new Application Server is installed in the system, the Agent Manager of the pool

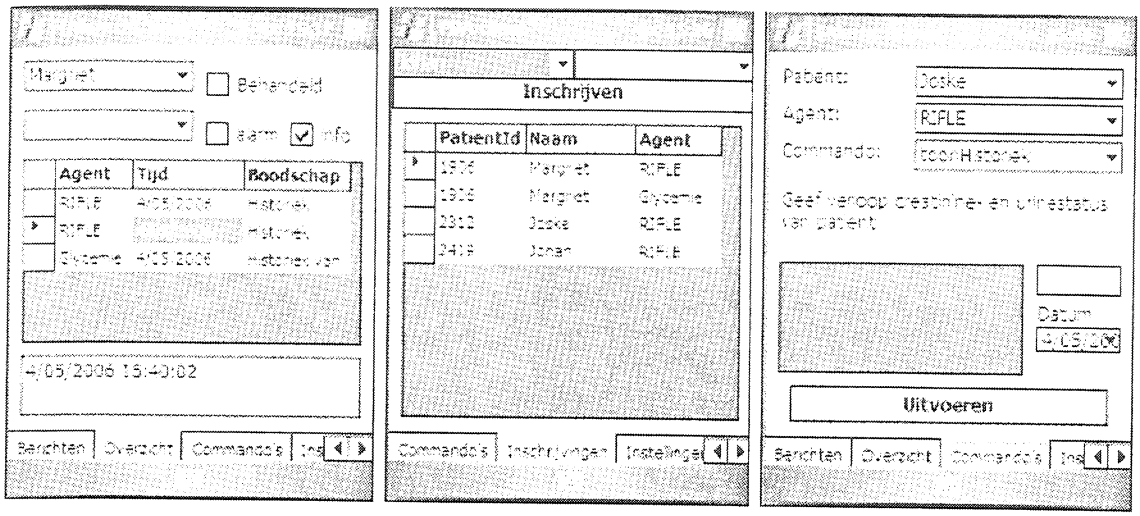


Fig. 1. Graphical front-end for managing agent/patient subscriptions

it belongs to registers that Application Server and its agents to the AMC. Agents 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 the AMC uses a combination of registration mechanisms and dynamic discovery.

Based on the load information provided by the Pool Management Server Coordinator, the AMC can select, by using advanced load balancing algorithms, the optimal Application Servers for executing the agents.

- **Agent Manager (AM):** The Agent Manager manages a pool of Application Servers. When a new Application Server (AS) is started, it has to register itself to the AM of that pool. The Agent Manager then registers that Application Server and its agents to the Agent Manager Coordinator. When new agents have to be started on specific Application Servers, the Agent Manager Coordinator contacts the Agent Manager of the pools these ASs belong to. The AM then forwards the start commands to the correct AS. Result sets from these Application Servers however are not sent back to the Agent Manager, but directly forwarded to the Event Handler.
- **Application Server (AS):** Application Servers execute the actual agents. Each Application Server can implement one or more agents, can have its own capabilities and can provide agents using different programming languages. When a new AS is installed in the system, it must register itself to the Agent 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 Agent Manager Coordinator. When selecting the required agents, the Agent 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 of 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 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.

D. Dynamic selection of agents

In order to process the telemonitoring data, the AMC component implements advanced load balancing algorithms for dynamically selecting agents, based on the functionality of the agents and the load of the Application Servers. Different load balancing algorithms can be used ranging from a simple random, round robin or least connections algorithm to more sophisticated algorithms such as weighted load balancing or minimum load. The ASM currently implements the minimum load algorithm for load balancing since this outperforms the other load balancing algorithms [1].

An OWL-S ontology is used for creating a hierarchical description of the agents, as well as relationships between the agents. Supplied with the semantic Web methods, the service interfaces and corresponding server load information, the AMC can select the best agents for processing the telemonitoring data. Using OLW-S for querying the agents, more in-

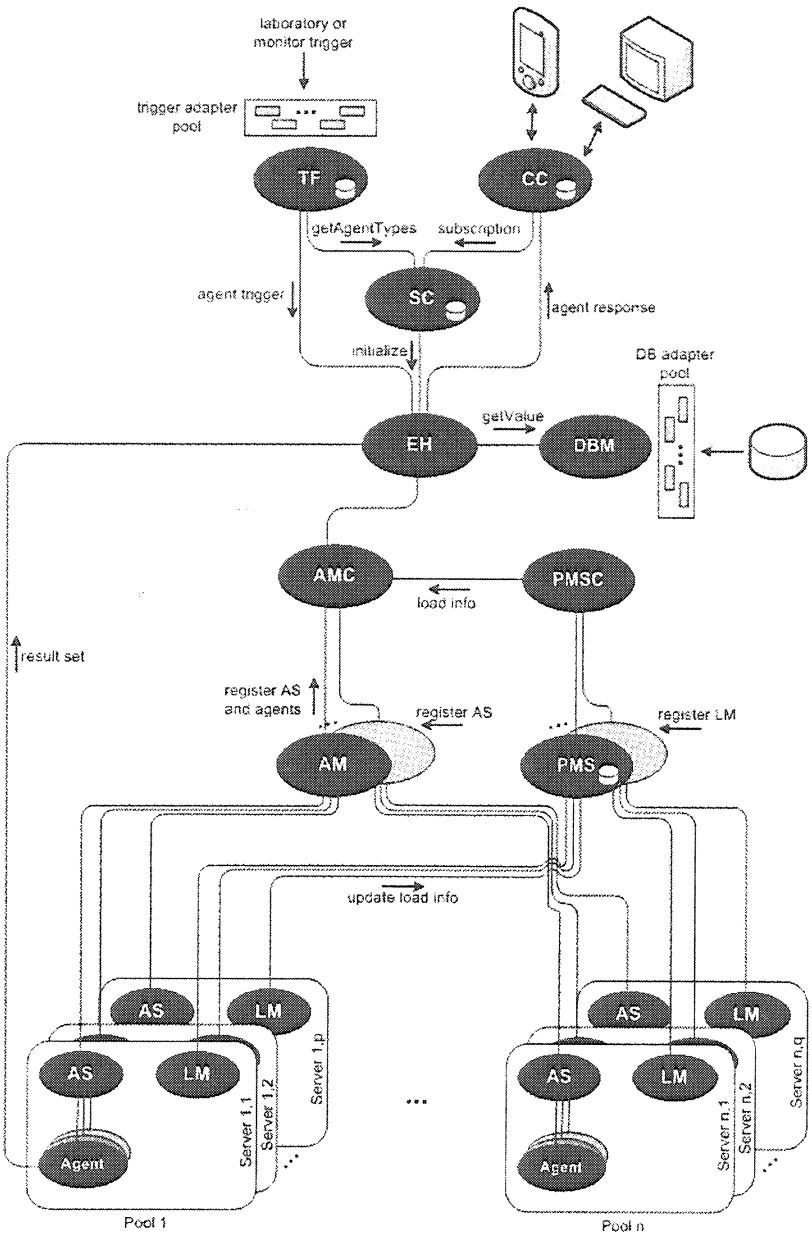


Fig. 2. Platform for agent subscription. The following components are shown: Trigger Forwarder (TF), Communication Component (CC), Subscription Component (SC), Database Manager (DBM), Event Handler (EH), Agent Manager Coordinator (AMC), Agent Manager (AM), Application Server (AS), Pool Management Server (PMS), Pool Management Server Coordinator (PMSC) and Load Monitor (LM).

telligent service selection mechanisms are possible. Currently most registries use a simple name to reference resolution. By using the semantic querying methods, a lookup for agents can be performed using a specification of characteristics and properties. The reasoner behind the registry-ontology can then infer the correct, or the most appropriate reference towards an agent. Using reasoning performed on ontologies, composition of several agents into a larger virtual agent is supported and discovery based on particular characteristics is possible. When one of the selected agents suddenly slows down or becomes unavailable, the AMC chooses an alternative agent. This selection process is repeated one or more times until the telemonitoring data is processed.

E. Security

As investigated in [2], secure Web services perform better using SSL, thus transport layer security, instead of WSS message level 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.

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

Although result sets from the agents are not sent back to the Agent Manager and Agent Manager Coordinator, but directly forwarded to the Event Handler, eliminating redundant intermediaries, result sets still cross intermediary components before reaching their destination, likewise for triggers and incoming agent requests who also cross intermediary components. These intermediary components however 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, we assure that the result sets can not be intercepted or altered by malicious users, independent of the transport layer to be secure or not. Only the sender (i.e. agent) and the ultimate receiver (e.g. doctor's PDA or monitor) will be able to read the complete message content.

F. Scalability

In the middleware platform scalability is ensured by avoiding single point-of-failures and bottlenecks by replicating components. Multiple instances of the components can exist, clustered and interacting with the system simultaneously. This architecture ensures also scalability in the agent provider domains by organizing the Application Servers into pools.

ACKNOWLEDGMENT

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

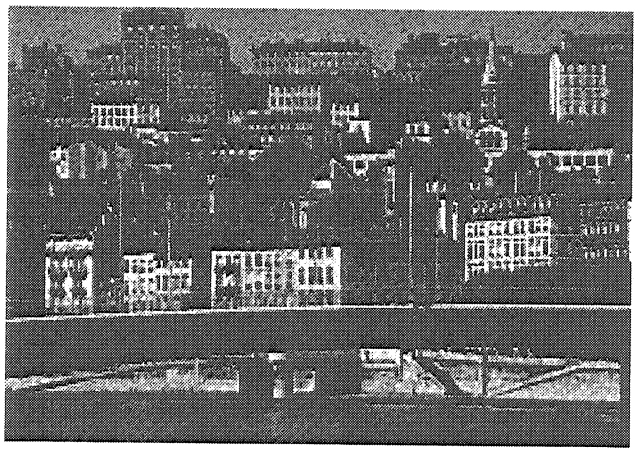
REFERENCES

[1] F. De Turck, J. Decruyenaere, P. Thysebaert, S. Van Hoecke, B. Volckaert, C. Danneels, K. Colpaert, G. De Moor, Design of a flexible platform for execution of medical decision support agents in the Intensive Care Unit, Elsevier Journal of Computers in Biology and Medicine.

[2] S. Van Hoecke, W. Haerick, G. De Jans, F. De Turck, E. Laermans, B. Dhoedt, P. Demeester, 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.

Sofie Van Hoecke received her M.Sc. degree from the Engineering Department of Ghent University in 2003. She is now a research assistant and Ph.D. student affiliated with the Department of Information Technology of the same university, and has received a scholarship by the IWT (Institute for Innovation in Science and Technology - Flanders). Her research interests are the design and performance modeling of Web service-based architectures for multimedia content delivery and ehealth. She especially addresses the QoS-brokering and composition of novel services.

Health Pervasive Systems
HPS06



June 29, 2006
Lyon, France

In conjunction with

ICPS **IEEE International Conference on Pervasive Services**

IEEE IEEE COMPUTER SOCIETY

THE UNIVERSITY OF ARIZONA. TUCSON ARIZONA

The papers in this book comprise the digest of the meeting mentioned on the cover and title page. They reflect the authors' opinions and are published as presented and without change, in the interest of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the Institute of Electrical and Electronics Engineers, Inc.

IEEE Catalog Number: 06EX1325

ISBN: 1-4244-0237-9

Library of Congress: 2006921095