AN ACTIVE APPROACH IN HEALTH MONITORING

Doan B. Hoang[1], Nor Faizah Ahmad[1] and Congduc Pham[2]
[1]*Centre for Innovation in IT services and Applications(iNEXT), University of Technology, Sydney, Australia*
[2]*LIUPPA, University of Pau, 64012, Pau cedex, France*

Keywords: Health monitoring, Active monitoring architecture, Active database, Active care application.

Abstract: Specific-purpose hardware or software modules are often designed to provide timely response to conditions or events that required attention. However, these application-specific mechanisms are not flexible and often can not be reused or adapted to different situations and applications. This paper proposes an approach whereby an active database with triggering mechanisms is deployed for surveillance and health monitoring applications. The paper presents a comprehensive monitoring architecture and an application to demonstrate its use in an Active Maternity Care system.

1 INTRODUCTION

We are surrounded by numerous events whereby our response might or might not be expected promptly. For example, an alarm has to be raised immediately if an intruder or a critical health condition is detected. Many hardware and software solutions are readily available to deal with these situations. First, a simple electronic circuit can be designed to deal with a particular condition. However, the device can not easily be modified nor reused in other different conditions. Secondly, using a software approach, an algorithm can be designed to handle the event where it is more flexible in dealing with dynamic circumstances and conditions as they can be automated and programmed with little effort. However, most software solutions suffer from a time-consuming and constant polling loop for monitoring the event of interest. They also introduce delays and therefore can not always deal with the event in a timely manner. In current practices, database management system (DBMS) is deployed to handle the events/data records systematically. The DBMS is passive as its only responds to queries or transactions explicitly submitted by users or application programs. When monitoring events in passive DBMS, an application program polls the database by sending queries about the monitored data (Paton, 1998), (Kotz-Dittrich, 1998). For instance, in health monitoring systems, a server agent is developed to monitor data inside the database. The server agent needs to constantly poll to the database to check the value of the health parameter that is being monitored. The agent then waits for the database to return the requested value. If the returned value is less than a preset threshold, the agent may trigger an alarm or send a notification. The problem with this approach is that if the frequency of polling is too high, resources for handling polling are wasted without gaining new information as the database will most likely return the same previous value. In addition, the database system will be overflowed by queries. However, if the frequency of polling is too low, changes of data in the database may not be detected in time. In a health care environment, time is very critical and it is not possible to predict the exact time of complication occurrence. The ideal solution is that once the complication occurs, the notification task should commence automatically. This is where active mechanisms are needed.

This paper presents an approach whereby an active database is deployed at the core of the solution. An active database provides mechanisms for triggering an action if a certain event occurs within a set of conditions that affect specific elements of a data record within the database. A conventional relational database is used to monitor data. The active database inherits all the desirable properties of a well-designed database including data consistency and integrity. Active component is designed to deal with triggering events. In particular we propose a complete active database management system (ADMBS) architecture that supports the implementation of a class of health monitoring and surveillance applications. The active database monitoring architecture consists of four main modules: the event validation module, the active engine module, the decision making module and the interface module for user interaction.

The rest of the paper is organised as follows. Section 2 discusses related work on active database research and applications. Section 3 presents our proposed active monitoring architecture and its components. Section 4 describes an application that we developed in the assistive healthcare and the implementation of the active monitoring architecture. Section 5 concludes the paper with discussion on future research.

2 RELATED WORK

An AI approach with multi-agent systems is often used to deal with health monitoring in several domains. *Health information access*. Multi-agent is used in providing ubiquitous health information access of the patients. Using the system, the patients' record can be accessed and viewed by clinicians at different towns (Moreno, 2002), (Koutkias, 2005). *Patient scheduling*. Due to the complexity of the medical appointment procedure including different types of diagnostics and treatments, a multi-agent approach is proposed to schedule the patients' appointment (Alamillo, 2003). *Internal hospital tasks*. A multi-agent framework is proposed for managing the patient care in the hospital, which involves the transaction of workflow processes within health departments. This includes monitoring applications of medical protocols (Alamillo, 2003) or controlling the usage of drugs taken by the patients (Godo, 2003). In general, multi-agent solutions consist of specific-purpose agents such as server agent, provider agent, information agent, client agent, etc. These systems are usually implemented with passive DBMS. In this paper, we focus on the active database approach.

Active DBMS (ADBMS) extends passive DBMS's functionality with active rules (Dayal, 1995). Active rules allow ADBMS to monitor the stored data, to automatically respond to the events and to execute defined actions in a timely manner. According to Dittrich (Dittrich, 1995), most researchers agree that ADBMS must detect event occurrences (E), satisfy specified condition (C), and perform defined actions (A). Active rule processing syntax includes two models: knowledge model and execution model. Knowledge model indicates what can be said about active rules in that system while an execution model specifies how a set of rules is treated at runtime. There are five phases involved in rules evaluation; signalling, triggering, evaluation, scheduling and execution (Paton, 1999). Some active databases have been developed by researchers but their usage has not been reported (Hanson, 1996).

Commercial databases such as Oracle (Adams, 2003) and MySQL may support some active features: triggers and events. Triggers in commercial databases, however, impose restrictions on the development of real applications. For instance, the limitation of number of triggers per table restricts the developer to build complex systems. Monitoring and control tasks can be implemented through ADBMS. In its simplest operation, all requests are usually filtered by an application layer that performs the monitoring tasks before they are sent to the DBMS. The problem with this approach is that it limits the way rule condition evaluation can be optimized.

3 ARCHITECTURE

This section discusses the architecture and the functional aspects of each component of the active database framework.

For health monitoring systems, it is essential to consider features such as reliability, security, reusability, scalability, and interoperability. Reliability is crucial so that the monitoring system can serve its purpose and acceptable to its users. Security is one of the primary design factors that has to be integrated at the design stage. Reusability allows the implemented system to be reconfigured, reused and extended with desired functionality appropriate for different monitoring/surveillance applications in different settings, environmental conditions and purposes. Scalability is necessary to handle a large number of users and devices that are geographically separated. Furthermore, the architecture must facilitate the interoperability whereby different types of sensor devices can be dynamically plugged into the system when necessary, or when the system has to interoperate with other existing systems.

Our active database monitoring architecture, as depicted in Figure 1 is divided into four main modules: the validation module, the active engine module, the decision making module and the user interface module. The details of each module are discussed in the following section.

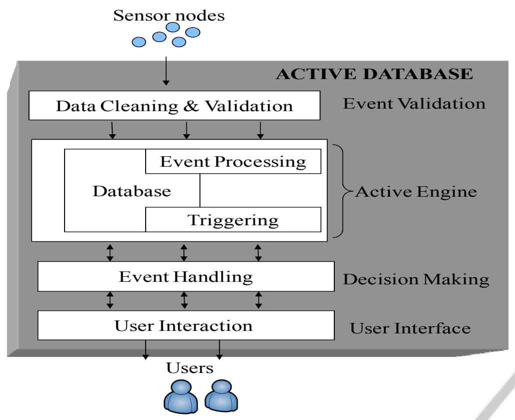

Figure 1: Active Database Monitoring Architecture.

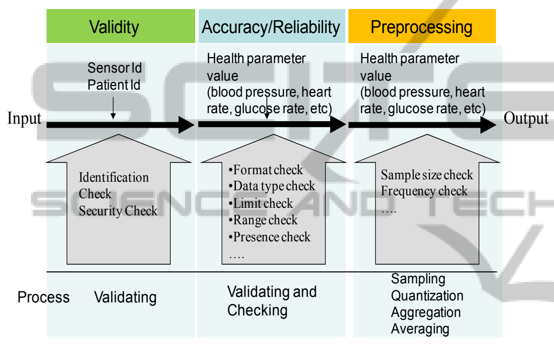

Figure 2: Event Validation Module.

3.1 The Event Validation Module

The Event Validation module serves as a pre-processing module for any monitoring systems. It is an essential component, however, often neglected. If data from the sensors are not properly validated, it will be difficult to build a reliable monitoring/surveillance system for the intended purposes such as detecting the correct digital signature of a disease, raising a critical disaster alert or delivering the right message to the intended user.

This module consists of an array of functions that deal with validating data, checking for data accuracy and selecting initial simple processing procedures (Figure 2). As not all functions are needed, these functions can be flexibly programmed or selected according to the intended application. Figure 3 illustrates the use of this module for a specific application where only relevant functions are selected and configured.

The main function of this module is to produce valid, accurate and summary data for further processing. For example, on validity check, the incoming data from the sensors are checked to make sure that they are from the intended sensor, from the intended users and are within the valid operational range. Therefore this element will possess mechanisms for device and user identification and will validate the signals to ensure they are within bounds and not altered by interferences including security and identity interferences. For accuracy check, this module aims to produce the best estimate of the received data to an acceptable resolution level. It will involve error detection, signal detection and filtering and interference cancellation.

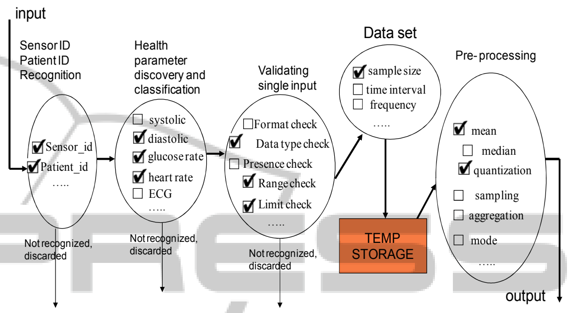

Figure 3: Example for Specific Application.

Furthermore, data properties such as numeral, data type, data range, sample size of each sensor data are assessed and checked at this module. Moreover, the outliers among the sensed data are detected and eliminated. For reliability check, this module guarantees that data are reliable and not spurious or transient ones. This module considers appropriate selection processes such as sampling frequency and simple data aggregation. Furthermore, simple forms of transformations such as averaging or selective filtering can be applied.

3.2 The Active Engine Module

This module consists of a database, an event processing and a triggering component. The pivotal database is developed based on entities that are identified in the application. Each of the entities represents a passive table which stores all data collected by the system. In our active database, the database also contains active tables. An active table stores the input which activates a trigger. Consequently, every tuple in an active table can be viewed as an expression of interest (EOI) of the system and to be handled appropriately.

The triggering component is responsible for assessing and evaluating the trigger and determines whether or not it should be treated as alert. This component and the event processing share the responsibility of combining and integrating triggered inputs with other relevant application parameters such as current and past data related to the targeted event. As the monitoring/surveillance task can be complex, each trigger invocation could convey different contexts and scenarios in the monitoring environment. For example, if the value of blood pressure (BP) is high, it needs to be assessed first before being treated as an alarm. This assessment is done by checking corresponding parameters related to the blood pressure. It is important to know the current activity of the person because it bears some effects on the interpretation of the BP value. For example, high BP reading of a patient who is at rest (the person is sitting or lying on the bed) or active (the person is walking, jogging or running) should be interpreted differently.

In our approach, active rules are constructed to monitor specific data records in the database. An active rule consists of three main components: event, condition and action (ECA). The event defines a rule-triggering occurrence. The condition is the set of circumstances that must exist for the action of the rule to be processed. The action is the task to be carried out if the relevant event has taken place and if the condition is fulfilled. Figure 4 shows a simple example of the ECA rule that is written in our implementation.

```
Event     { CREATE TRIGGER ai_CheckBP_fer
          {   AFTER INSERT on Visit
            FOR EACH ROW BEGIN
Condition {        IF   ( Condition 19) THEN
Action    {        CALL  Procedure 112;
            END IF; END
```

Figure 4: An Example of an ECA Rule.

Some existing database management systems do support simple triggering mechanism but they only allow simple ECA rules with single input, single condition and simple action. Our active engine module allows complex events to be constructed and programmed from multiple simple events and multiple conditions. This extension is required to deal with complex situations and the intelligent level of the application. Complex situations may involve interrelated multiple events, multiple conditions, current as well as past history of relevant data. Examples of these are as follows:

a) An action is triggered only if an event occurs over some specified conditions and when certain relevant historical data are presented.

b) An action is triggered only if an event occurs over some specified conditions and when certain relevant multiple sets of historical data are presented.

c) An action is only triggered if multiple events and conditions are met.

d) An action is only triggered if multiple events and conditions are met and when certain relevant historical data are presented.

e) An action is only triggered if multiple events and conditions are met and when certain relevant multiple sets of historical data are presented.

f) An action is triggered as in (a) – (e) and some condition on the immediate past actions.

Cases (e) and (f) are illustrated in Figure 5.

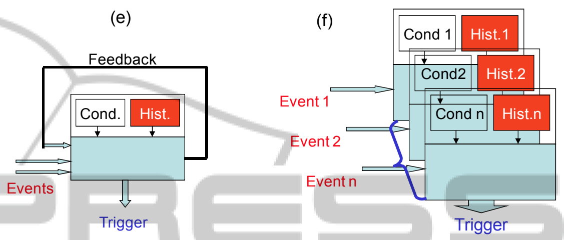

Figure 5: An action is triggered as resulted from events.

3.3 Decision Making Module

This module analyzes relevant information to produce a decision that must be acted upon in response to the triggered event. For instance, it is responsible for generating appropriate alert and notification messages. Our system considers both internal and external conditions for dynamic handling of the event instead of statically defined actions. We provide a comprehensive set of patterns for handling the alert events:

- Actions only involve the local database. Some value of a data record is deleted, altered.
- Actions may cause some external processes to be invoked.
- Actions may cause some external processes to be invoked and some internal data values are altered.
- Actions may involve a series of internal processes and may invoke external processes and some internal data values are altered.

This design allows developers to select the handling pattern that is most appropriate for their application.

3.4 The User Interface Module

Lastly, the actions, alerts and/or notification messages are communicated to the affected users or the environment. The active database architecture, however, does not deal with end user interaction. It mainly provides an interface for accommodating various sets of user interaction protocols. The active database architecture is not concerned itself with the effects of its action as far as the users are concerned, however, the user interface facilitates user interaction that is considered important to the design of a complete monitoring/surveillance system.

4 AN ACTIVE MATERNITY CARE APPLICATION

Based on our proposed active database architecture, we have developed the Active Maternity Care (AMC) system with a particular Assistive Pregnancy Care Loop application. The system was tested in a trial workshop setting with midwives and pregnant women from a New South Wales Health area service (Hoang, 2008).

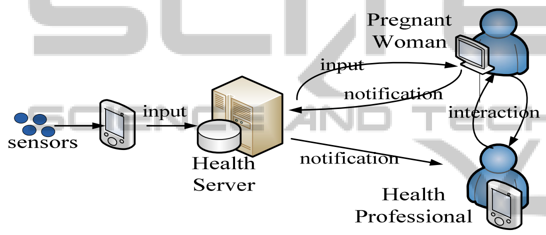

Figure 6: AMC current implementation.

Figure 6 depicts the implementation set up. The AMC system can be accessed using Internet browser on laptop/desktop computers or Personal Data Assistants (PDAs). In this implementation, Internet connectivity through the HP iPAQ HW6960 PDA is established using GPRS networks. The database which maintains the health data was developed using MySQL 5.0. We used Apache Tomcat 5.0 as the web container and Internet Information Service (IIS) as the web server. We adopted Nokia 6280 as GSM modem connected to the health server via the Bluetooth communication port. In an ideal scenario, the system anticipates an input from the woman. The woman then inserts the input to the system. Upon receiving the input from the woman, the system will send an acknowledgement to the woman. There could be several issues such as loss of data while transferring, sensor or woman does not respond to the request, etc. Therefore, during the testing, we developed three scenarios to test the functionality of the AMC system. In the first scenario, the woman failed to respond to the reminder message sent by the system. After sending a maximum of three reminder messages, the system will send alerts to the responsible clinician. The second scenario was when the value of the blood pressure (BP) was out of range. The system responded with an assistive advice and message to the woman as a reminder. Consequently, the system requested for another BP measurement. In the third scenario, a severe BP reading was recorded. During this time the system not only responded with a clear message to the woman but also notified the responsible midwife about the situation. The communication between the pregnant woman and the midwife was done via SMS and email, depending on the urgency of the notification. We are currently extending the interface to allow Bluetooth and Zigbee sensor nodes to send vital health data wirelessly to the PDAs that then relay the data to the pre-processing module of the system. We simulated the value for the sensors and forwarded it to the application.

At present, the event/data conditioning module mainly performed simple error checking and data validation. The active engine for the AMC system was developed using MySQL triggers and events.

Our initial implementation was not an efficient one because of a limitation of the MySQL database: the trigger mechanism could not activate an external procedure for processing. We have extended the triggering mechanism with listening threads and generating internal trigger information files. With this extension, an automatic notice was received by the application. However, without the ability to call an external process, the system could not handle multiple triggers and complex event processing.

Our current implementation has overcome this limitation and allowed efficient external procedures to be invoked. This is significant as the overall system is more responsive and allows more complex event handling mechanisms and intelligent decision support algorithms. Figure 7 illustrates the logic of the active engine of our system.

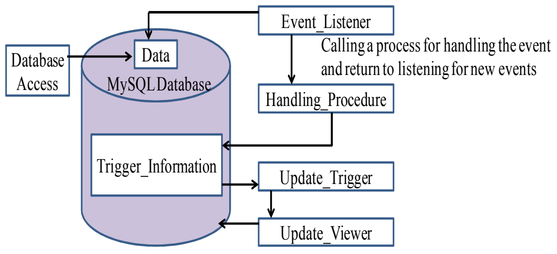

Figure 7: Active Database Implementation with MySQL.

Basically, an Event_Listener is waiting and listening to a triggered event. When it receives a trigger it forks another process to handle the triggered event and comes back to listening. Multiple triggered events hence can be handled without being delayed or lost. The handling processes are external from the database and hence sophisticated processing can be done without degrading the performance of the database. Figure 7 illustrates a particular example where a new piece of data was inserted in the "monitored record" and a trigger was initiated. When the Event_Listener caught this event, it forked the Handling_Procedure to update the trigger information in the database. The Update_Trigger and the Update_Viewer are used to update and display the changes to the user.

5 CONCLUSIONS

This paper presented an approach whereby critical events of interest can be automatically captured and appropriate responses can be pushed timely to affected users. The proposed active database component may be deployed in many security surveillance or monitoring vital health condition applications. A simple application to the maternity care is demonstrated. In the future, we plan to enhance our implementation to handle more complicated situations including i) new processing algorithms that deal with a richer set of data and events; ii) intelligent algorithms and new methods of user/system interaction; c) efficient scheduling algorithms to handle complicated situations that demand coordination, synchronization of arrangement. We will also explore the use of the information provided by the database on relevant history of the events to reliably determine the course of action. We will also describe our mechanism for dealing with critical events in mission-critical surveillance applications.

REFERENCES

Adams D. and Paapanen, E., 2003. Part III-The Active Database. In *Oracle Database Application Developer's Guide - Fundamentals, 10g Release 1 (10.1)*: Oracle Corporation., 2003.

Alamillo, T., Alsinet, Bjar, R., Anstegui, J., Fernndez, C. and Many, F., 2003. In Automated monitoring medical protocols: a secure and distributed architecture," *Artificial Intelligence in Medicine,* vol. 27, pp. 367-392, 2003.

Dayal, U., Hanson, E. N. and Widom, J., 1995. Active Database Systems. In *Modern Database Systems: The Object Model, Interoperability, and Beyond*, W. Kim, Ed. Massachusetts: ACM Press and Addision-Wesley, 1995, pp. 434-456.

Dittrich, K. R., Gatziu, S. and Geppert, A., 1995. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. In *Proceedings of the Second International Workshop on Rules in Database Systems*, Athens, Greece, 1995.

Godo, L., Puyol-Gruart, J., Sabater, J. and Torra, V., 2003. A multi-agent system approach for monitoring the prescription of restricted use antibiotics. In *Artificial Intelligence in Medicine,* vol. 27, pp. 259-282, 2003.

Hanson, E. N., 1996. The design and implementation of the Ariel active database rule system. In *Knowledge and Data Engineering, IEEE Transactions on,* vol. 8, pp. 157-172, 1996.

Hoang, D. B., Homer, C., Lawrence, E., Foureur, M., Ahmad, N. F. Balasubramanian, V. and Leap, N., 2008. Assistive Care Loop with Electronic Maternity Records. In *10th International Conference on e-Health Networking, Applications and Services (Healthcom)* Singapore, 2008.

Kotz-Dittrich, A. and E. Simon, E., 1998. Active Database Systems: Expectations, Commercial Experience, and Beyond. In *Active Rules in Database Systems*, N. W. Paton, Ed. New York: Springer-Verlag, 1998, pp. 367-404.

Koutkias, V. G., Chouvarda, I. and Maglaveras, N., 2005. A multiagent system enhancing home-care health services for chronic disease management. *Information Technology in Biomedicine, IEEE Transactions on,* vol. 9, pp. 528-537, 2005.

Moreno A. and Isern, D., 2002. Accessing distributed health care services through smart agents," in *Proceedings of the 4th IEEE International Workshop on Enterprise Networking and Computing in the Health Care Industry (HealthCom 2002)*, Nancy, France, 2002, pp. 34-41.

Paton, N. W., 1998. Active Rules in Database Systems. D. Gries and F. B. Schneider, Eds. New York: Springer-Verlag, 1998.

Paton N. W. and Diaz, O., 1999. Active Database Systems. In *ACM Computing Surveys,* vol. 31, pp. 63-103, March, 1999.