Updating Electronic Health Records with Information from Sensor Systems: Considerations Relating To Standards and Architecture Arising From the Development of a Prototype System

Bruce Moulton[*1], Zenon Chaczko[2], Mark Karatovic
[*1] **Corresponding author** *School of Electrical, Mechanical and Mechatronic Systems,*
[2,]*School of Computing and Communication,*
Faculty of Engineering and IT, University of Technology, Sydney, Australia
brucem@eng.uts.edu.au

doi: 10.4156/jcit.vol4.issue4.3

Abstract

Several countries around the globe are moving towards national and international standards for Electronic Health Records (EHRs). One function of the standards is to guide the long-term convergence of local systems into integrated evolving national health information systems. The Australian commonwealth government is implementing a nationwide EHR system whereby every Australian will be able to upload data to his or her EHR. Thus Australians, if they wish, will eventually be able to upload data from on-body sensors and in-home monitoring systems to their EHRs. This article explores issues associated with the architecture of systems which allow medical records to be updated with information from monitoring/sensor systems. A prototype was developed to determine some of the key architectural considerations. A sensor simulator was implemented for testing purposes which allows a user of the simulator to impersonate a bed or group of in-home or on-body sensors connected with a person who is in a hospital, retirement home or private home. Findings are discussed relating to key architectural considerations including security, maintainability and modularity.

Keywords

Electronic health records, eHealth, remote sensing, monitoring, vital statistics, aged care.

1. Introduction

An Electronic Health Record (EHR) stores information relating to the health status of an individual. The information contained within an EHR may be used for several purposes. First, it is used to support the provision of health care to the owner of the record. Second, with the owner's consent, it may be used to benefit the health of the wider community by way of analysis for purposes of quality management, research or education.

Several countries around the globe are moving towards national and international EHR standards including ISO and HL7. In Australia, a recently released June 2009 *Final Report of the National Health and Hospitals Reform Commission* recommends that by 2012 every Australian should be able to (1) have a personal electronic health record that will at all times be *owned and controlled by that person*; (2) approve designated health care providers and carers to have authorized access to some or all of their personal electronic health record; and (3) choose their personal electronic health record provider [1].

The report further recommends that the Australian commonwealth government should take responsibility for a national policy and open technical standards framework for e-health. According to the report, these standards should relate to interoperability, compliance and security, and should be developed with state governments, the IT vendor industry, health professionals, and consumers [1]. A function of the standards is to guide the long-term convergence of the currently deployed local systems into an integrated (and continually evolving) national health information system .

Standards Australia's IT-014 Health Informatics Committee indicates that the benefits of standardizing EHRs include improvements in patient safety, quality of health care, consistency of language, referrals, reporting of test results, and transfer of information between health providers [2].

The IT-014-09 Electronic Health Records Interoperability Subcommittee is developing standards to facilitate the sharing of EHR information. One of the objectives of the Subcommittee is to improve alignment with proposed ISO, HL7 and other global standards that are relevant for EHR interoperability.

The EHR Interoperability Subcommittee recommends that EHRs will not only be accessible to consumers, but may also "incorporate their views and comments resulting from self-monitoring of illness or wellness, dietary notes, notes on self monitoring of sport and exercise performance, behavioral activities and moods, etc" [3]. It also notes that rapidly changing technology means that the architecture of the EHR must be able to accommodate new forms of clinical knowledge and completely new types of data.

Such specifications open the possibility for on-body sensors or in-home health monitoring systems to directly upload data to our EHRs. While this may be unlikely to occur in the immediate future, the standards are intended to be applicable for the "evolving" health information systems, and it seems inevitable that on-body sensors (and eventually, implanted sensors) will have an increasingly important role in the provision of health services in the coming decades.

Taking this into consideration, we report on part of a project which explores ways of integrating data from sensors with an electronic health record. Prior research suggests that a referent tracking system may be useful for addressing these goals, but that approach can require approaches radically different from those currently used for clinical coding and terminology [4].

2. Method

An objective of the project is to explore issues that arise when designing a system which allows medical records to be updated with information from monitoring/sensor systems. Requirements analyses have been undertaken which consider sensor-related EHR use-case scenarios. One use-case scenario was that a sensor system might detect a gradual change in a measure such as weight or ECG, and send this information to the EHR to be flagged as a significant event. Another scenario was that information such as daily weight from an outpatient or in-home sensor-system would be regularly logged on the EHR, regardless of whether that information would be flagged as a "noteworthy" event. In the first scenario, the system would in part allow real time monitoring of any significant events and both store and relay them to an allocated doctor or nurse within the institution where the patient is located. In the second scenario, the system might track the person's body movements, position and any significant changes to the person's vital statistics.

The requirements analysis also considered general requirements of medical records include such as:

- Reports should be brief, accurate and complete
- Reports should be objectively written;
- Reports should not contain ambiguity
- Abbreviations should not be used unless well known
- Any errors should be crossed out and rewritten in a way that the error is still legible allowing for basic accountability in changes to the records.
- No entry should be written on behalf of another, to allow accountability for all actions performed by people or sensor systems that write to the EHR [5].

Data models, class diagrams and architecture diagrams were developed, and a prototype was constructed and evaluated. The initial evaluations led to a revised architecture, which was used to develop a proof of concept.

A sensor simulator was implemented for testing purposes. The simulator implements the sensor protocol used to communicate to the server. This allows a user of the simulator to impersonate a bed, regardless of whether that bed is within a hospital, retirement home or private home. The simulated sensors in the program then attach themselves to that bed/home and download the configuration for the patient assigned to that bed. Based on this configuration they can send sensor events to staff assigned to the person on that bed.

3. Results/Discussion

It was initially expected that the JINI framework would be potentially suitable, in part because it can conveniently handle broadcast to several different devices and allows for device discovery when devices are attached to the network such as sensors. It also has a distributed event mechanism that allows devices to subscribe and observe other devices on the network which would be highly suitable for broadcasting information across a network. Programming the solution via TCP/IP socket connections was considered undesirable because it would require some reinventing of the wheel. However, an advantage over JINI is that when the connection is dropped the framework can notify the server that an intended recipient is not available instantly rather than waiting for the next lease discovery phase. Java Message Service (JMS) would allow software to broadcast data asynchronously across multiple subscribers similar to JINI. It does not however allow for automatic discovery of services and does not give any warning if one intended destination is not available. A result of the relevant considerations was that the prototype would implement a new framework via its own protocol that allows for customized polling, centralized information distribution and that has its own security protocol defined. The initial architecture is given in Figure 1.

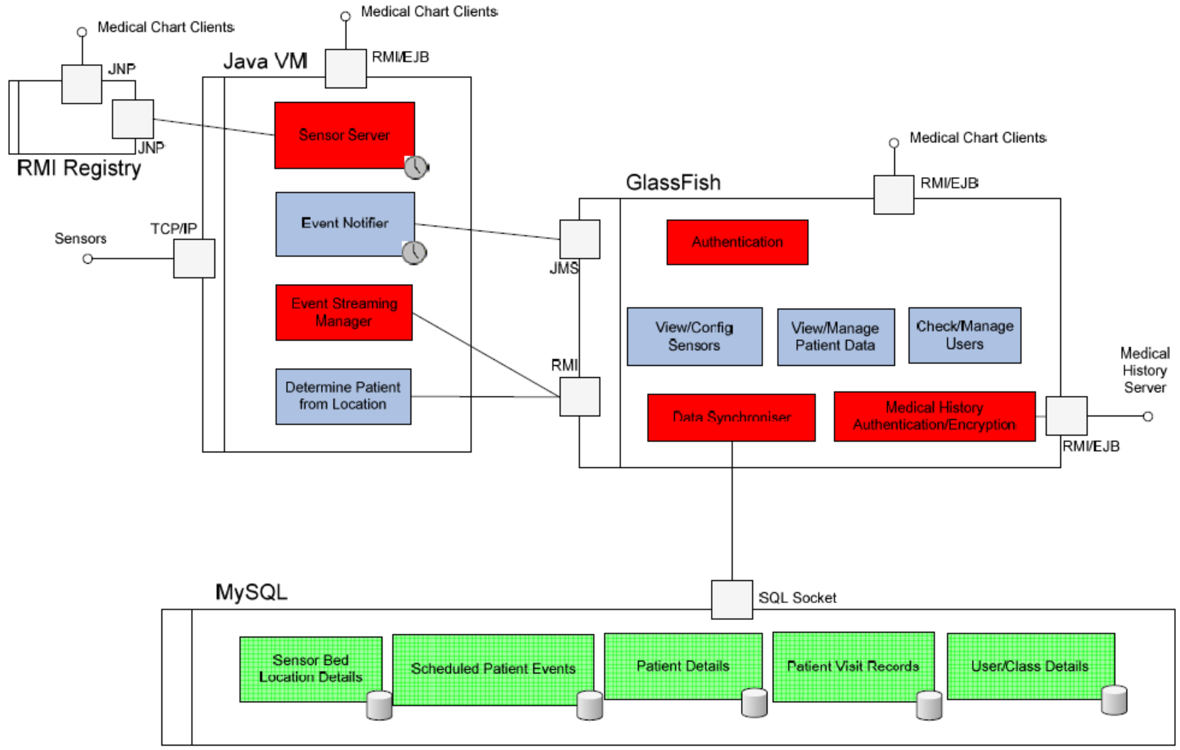

Figure 1. Initial Architecture

The initial architecture can be split into two different components. These are the Java VM stand-alone "event server" application and the Java EE server side components running on top of the Glassfish Application Server. The event server in its own Java VM communicates with the Glassfish server to fetch data on patients, their bed locations and to send events to the clients. Clients connect to the event server via RMI after looking up the server via a JNDI request using the JNP protocol on the lookup server. When sensors issue events via their TCP/IP connections to the server the server alerts each of the clients via a callback RMI call. Clients can also create new events for a particular patient. When a new event is created the EJB Glassfish server will send a message to the event server via a JMS (Java Messaging Service) message. When the event server receives this message it will perform an EJB call via RMI to fetch the new event cache. The EJB Glassfish server communicates with its database via an SQL TCP/IP connection to the MySQL server. In addition any EJB calls that alter a patient's record or details are sent to the medical history server by means of an EJB call.

After the preliminary evaluation, alterations were made to parts of the architecture. Requirements were relaxed on the event server including the need for total real-time performance and reliability. The requirement for staff to be notified when a patient's sensor is no longer connected to the server was also relaxed for the prototype. With these constraints no longer an issue, the Event Server was able to be moved to the EJB Glassfish server, and implemented by a combination of EJB calls and JMS messages. JMS messages are used to broadcast to particular sensor a request for information. This request can either be to send a snapshot of their current reading to the server generating a patient record or to refresh their configuration when a user updates the sensor configuration for a patient. JMS messages are addressed to a particular sensor on the network determined by a combination of the sensors type and the bed it is assigned to. To honour any JMS requests the sensor calls an EJB on the Glassfish server to either send a sensor event to any interested clients or to create a record of the current sensor reading. In order for the sensor to comply with the data model, any records that it creates must be associated with a user in this case the system user dedicated to the sensor network.

For sensors to be able to join this network they need the network security key which is stored in the sensor system user object instance as its password. All records are then marked as created by this sensor network user representing all sensors of the current medical institution. In order to simplify the communication implementation for the proof of concept another requirement was relaxed, that being the need for records to be created when no client received the sensor event. This allowed the system to broadcast events to clients via a JMS message. The JMS message contains the Event object or one of its subclasses with its appropriate records attached. It also contains an array of users meant to be receiving it. Each client runs a JMS protocol driver which decodes the message and filters out any messages not intended for the current user. It then updates any threads on the current clients that are subscribed to the current message.

For the prototype, SSL based connections were not implemented. In a production environment some form of encryption would be necessary so that patient data transported across the network and across public networks would be less susceptible to being accessible to an unintended third party.

The resulting event subsystem architecture is given in Figure 2.

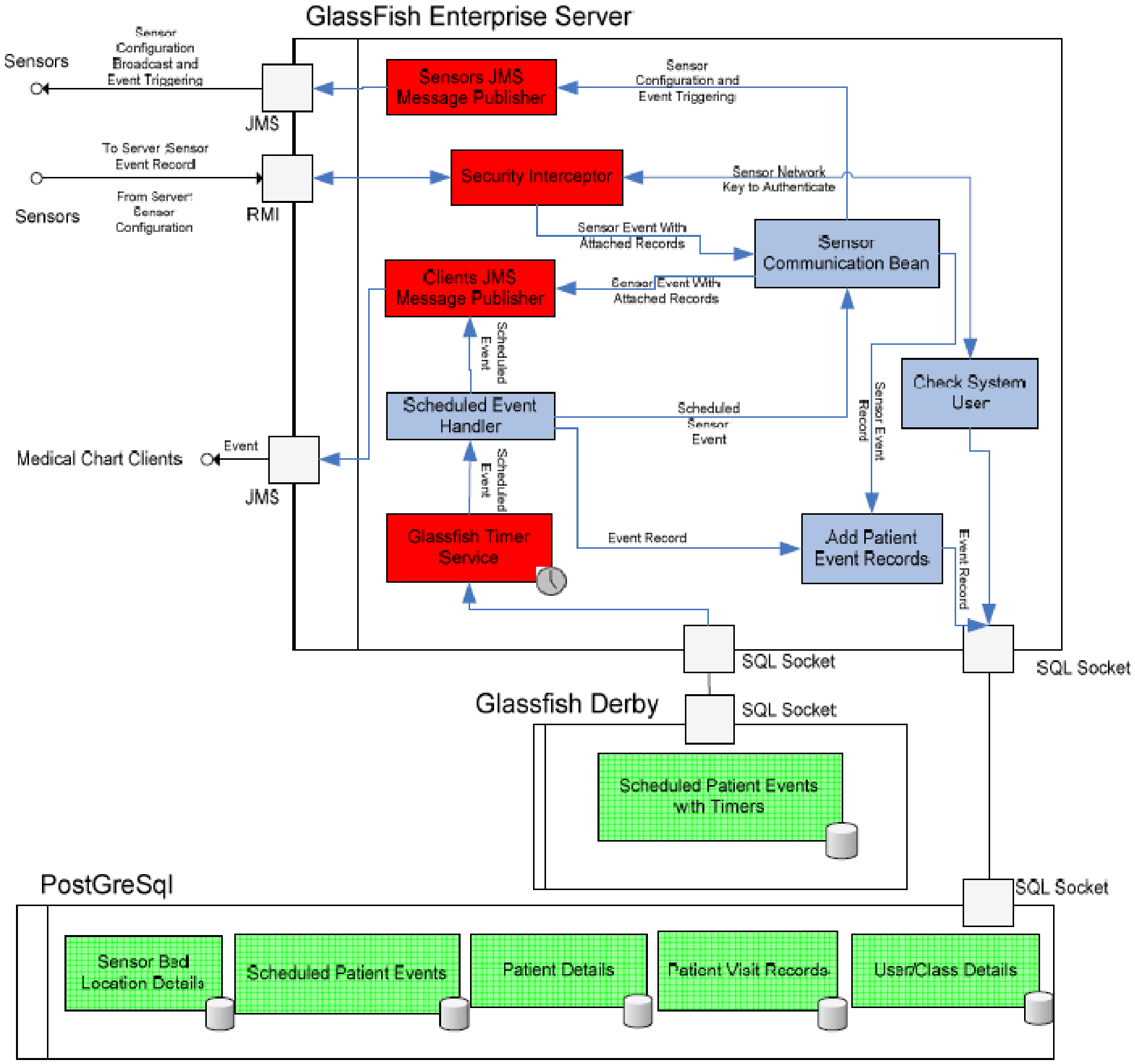
GlassFish Enterprise Server
Sensors
Sensor Configuration Broadcast and Event Triggering
JMS
To Server Sensor Event Record
From Server Sensor Configuration
Sensors
RMI
Sensors JMS Message Publisher
Sensor Configuration and Event Triggering
Security Interceptor
Sensor Network Key to Authenticate
Sensor Event With Attached Records
Sensor Communication Bean
Clients JMS Message Publisher
Sensor Event With Attached Records
Scheduled Event
Scheduled Event Handler
Scheduled Sensor Event
Sensor Event Record
Check System User
Medical Chart Clients
Event
JMS
Scheduled Event
Glassfish Timer Service
Event Record
Add Patient Event Records
Event Record
SQL Socket
SQL Socket
Glassfish Derby
SQL Socket
Scheduled Patient Events with Timers
PostGreSql
SQL Socket
Sensor Bed Location Details
Scheduled Patient Events
Patient Details
Patient Visit Records
User/Class Details

Figure 2. Final prototype event subsystem architecture

All bean classes are assumed to implement the EJB 3.0 standard. This includes them implementing the correct interfaces, that the correct annotations are defined, and they are being called upon inside a Java EE container. Another assumption is that the get() and set() methods for data classes are implemented for each attribute defined and as such are not defined as operations in their respective classes. This being a common practice in class diagrams. All bean classes will conform to the Enterprise JavaBeans 3.0 specification and as such will be the interfaces to the Glassfish server.

A class named SensorServer implements the sensor server, which clients connect to. The class is threaded, hence the run() method. A SensorDataBean class is connected to by clients whenever they wish to customize the configurations from the sensors. It allows them to get and set the configurations which are stored on the sensors and the connection status to the sensors registered.

For the prototype, efforts were concentrated into validating the process for sending and receiving sensory events from patients. Working sensors that adhere to this process can be implemented in the future should the project proceed. A subsequent phase could change the design of the sensor framework so that it either uses the proposed architecture or the JINI framework for monitoring sensors and their current status. For the proof of concept, the sensor configuration allows the tester to specify higher and lower bound threshold values. If the sensor records a reading that is not between these values the sensor sends a snapshot of the current reading to assigned staff members as a severe event that needs attention. A more customizable configuration would allow different events with different severity levels to be sent when the sensor is recording values in different ranges. It would also allow configuration of whether further information should be sent such as a diagram allowing for non-numerical sensor data.

As a result of the analyses and evaluations, several architectural issues of importance were identified. Key results concerning architectural qualities are given in Table 1.

Table 1. Summary of findings concerning key architectural considerations

Scalability	The technology needs to be scalable so that more users can be added without interruption (reliability) or without redesign. It should be able to keep alive connections to any patient hardware connected to it and it should be scalable to any number of patients and sensors given hardware constraints. Hardware should be able to be added to increase the capabilities of the system with little or no interruption. The medical system will need to be deployed in many locations.
Reliability and Robustness	The system and medical history clients should have a backup in case a failure of the system occurs. The system and medical history clients should have a local redundancy in case the link to the main server is down. This can be synchronized when the link is back up. The medical history system should have some sort of offline mode in case the system goes down in which it will synchronize the information later. This needs to be bundled with security to make sure that the information can not be accessed locally in an unsecured manner by non-authorized people. The medical chart system should have a local redundancy that if the link to the main server is down the medical institution can still look up their own records.
Modularity	By its very nature the system incorporates a wide variety of different clients at different locations. It must be able to meet all the requirements whilst spanning a large area with many different subsystems.
Maintainability	This system should be able be maintained in stages so that the overall functionality of the system is not affected. The user should be notified of any real-time data links that do break so that they may be fixed. Relevant stakeholders should be notified of links that are non-functioning to alleviate the risk to the patients affected and to the medical institution.
Performance	The system on its real-time links should meet any specified performance requirements. Information synchronization and information requests need to occur within an acceptable time for the user.
Security	The system at all points should have authentication layer to make sure all data being passed back and forward has been authorized and encrypted if necessary. The authorization layer shall make sure that data being accessed is relevant for: the class of user accessing the data and the circumstances of the user and their relationship to the client (e.g. a patient appointment with a GP).

4. Conclusions

One of the motivators for the introduction of globally standardized and nationwide EHRs is to allow more-integrated sharing of information between members of multi-specialty and multi-disciplinary healthcare teams. The Australian commonwealth government is implementing a nationwide EHR system whereby every Australian will have access to his or her individual EHRs, and will also be able to control who else has access to various elements of the EHR. In addition, each Australian will be able to upload data to his or her EHR. An implication of this is that Australians, if they wish, will be able to upload data from on-body sensors and in-home monitoring systems to their EHRs. This article explores issues that arise when designing a system which allows medical records to be updated with information from monitoring/sensor systems. A prototype was developed to determine some of the key architectural considerations. Security, maintainability and modularity were identified as three key considerations for the design of such systems.

5. References

[1] A Healthier Future For All Australians – Final Report of the National Health and Hospitals Reform Commission – June 2009 http://www.health.gov.au/internet/main/publishing.nsf/Content/nhhrc [accessed 31.6.09].

[2] Standards Australia e-health IT-014 http://www.e-health.standards.org.au/cat.asp?catid=5 [accessed 30.6.09].

[3] AS ISO 18308-2005 Health Informatics - Requirements for an electronic health record architecture (ISO/TS 18308:2004, MOD).

[4] Ceusters W, Smith B. (2006) Strategies for referent tracking in Electronic Health Records. Journal of Biomedical Informatics Volume 39, Issue 3: 362-378.

[5] Staunton & Whyburn (1997), Nursing and the law, 4th edn., W B. Saunders, Sydney.