9th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS, Suceava, Romania, May 22-24, 2008

TELEMON – an SOA-based e-Health System. Designing the Main Architectural Components

Lenuta Alboaie, Sabin C. Buraga, Victor Felea
"A. I. Cuza" University of Iaşi, Faculty of Computer Science
16, Berthelot Street, RO-700483 Iasi
e-mail: {adria, busaco, felea}@infoiasi.ro

Abstract—**The paper proposed a service oriented approach for the modular architecture of an e-health system called Telemon. We identify the most important issues regarding the application interoperability and we proposed several solutions, following the SOA (Service Oriented Paradigm) paradigm.**

Index Terms—**E-health, Design, SOA, Web Services**

I. INTRODUCTION

This paper focuses on the component modules of an e-Health system, called *Telemon* [1, 9]. Telemon is an e-Health system, intended to allow real time patient monitoring by using Web technologies. The current work continues previous work described in [1]. Previously, we presented the context and the arguments that sustain our solution – an SOAarchitecture for the Telemon project.

In order to create a viable architecture, we propose several important modules regarding certain aspects such as the user-interaction, the SSO-like user authentication, the message routing, the data storage, and the overall management of the application.

The paper has the following structure: section II presents the most important aspects of SOA [12, 15] paradigm and section III gives information regarding the general architecture of the Telemon system. In section IV, we present a BPM (Business Process Modeling) analysis of our architectural approach and we detail the main Telemon components. The article ends with the conclusions and further work.

II. SERVICE ORIENTED ARCHITECTURE

The term SOA (Service Oriented Architecture) refers to the design of a distributed system. SOA is an approach that leads to take concrete decisions when you design concrete software architecture. Therefore SOA is a design methodology, aimed at maximizing the reuse of multiple services (possibly implemented on different platforms and using multiple programming languages)[14, 15].

In a SOA platform, the services generally have some important characteristics [3, 4]:

- Services are individually useful – they are autonomous;
- Services must be loosely coupled. This term implies that services discover the needed information at the time they need it. The benefits offered by this characteristic are: flexibility, scalability, ability to be easily replaced, and fault tolerance;
- Services can be composed to provide other services. This promotes the reuse of existing functionality;
- Services can participate in a workflow. An operation performed by a service will depend on the messages that are sent or received - service choreography;
- Services can be easily discovered, eventually in an automatic manner. Therefore, services must expose details (and additional meta-data) such as their capabilities, interfaces, policies and supported protocols. Other details such as the programming language or the information about the platform are not useful for consumers and – usually – are not revealed.

SOA is a paradigm based on three major concepts: services, interoperability through an enterprise service bus (ESB) and loose coupling. ESB is an infrastructure that enables interoperability.

To use in practice services, we need an ESB, which role is to enable consumers to call the services providers.

Mainly, ESB has the following responsibilities [4]:

- Providing connectivity;
- Data transformations;
- Routing;
- Dealing with security;
- Dealing with reliability;
- Service management;
- Monitoring and logging.

III. TELEMON ARCHITECTURE. AN OVERVIEW

In [1], we proposed a general architecture consisting of two types of components: local sub-system components and a central system component.

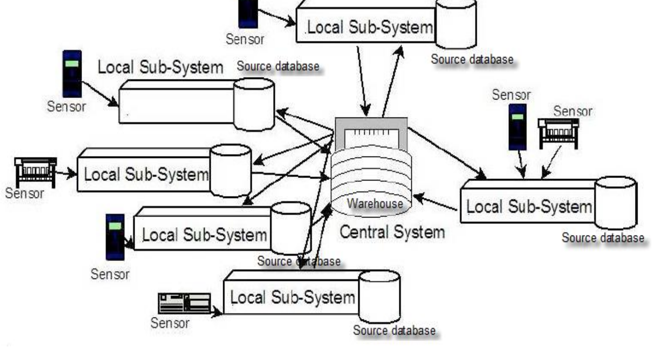

Figure 1.Telemon – general architecture

The local sub-systems are components that collect information from different sources (devices that observe the patient). All this information from local sub-systems is transferred (through an update mechanism) to the central system.

The general architecture of our health system platform is depicted in Figure 1.

We will outline the general structure of the components – local subsystems and the central system, both of which conforming to SOA principles. For each of them, we will decompose the architecture in several levels.

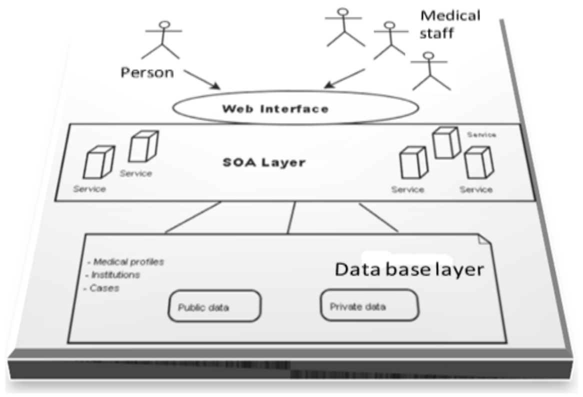

Figure 2. A sliced overview of a component

Each of the components consists of the following layers:

- User-interaction layer;
- Core layer;
- Database layer.

For the user-interaction layer, we consider our system is enriched with accessibility capabilities, from the user interaction perspective. The provided user-interface must support persons having various disabilities, according to WAI (Web Accessibility Initiative) [10]. The performed activities must be effective, efficient and secure, at the conventional Web browser level and at the mobile application level.

An important feature will be the support offered by the GIS (Geographic Information Systems) [6] Web services, which will offer to authenticated patients the information about the known pharmacies, clinics, medical offices in their proximity.

Concerning the Telemon core layer, from the technical point of view the system must conform to the SOA paradigm. This layer consists of a set of modules detailed in the next section.

The aimed architectural solution will be a multi-platform one, loosely coupled, facilitating the integration of applications, services and systems at the Web level.

The system performance will be assured due to the scalability of the architecture based on actual Web standards (XML, SOAP, WSDL, REST, WS-*, etc.) [2, 4, 10, 11,13]. Also, Telemon can be integrated in so called e-Health meta-systems, like ARTEMIS [5] or Semantic Health [8], which are able to integrate different other e-Health systems, in order to share information and access a larger amount of data.

Using SOA architecture is beneficial for our system since it allows easily adding new features without modifying the existing ones. Because these are based on existing services, the code reuse is maximal, and then development and testing time is minimal.

At the database level, Telemon comprises two types of databases. The one located at the sub-system level is an operational database that records data sent by the sensors (they are named “source databases”). The other is a warehouse and is located at the central system level. All information is stored at this level.

IV. ARCHITECTURAL COMPONENTS OF THE TELEMON SYSTEM

For our architecture, we have two kind of ESB:

- In the local sub-systems, an ESB will provide a point-to-point connection. In this case, the consumer knows the endpoint and sends the request to a specific receiver. This type of connection has a problem when the receiver is not available; in this case the call will fail;
- Between central system and the rest of components, an ESB will offer a mediation connection. In this case, the consumer does not need to know the endpoint of the provider. The consumer will supply certain service identifiers like a tag that ESB interprets and find an appropriate provider. A tag contains service name and other elements such as routing information, accessing information, etc.

The advantage of this approach is that ESB can deal with dynamic changes of SOA components.

First step in our proposed architecture is to clearly define roles, policies and processes. The processes define a service lifecycle and implementing model-driven service development.

We discuss about necessary components for local sub-system architectures, and we take in count two directions: the consumer point of view (our consumer can be a patient or some person from medical staff) and the backend “point of view”.

We identify the following modules:

- Frontend module,
- Identity management module,
- Service management module,
- Routing module,
- Database module,
- Management application module,
- Warehouse management module.

The frontend module guarantees the user interaction with the system, at the *user-interaction level* in Telemon architecture.

The identity management module assures the authentication and authorization of users.

Within Telemon, the information can be accessed using an automated authentication and authorization system that allows a user that is authenticated and authorized on a local sub-system, to be automatically authenticated and authorized on the central system. The procedure works the other way around as well, that is, the user that is authenticated in the central system will be automatically authenticated in all the local sub-systems. The authorization to access information is based on user types and user groups (family doctors, specialist doctors, statisticians, researchers, patients). Furthermore, logging out from a local sub-system involves logging out from the central system.

To achieve this goal, we use a single sign-on (SSO) mechanism, which – according to [7] – is a system whereby a single action of user authentication and authorization can permit a user access to all computers and systems where she/he has access permission to, without the need to enter multiple passwords. Single sign-on reduces human error, a major component of systems failure and is therefore highly desirable for this kind of e-Health systems.

The service management module is a component that assures the access to available services.

The routing module guarantees that a request can reach at the Service Module level.

The database management module is a wrapper for local database and the management application module will allow operations that are not available for the customary user or medical staff – for example, log verifying or the management of update operations. This module interacts with the database component.

In the next paragraphs, we will describe the overall process from the user point of view.

When a customer needs by a generic service, the following general steps must be performed:

- Authentication and authorization realized by the identity management module;
- The request of user is routing (through the routing module) to the service management module (in local sub-system case, we have an ESB point to point connection).
- Sometimes, certain transformations are necessary to change data from one format to another format supported by system; in this case, the data transformation module is used.

In Figure 3 is depicted the local subsystem architecture.

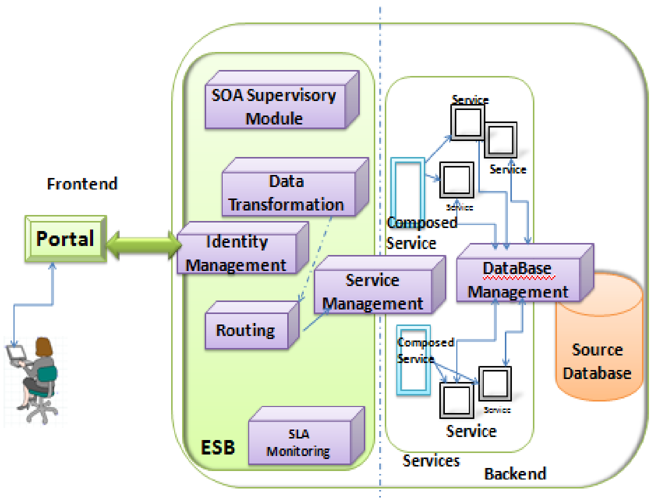

Figure 3. Local sub-system architecture

The warehouse management module is a wrapper for the central system warehouse. This module interacts – using special services – with the database management modules from every local subsystem. Therefore, in the warehouse we can find consistent data from any local sub-systems.

All this data came from services like *consumer services* which can be: update services or can be a particular request data service.

The source databases feed the warehouse with data. Our system, partially decentralized, ensures the data transfer from the local sub-systems to the central system.

To optimize the data traffic we design two different types of warehouse updates: instant updates and periodical updates.

Periodical updates are done on a regular basis and they apply to all the data. The instant updates depend on an emergency threshold, which refers to the severity of the patient's condition.

In the cases where the severity is above the emergency threshold, an instant update is applied in the warehouse.

This is crucial when an emergency case occurs and the doctor can consult the specialists that are logged in the central system at that time. The specialists logged into the central system are automatically logged on to the local sub-system (for example, using an OpenID module). In this manner, they can interfere in the patient treatment directly in the local sub-system in whose proximity the patient is situated.

The information concerning the treatment synchronizes back with the central system via the periodical or instant updates. Furthermore, when there is a serious case of a similar type in another sub-system, the doctor who is logged on there consults the warehouse to get the previous treatment in order to take a decision.

Figure 4 denotes the modules we described earlier from the perspective of the central system architecture.

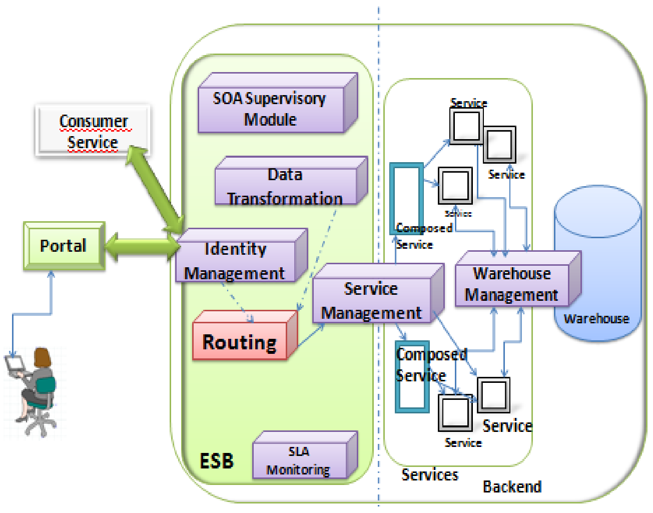

Figure 4. Central system architecture

SOA is a concept for large distributed systems [16]. Our architecture respects the properties of large distributed systems. From this point of view, we must consider the following important characteristics:

- *Legacies*: these mean that our SOA architecture must be integrated within an existing framework. In fact, SOA is an approach for the maintenance of an existing system.
- *Redundancy*: it is difficult to eliminate redundancy in a real system and it is not possible to have all data normalized and stored in only one place. So, the solution is maintaining consistency. For that, we must develop special update services.

- *Bottlenecks*: this problem creates difficulties for scalability.
- *Heterogeneity*: an SOA architecture deals with these situations. We can meet heterogeneity at many levels: platforms, programming language, technology, and vendor diversity. In our case, we consider a homogeneous system. The heterogeneity problem will appear when we integrate our system with other e-Health systems.
- *Loose coupling*: a minor problem can stop all functionality. To avoid that we must consider the following aspects: flexibility, scalability, and fault tolerance.

Loose coupling is a concept of minimizing dependencies. Even a problem appears the system still runs. For example in our case if a local sub-system is temporarily down, information belongs to that point can be accessed from central system.

In the same time, if the central system is down, local sub-system will store all information and will run in an independent way. Therefore our system treats fault tolerance and flexibility problems.

Other two important aspects concern the *reliability* and *performance*.

For our system, ESB contains a module that deals with reliability issues. Also, we plan to design a BAM (Business Activity Monitoring) module. This module allows us to learn about the state of our system (e.g., if a certain service is often call).

V. CONCLUSIONS

The paper provided an analysis of use of the SOA paradigm in the context of an e-health Web-based system, called Telemon [1, 9]. We proposed several important modules regarding certain aspects such as the user-interaction, the SSO-like user authentication, the message routing, the data storage, and the overall management of the application. Our approach was focused on the use of ESB (Enterprise Service Bus) techniques.

Our further direction of research is to provide more detailed view of the each module structure and to study the multiple problems that can arise, including the possible technological solutions.

REFERENCES

[1] L. Alboaie, S. Buraga, ”Service-oriented architecture for health systems,” Proceedings of the e-Health and bioengineering (EHB 2007) Workshop, 2007

[2] T. Bray et al. (eds.), Extensible Markup Language – XML 1.0 (Fourth Edition), W3C Recommendation, Boston, 2006

[3] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, 2005

[4] N. Josuttis, SOA in Practice. The Art of Distributed System Design, O’Reilly, 2007

[5] ARTEMIS Project, Available: http://www.srdc.metu.edu.tr/webpage/projects/artemis/

[6] Geographic Information Systems, Available: http://www.gis.com/

[7] Single Sign-On, Available: http://www.opengroup.org/security/sso/

[8] Semantic Health, Available: http://www.semantichealth.org/

[9] Telemon Project, Available: http://thor.info.uaic.ro/˜telemonfcs/

[10] World Wide Consortium, Available: http://www.w3.org/

[11] WSDL (Web Services Description Language), Available: http://www.w3.org/TR/wsdl

[12] Ramesh Loganathan, Poornachandra Sarang, Frank Jennings, Matjaz Juric, SOA Approach to Integration, 2007

[13] Eric Newcomer, Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison-Wesley

[14] Leonard Richardson and Sam Ruby, RESTful Web Services, O’Reilly, 2007

[15] L. Alboaie, S. Buraga, WebServices (in Romanian), Polirom Publishing House, Iasi, 2006

[16] George Coulouris, Jean Dollimore, Tim Kindberg, Distributed Systems: Concepts and Design, 2002, Addison-Wesley

[17] Thomas Erl, Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services Prentice Hall, 2007