Layered Implementation View of a SOA Based Electronic Health Record

Josimar S. Lima[2], Joyce M. S. França[1], Jislane S. S. Menezes[2], Adicinéia A. Oliveira[2] and Michel S. Soares[2]

[1]Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil
[2]Department of Computing, Federal University of Sergipe, São Cristóvão, Brazil
{*osimar.lima2007, joycefranca, jislanesds, adicineia, mics.soares*}@*gmail.com*

Abstract

Interoperability between legacy systems has crucial importance for integration of many distributed systems. Within health information systems, the need for interoperability between legacy systems is well-known, as these are heterogeneous systems which have a long lifespan. An interesting example of health information systems is the Electronic Health Record (EHR), a virtual record of every health-related event, including hospital admission, general practitioner visit, exams, and allergies experienced by individuals over their lifespan from in uterus to death. A great challenge for implementing EHR applications is that these systems often have to integrate large amount of data gathered from legacy systems into a single application. This paper describes in detail the implementation view of a software architecture based on web services for developing an EHR application in a public hospital. The architecture has been defined with a layered style for implementation and on services identified within current legacy systems. Detailed explanation on the software development, including architectural elements and services catalog are described as well.

***Keywords*-Interoperability, Service Oriented Architecture, Layered Implementation, Health Information Systems.**

I. Introduction

Health information systems are becoming common in health institutions not only for management purposes, but also to improve health care. Complexity of developing these software systems is well-known in the literature for many reasons, including heterogeneity of data and issues with interoperability between legacy systems [1], and design and architecture problems [2]. In addition, health applications are sensitive to important non-functional requirements, such as performance, security and safety of data [3] [4]. Therefore, developing and managing health information systems is a hard task. New functionalities arises constantly due to new ideas or government laws, and they have to be implemented and integrated to other legacy systems.

An Electronic Health Record (EHR) is a virtual record of every health-related event (e.g., hospital admission, general practitioner visit, exams, allergies) experienced by individuals over their lifespan from in utero to death [5]. EHR systems can be considered complex health information systems because they are used for many purposes, including recording data for decades and providing information gathered from data stored in many legacy software systems, developed using different technologies, databases, platforms and programming languages. EHRs normally have to integrate large amount of data gathered from legacy systems in a single application.

Benefits of an EHR system are better patient care, improved and faster communication with other systems, including external systems (government, health insurance, banking), improved sharing of data between health professionals, improved decision making, among others [6] [7]. However, these are complex systems to develop, maintain and operate. Failures during development or execution of EHR systems have been documented in the literature. Among the reasons for failures during development, a systematic literature review [5] mentioned a number of issues, including low level of user involvement, the need of redesigning work practices, and redesign of the record

DOI reference number: 10.18293/SEKE2016-064

format.

More specifically, the core subject of this paper is interoperability between legacy systems to implement a new EHR system. This problem has been addressed before in many ways [8], including by means of Enterprise Application Integration and Service Oriented Architectures [9]. For instance, the authors of paper [10] presented a framework with focus on defining services based on a Model-Driven Architecture approach supported by a SOA meta-model. In [11], the authors present a solution to interoperability in healthcare based on a middleware software architecture used in enterprise solutions. The general architecture is described, but implementation view and further development details are not explained. Authors of paper [12] describe an attempt to use SoaML with the purpose of modeling a real problem of system interoperability in health information systems. Integration of data in Electronic Health Information Systems is discussed in [13], in which, according to the authors, data interoperability in healthcare is, at present, largely an unreached goal. Therefore, the authors suggest the adoption of a standardized healthcare terminology, as well as the connection of legacy systems to the health network as ways of achieving complete interoperability of Health Information Systems.

Developing an EHR considering interoperability constraints between legacy software systems remains an interesting research problem. In this research, the issue of interoperability is considered by designing a Service Oriented Architecture (SOA) in order to integrate legacy systems by means of web services. An EHR application is developed as a case study. The layered implementation view of the architecture is presented, as well as detailed explanation on the software development, including architectural elements and services catalogue.

II. Layered Architectural Ambient

Developing complex software systems is considered easier when a proper software architecture is proposed to describe major elements, global structure and design decisions [14]. Architectural environment in this paper is based on SOA Reference Model (SOA RM) and on SOA Reference Architecure (SOA RA).

SOA RM defined the vocabulary of SOA elements and its context relationships. SOA RM is created to semantically establish services definitions, contract description, service propagation, data model and service contracts [15]. SOA RM proposes a local basis on which reference architectures, software structure and implementation can be developed.

SOA RA is composed of a variety of models and specifications which define a logical platform for implementation. SOA RA combines SOA RM concepts with common IT architectures by using models and visualizations of common architectural domains.

Some commercial vendors of SOA solutions propose well-defined reference architecture templates for developing SOA applications. Our proposal in this research is to use only open source software. Our reference architecture has no close connection to any platform. SOA reference architecture used in this research is the one proposed by The Open Group [16], a solution often applied to various projects since 2002. The Open Group reference architecture provides an instrument to create or evaluate an architecture in terms of layers, components (building blocks), and roles to be considered in such a way to assure return of investment in technology and that the objectives are achieved.

Fig. 1 depicts the architectural ambient proposed in this research based on The Open Group SOA RA. The proposed architecture is composed of five horizontal layers (functional) and four vertical layers (non-functional). Functional layers are described as follows.

- Operational Systems provide basic infrastructure to provide SOA functions. Besides, promotes integration with legacy systems, data bases, and so on, and allows services execution.
- Software components which implement the services. These components connect (bind) the Service Contract with its implementation, and provide loose coupling between consumer and implementation.
- Services work as a Service container in which we can find all Service Contracts, including Service Tasks, Entity, Information, and so on.
- Business Process, layer responsible for composition and orchestration of services. It supports long process, and executes tasks (sequential or parallel) according to policies, business rules, and constraints.
- Consumer Interfaces, layer which deals with communication with users, giving support to different channels between users and applications. It also provides communication between applications and loosely coupling between consumer and implementation.

Non-Functional layers are described as follows.

- Integration, which allows mediation, transformation, routing and transport. Services are exposed only through this layer, which centralizes business rules and provides loosely coupling between provider and consumer. This layer is generally supported by an ESB-Enterprise Service Bus.
- Quality of Service (QoS), which captures and monitors operational metrics, assuring reliability, availability, control, scalability and security.
- Information, including data architecture, data structures (XML-Schemas) and data protocols.
- Governance, which defines SOA objectives and as-

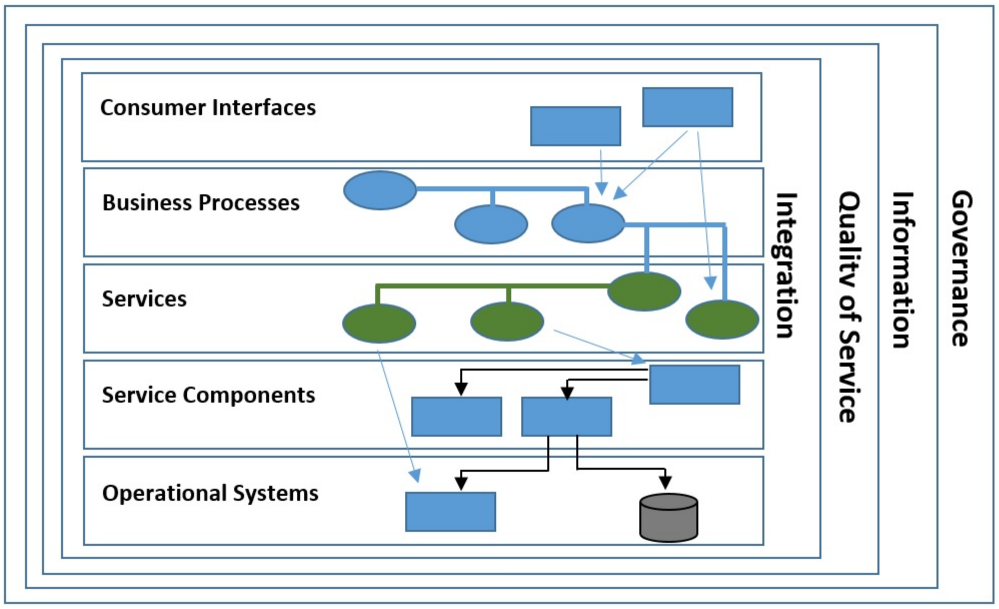

Fig. 1. Proposed Architectural Environment

sures conformity between policies and processes. It also defines solution portfolio. Governance is applied to all layers.

III. Architectural Elements

TABLE I. Architectural Elements

Analysis	Design	Implementation
Data	Relational Data Base	PostgreSQL
Systems Integration	XML Interface	Web Services
Web Services Integration	ESB	Mule ESB
Distribution layer	Object-Relational Mapping	Hibernate
Front-End	User interface communication	JSF, AJAX, PrimeFaces
Exception Handling	Exception Layer	Java
Deploy	IDE configuration for Deploy	Apache Maven
Log	Log resources implementation	Log4J

This section describes the architectural elements (Table I) that will compose the architecture, as described in the architectural ambient.

Architectural elements represent fundamental technical concepts standardized throughout the solution. They are refined during project development into three categories: analysis, design and implementation. These categories reflect the state of the architectural element in time.

Each state changes according to successive levels of detail discovered during requirements refinement. For instance, during initial analysis phases, data to be used in the EHR application are identified. Then, the correspondent design element is a relational data base. Further, correspondent implementation elements are tables implemented in a Relational Data Base, PostgreSQL in this project.

IV. Architecture Implementation View

Proposed architecture in this paper use concepts from the reference architecture depicted in Fig. 1 and architectural elements presented in Table I.

The Implementation View of the architecture is composed by Mule ESB 3.6 as Enterprise Service Bus, services catalogue, and flow of services. The view is based on the layered style, described by a UML Package diagram as depicted in Fig. 2.

Fig. 2 depicts how packages are organized. Files related to flow of services generated by mule ESB as well as web services source code are organized in package App.

Figs. 3 and 4 depict the flow of services responsible by the bus management.

Input bus is the single entry point of a message from the services bus. In this initial stage, important tasks, such as security, input log, auditing, transformation, formatting and validation are executed. After these functions, the message flow is routed for the specific service asked by

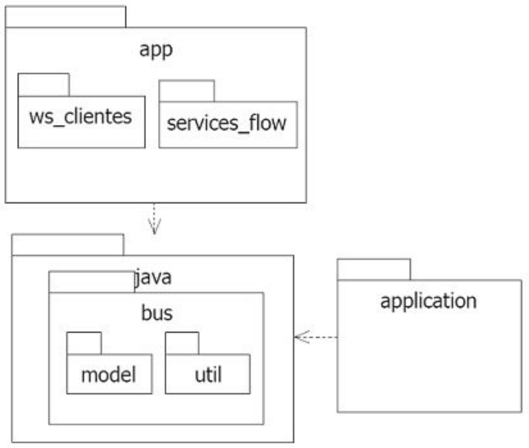

Fig. 2. ESB Package

the consumer. As depicted in Fig. 3, Mule receives objects through HTTP and executes all necessary transformations.

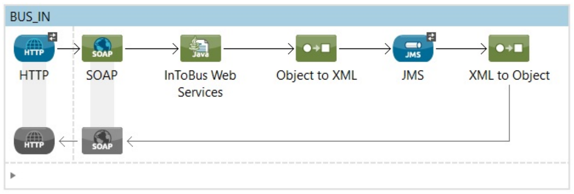

Fig. 3. Bus In

Output bus is the single output point in the services bus, being responsible for routing responses from the flow of services to the consumers. Some necessary tasks to finalize the message flow in the service are executed here, including log registry, data formatting, data transformation, validations, among others. Output bus implemented in this application is presented in Fig. 4, which makes it clear that the output bus does the inverse operation of the input bus.

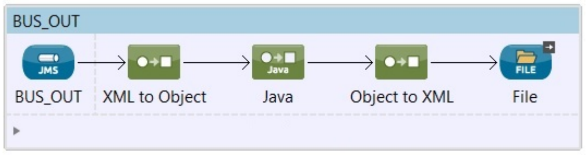

Fig. 4. Bus Out

Package Java has bus as subpackage, which has bus util and model as subpackages. Subpackage bus has source code to manipulate services flow as depicted in Figs. 3 and 4. Package model has source code related to entities that will be retrieved by client web services. Package util has source code used to manage services flow. Package applications is the package to store source code related to all applications developed based on this implementation architecture.

V. Case Study

Our proposed architecture has been considered for development of an EHR application in a public hospital. Development of the application is based on the MVC pattern, as depicted in Fig. 5, and the architectural elements presented in Table I.

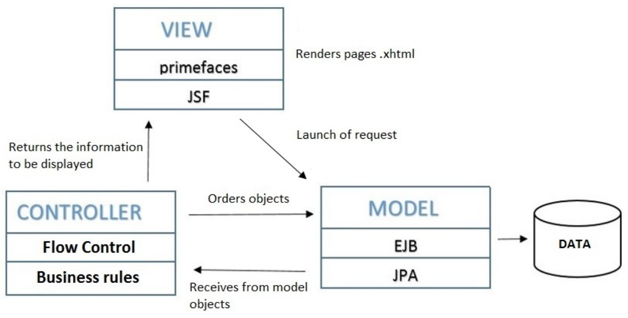

Fig. 5. MVC model for application implementation

The developed EHR application involves a number of legacy systems. Each one provides important patient information that will be consumed by the EHR. From the legacy systems which are integrated to design the EHR, Services are identified and then modeled by using the SoaML modeling language [17].

TABLE II. Service Catalogue

ID	DESCRIPTION	PROVIDER
1	Data for Login	AGHU
2	Create medical appointment	ACONE
3	Validate health card	CADWEB
4	Get patient's data	AGHU
5	List of appointments	MEDLYNX
6	List of exams	MEDLYNX
7	List of medical hospitalization	IMHOTEP
8	List of procedures	AGHU

Patient registration is performed by ACONE, which maintains also appointments schedule. CADWEB, a national health system contains registration of country citizens. Each registered citizen in CADWEB has a unique health card number, a national identifier that allows free access to public hospitals, exams, appointments, medicines and surgeries. AGHU is the hospital's main legacy system that contains important information about patients. One module of AGHU is responsible for maintaining patient personal data such as name, address, identity documents,

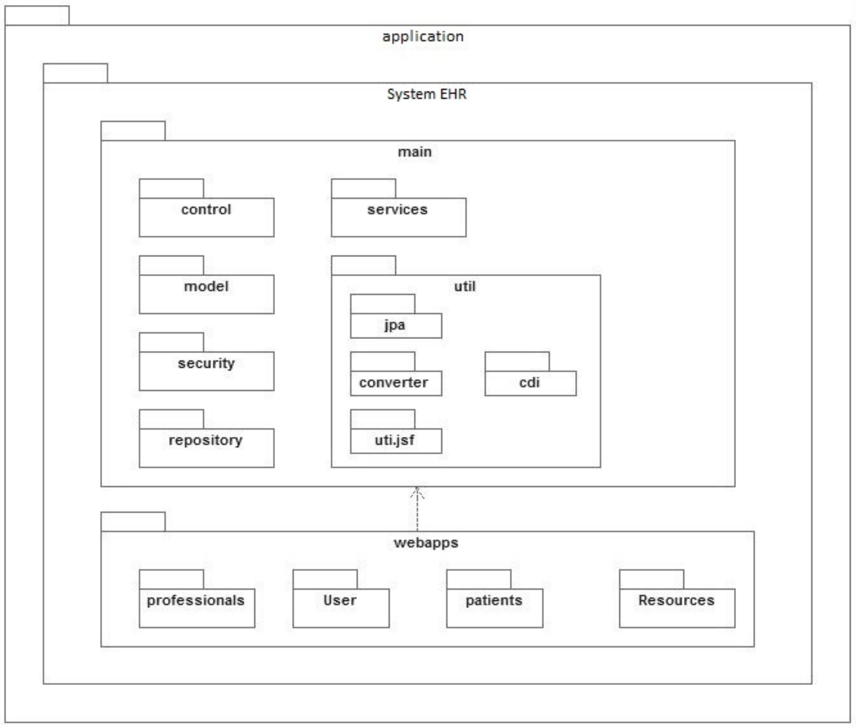

Fig. 6. Application components

and health card number. The other AGHU module controls patient exams, including storing all exams to which patients were submitted, exam results and a list of approved and pending exams. There are yet other modules that are responsible for maintaining history of attendances, hospitalization, surgery, medicines and medical procedures. Medlynx is a legacy system used to manage medical appointments and exams. Imhotep is a legacy system responsible to manage financial data and all information about products at the hospital, including medicines and other health related materials.

After analyzing all legacy systems we obtained a list of services. Services catalogue is the local where all identified services are registered. These services (Table II) can be used by the whole organization in their process behavior with the purpose of achieving a service oriented integration.

Java is the programming language used for implementation, together with frameworks Java Server Faces and Primefaces. Postgre is the relational database, together with Hibernate as JPA framework. We have also used Apache Maven for dependency injection, JBoss as application server, and Mule ESB as Enterprise Service Bus.

Fig. 6 depicts the organization of the implementation view applied for development of the EHR application. This picture details package application, as previously presented in Fig. 2.

VI. Conclusions

Developing new software systems from scratch is frequently not possible, as there are many legacy systems with stored data that need to be maintained. With this in mind, the SOA paradigm is useful to integrate legacy systems by means of web services. For this reason, considering that having a well-defined software architecture is crucial for success of complex systems, this paper proposes and describes the layered implementation view of a software architecture to develop EHR systems based on the SOA paradigm to integrate legacy systems. This view of the architecture has been used to develop an EHR in a public hospital.

The complexity of developing health information systems is well-known in the literature for many reasons, including heterogeneity of data and issues when defining the system architecture. In addition, within health information systems, the need for interoperability between legacy systems has been described in the literature, as these are normally heterogeneous systems which have a long lifespan.

Complexity of Electronic Health Record (EHR) development is addressed in this paper by proposing a layered style for implementation. The choice for interoperability in this paper is to identify services from legacy systems, providing a services catalogue, and then integrating them into an EHR application. This choice was taken as the SOA paradigm is useful to integrate legacy systems by means of web services. Future research will focus on exploring further other views of the architecture for developing the EHR application.

Acknowledgment

The authors would like to thank the Brazilian research agency CNPq (grant 445500/2014-0).

References

[1] T. Greenhalgh, K. Stramer, T. Bratan, E. Byrne, Y. Mohammad, and J. Russell, "Introduction of Shared Electronic Records: Multi-site Case Study using Diffusion of Innovation Theory," *BMJ*, vol. 337, no. 7677, pp. 1040–1044, 2008.

[2] P. Carayon, P. Smith, A. S. Hundt, V. Kuruchittham, and Q. Li, "Implementation of an Electronic Health Records System in a Small Clinic: the Viewpoint of Clinic Staff," *Behaviour & IT*, vol. 28, no. 1, pp. 5–20, 2009.

[3] T. Greenhalgh, S. Hinder, K. Stramer, T. Bratan, and J. Russell, "Adoption, Non-adoption, and Abandonment of a Personal Electronic Health Record: Case Study of HealthSpace," *BMJ*, vol. 341, no. 7782, 2010.

[4] L. Boyer, J. Samuelian, M. Fieschi, and C. Lancon, "Implementing Electronic Medical Records in a Psychiatric Hospital: a Qualitative Study," *Int. J. of Psychiatry Clinical Practice*, vol. 14, no. 3, pp. 223–227, 2010.

[5] L. Nguyen, E. Bellucci, and L. T. Nguyen, "Electronic Health Records Implementation: An Evaluation of Information System Impact and Contingency Factors," *Int. J. of Medical Informatics*, vol. 83, no. 11, pp. 779–796, 2014.

[6] S. Zimeras and A. N. Kastania, "Statistical Models for EHR Security in Web Healthcare Information Systems," *Certification and Security in Health-Related Web Applications: Concepts and Solutions: Concepts and Solutions*, p. 146, 2010.

[7] A. Sheikh, A. Jha, K. Cresswell, F. Greaves, and D. W. Bates, "Adoption of Electronic Health Records in UK Hospitals: Lessons from the USA," *The Lancet*, vol. 384, no. 9937, pp. 8–9, 2014.

[8] L. D. Xu, "Enterprise Systems: State-of-the-Art and Future Trends," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 630–640, Nov 2011.

[9] M. S. Soares and J. M. S. França, "Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review," *Journal of Software*, vol. 11, no. 4, pp. 403–417, 2016.

[10] S. Alahmari, E. Zaluska, and D. D. Roure, "A Service Identification Framework for Legacy System Migration into SOA," in *IEEE Int. Conf. on Services Computing*, July 2010, pp. 614–617.

[11] A. Ryan and P. W. Eklund, "A Framework for Semantic Interoperability in Healthcare: A Service Oriented Architecture based on Health Informatics Standards," in *Proc. of MIE2008, The XXIst Int. Congress of the European Federation for Medical Informatics*, 2008, pp. 759–764.

[12] F. G. Silva, J. S. S. de Menezes, J. de S. Lima, J. M. S. França, R. P. C. do Nascimento, and M. S. Soares, "An Experience of using SoaML for Modeling a Service-Oriented Architecture for Health Information Systems," in *Proc. of the 17th Intern. Conf. on Enterprise Information Systems*, 2015, pp. 322–327.

[13] O. Iroju, A. Soriyan, I. Gambo, and J. Olaleke, "Interoperability in Healthcare: Benefits, Challenges and Resolutions," *Int. J. of Innovation and Applied Studies*, vol. 3, no. 1, pp. 262–270, 2013.

[14] G. Booch, "The Economics of Architecture-First," *IEEE Software*, vol. 24, no. 5, pp. 18–20, 2007.

[15] OASIS. (2006) Reference Model for Service Oriented Architecture.

[16] O. Group, "SOA Reference Architecture," Tech. Rep., 2011.

[17] OMG, "Service Oriented Architecture Modeling Language (SoaML) Specification," Tech. Rep., 2012.