Software Architecture Modeling for Legacy Health Information Systems Using Polyglot Persistence and Archetypes

André Magno Costa de Araújo[1,*], Valéria Cesário Times[1] and Marcus Urbano da Silva[1]
[1] Center for Informatics, Federal University of Pernambuco, Recife, Brazil
e-mail:{amca,vct,mus}@cin.ufpe.br

Carlos Andrew Costa Bezerra[2]
[2] Software Engineering Department, Recife Center for Advanced Studies and Systems, CESAR
Recife, Brazil
e-mail: andrew@r2asistemas.com.br

***Abstract*— Electronic Health Record (EHR) data management in a Health Information System (HIS) has traditionally been done using a single database model. Due to the heterogeneity of such data, this practice increases the complexity in HIS development. This article presents a software architecture for a legacy HIS, which improves data management by using polyglot persistence to decentralize data storage into heterogeneous databases (i.e., relational and NoSQL). In addition, we have developed a tool to dynamically create NoSQL data schemas and Graphical User Interfaces (GUI) using a health informatics standard called archetype. The tool aims to build new functionalities in a legacy HIS using archetype-based EHR specifications imported and customized by the health professionals, thus reducing their dependence on a software team. We validated the proposed solution in a local institution, modeling a new software architecture, creating a NoSQL data schema for heterogeneous data storage and GUIs using archetypes.**

Keywords-Archetypes; Database related software; Software Architecture; E-health related software.

I. INTRODUCTION

Health information systems (HIS) are normally built using a single data model to store the various data types (i.e., structured and unstructured data) that make up the Electronic Health Record (EHR) [1]. The variety of data types is justified by the large number of subsystems in the HIS, such as the text from a medical prescription, hierarchical data of a laboratory exam or images used in diagnostic.

Because data heterogeneity often requires the development of solutions that go beyond the capability of a given database model [2], the use of a single data model is limiting at best. For example, representing hierarchical data structures in a relational database requires complex data retrieval sentences that can compromise the performance of an application. In addition, the creation of programming routines to ensure data referential integrity in NoSQL models increases the complexity in HIS development.

Another recurring problem in HIS development is the lack of uniformity in data modeling for attributes that define the EHR and terminologies that give a semantic meaning to clinical data [3][4]. In this context, it is common for software companies to use their own standards when modeling EHR requirements, which in turn hinders data exchange between health applications [5].

To address the issues posed by the use of a single data model and the heterogeneity of EHR data, the concepts of polyglot persistence and archetypes are used in this paper. Polyglot persistence is the use of different data models to deal with different storage needs [6], such as the use of a relational database to store structured data and NoSQL for unstructured and frequently changing data. An archetype is a health informatics standard proposed by the openEHR foundation to standardize EHR data attributes, terminologies and constraints [7]. Among other advantages, it allows health professionals to specify EHR requirements and seamlessly share data with other health sectors and organizations [8].

This paper specifies a software architecture for a legacy HIS applying the concept of polyglot persistence to improve data management in the healthcare sector. In addition, we have developed a tool to dynamically create NoSQL data schemas and Graphical User Interfaces (GUI) using archetypes. The tool aims to build new functionalities in a legacy HIS using archetype-based EHR specifications made by health professionals, thus reducing their dependence on a software team. To validate the solution proposed herein, we modeled a new software architecture for a legacy HIS and evaluated its dynamic generation of NoSQL data schemas and GUIs.

The sections of this article are organized as follows: Section II contextualizes the basic concepts used in this work, while Section III presents and discusses the proposed software architecture for health applications. Section IV demonstrates the generation of data schemas and GUI using the proposed tool, while the final considerations and future work are found in Section V.

II. BASIC CONCEPTS AND RELATED WORKS

This section contextualizes the basic concepts used in the development of this work and provides an analysis of main related works.

A. Archetypes

The openEHR software architecture for HIS aims to develop an open and interoperable computational platform for the healthcare sector [9]. It separates generic EHR structural information and patients' demographics from the constraints and standards associated with clinical data. An archetype consists of a computational expression based on a reference model and is represented by domain constraints and terminologies [9] (e.g., data attributes of a blood test).

 ISBN: 978-1-61208-590-6

Templates are structures used to group archetypes and allow their use in a particular context of application. They are often associated with a graphical user interface.

Dual modeling is the separation of information and knowledge in HIS architecture. In the proposed approach, the components responsible for modeling EHR clinical and demographic data are specified through data structures composed of data types, constraints and terminologies.

In an archetype, the specification of data attributes is achieved through data entry builders named generic data structures. Such structures allow the representation of EHR heterogeneous data through the following types: ITEM_SINGLE, ITEM_LIST, ITEM_TREE and ITEM_TABLE.

ITEM_SINGLE models a single data attribute, such as a patient's weight, height and age. ITEM_LIST groups a set of attributes in a list, such as a patient's address. ITEM_TREE specifies a hierarchical data structure that is logically represented as a tree. It can be used, for instance, to model a patient's physical or neurological evaluations. Finally, ITEM_TABLE models data elements by using columns for field definition and rows for field value, respectively. Each attribute of a structure is characterized by a type of data and can have a related set of associated domain restrictions and terminologies. The terminologies give semantic meaning to clinical data and can be represented as a set of health terms defined by a professional.

B. Polyglot Persistence

Polyglot Persistence is the use of different data storage approaches to deal with different storage types and needs [6]. The core idea is to store structured data through a relational approach, while semi-structured or unstructured data is stored in NoSQL data models. Figure 1 shows how the different types of healthcare data can be stored using polyglot persistence.

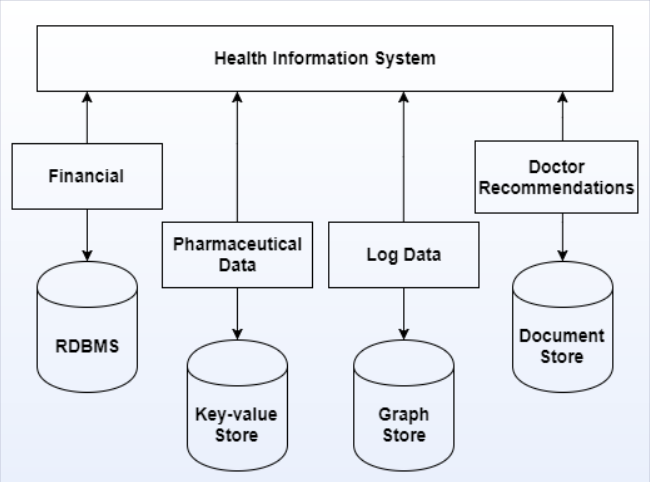

Figure 1. Example of multi-model storage in the healthcare sector.

As shown in Figure 1, polyglot persistence offers different data models such as Relational, Key-value store, Document store and Graph Store, among others.

A relational database is a set of tables containing data arranged in predefined categories. Each table contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns [10].

NoSQL data models differ from more traditional approaches and offer better support for non-conventional data storage and flexibility when creating or altering a data schema [11]. The key-value store saves information in a table with rows, keys for description and a value field. The document store creates document sets which are collections of fields to be displayed as a single element, a list or nested documents. A graph store uses nodes, relationships and properties to store information. The node represents the vertices in a graph, the relationships, its edges and the properties, the attributes.

C. Related Works and Motivation

Polyglot persistence has been applied in a variety of applications, such as IBM's auto scaling PaaS architecture [12], source-code-based data schema identification tools [13] and the re-engineering of legacy systems for heterogeneous databases [6]. To minimize the rigidity caused by relational data schema and provide support to the continuous data requirement changes which commonly occur in legacy HIS, Prasad and Sha [14] specify an architecture and a HIS prototype that allows polyglot persistence and improves health data management in a legacy application. Similarly, Kaur and Rani [6] specify a polyglot storage architecture to store structured data in a relational database (PostgreSQL), while two NoSQL databases (MongoDB and Neo4j) store semi-structured data such as laboratorial exams and medicine prescriptions. Nevertheless, neither solutions use archetypes to standardize EHR data attributes and terminologies.

Recent studies based on openEHR specifications include EHR construction using and customizing archetypes [15], Computer-Aided Software Engineering (CASE) development tools for data schema creation [16] and a study on development patterns for healthcare computing [17]. However, the use openEHR archetypes to build heterogeneous data schema, store and standardize EHR data is an open issue found in the state-of-art.

III. PROPOSED SOLUTION

This section discusses the main problems found in legacy HIS management, describes the proposed software architecture using polyglot persistence and how GUIs and data schemas are dynamically generated using archetypes.

A. Legagy Health Information System

Before implementing the proposed solution, we carried out field observation in a local health institution located in northeastern Brazil, where patient care activities are registered in a HIS, including hospitalization, prescription records and laboratory exams. To maintain and develop new HIS functionalities, the institution has a team of eight programmers.

 ISBN: 978-1-61208-590-6

Analyzing HIS implementations carried out in the last 12 months, we found that most requests were related to patient care functions such as prescription, medical history, test results, reports, etc. Meanwhile, structured data such as supply requests, financial and management requirements had suffered few alterations in the same period.

As shown in Figure 2, the legacy HIS architecture is known as Three-Tier. The client layer is responsible for designing the GUI of each feature, while the business logic layer groups and organizes all the application source code. The database access layer contains the classes that perform data persistence in the DBMS. For application development and maintenance, the following technologies are used: Oracle 11g relational database for data storage and Microsoft Asp.Net C# for GUI generation and source code implementation.

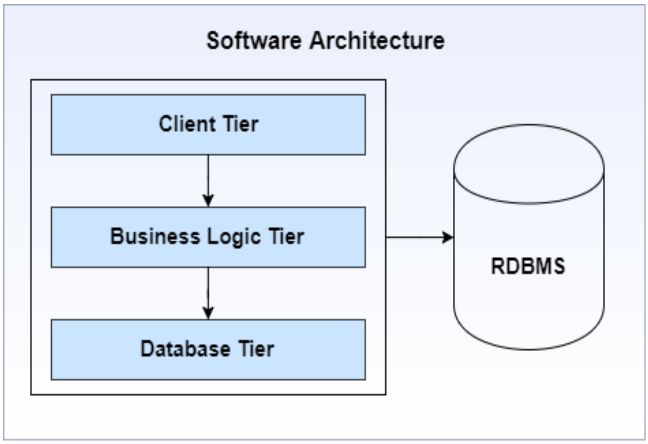

Figure 2. Legacy software architecture

The software architecture illustrated in Figure 2 represents the reality of several HISs in Brazil. In this scenario, two important problems arise: EHR data heterogeneity is represented in single data model and the reliance on a team of programmers for development and maintenance.

B. *Modeling a New Software Architecture*

The main motivation for the proposed solution is to improve data management by providing a software architecture which uses polyglot persistence to store EHR data in heterogeneous databases and openEHR archetypes to dynamically build HIS features.

Considering the scenario described in Section III-A, we used the following approach in designing the proposed software architecture; structured data which is rarely altered is stored in a relational database, while constantly changing data which requires a flexible data schema is stored in a NoSQL database (Figure 3). We developed a tool capable of reading archetypes to generate new functionalities because their very purpose is to enable health professionals to specify EHR requirements, thus minimizing their dependence on programmers.

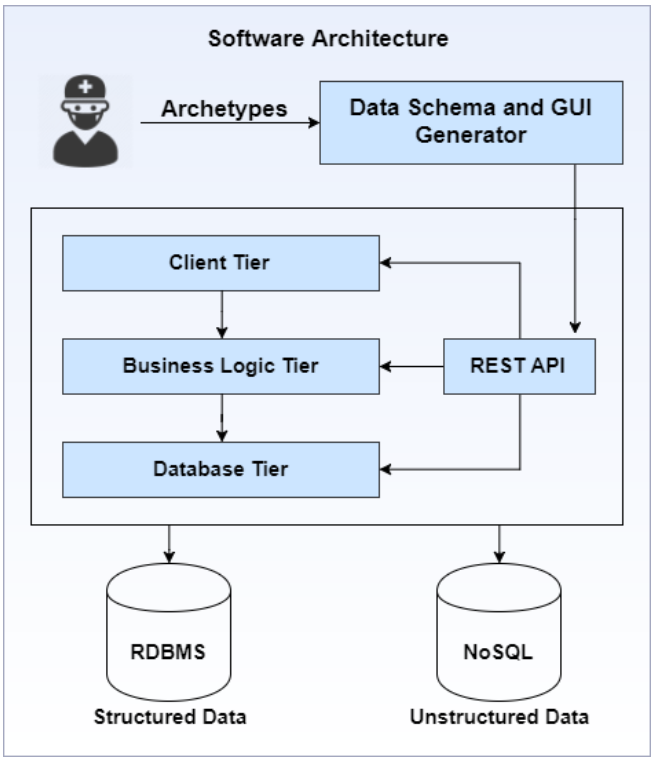

Figure 3. A new software architecture using polyglot Ppersistence and archetypes

As shown in Figure 3, we maintained the three-tier architecture (client, business logic and database) but decentralized EHR storage using two database models (relational and NoSQL). Through a REST API, we extracted data attributes, terminologies and constraints from archetypes and generated GUIs and data schemas at runtime. Because the GUIs generated by our tool use the same web-based technology as the legacy HIS, we inserted the GUIs into the HIS using frames.

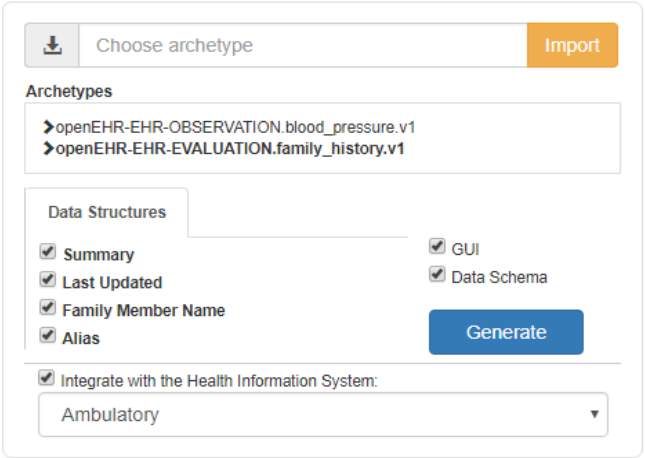

Figure 4. Main features of the developed tool

Figure 4 shows the main functionalities of the tool proposed in this article, while Figure 5 depicts a use case with

the following features: i) archetype reading, ii) archetype element selection for GUI and data schema generation, and iii) GUI integration into the legacy application.

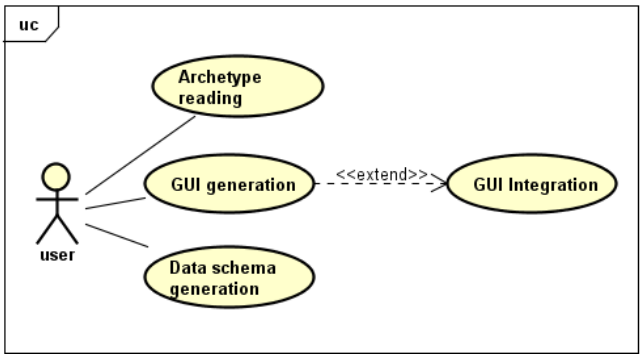

Figure 5. Use case of the developed tool

The developed tool allows one to import and map the archetypes that will be used to generate GUI and data schema. When importing an archetype, the tool enable the user to choose which elements will be part of the data schema and manage the GUI elements by adding, removing or disabling fields. Such elements can be modified at a later time. In this case, the tool will automatically extend the database schema created.

C. Graphical User Interface and Data Schema Generation

The *Generator* component shown in Figure 3 extracts from the imported archetype the attributes that define the EHR, the health care terminologies and vocabulary that give a semantic meaning to the clinical data, as well as constraints specified in the attributes.

```
Code
{
  "Archetype": {
    "DataAttributes": [
      {
        "ArchetypeID": 2,
        "Code": "at0026",
        "Label": "Last Updated",
        "Description": "The date this
        Family History Summary was last updated.",
        "DataType": "DV_DATE_TIME"
      }
    ],
    "Terminologies": [
      {
        "ArchetypeID": 2,
        "Code": "at0002",
        "Label": "Summary"
      }
    ],
    "Constraints": [
      {
        "ArchetypeID": 2,
        "Code": "at0026",
        "Type": "DateTime"
      }
    ]
  }
}
```

Figure 6. A REST API Example

With the extracted elements, the *REST API* executes the following tasks: i) transform data attributes (i.e., text, ordinal, boolean, count, quantity, date and time) into data entry fields in the GUI, ii) use constraints extracted from archetypes as data entry validation mechanisms (e.g., range of values, data type constraints), and iii) provide the terminologies extracted from archetypes to give a semantic meaning to their respective GUI fields. As the Slot type does not represent a data entry attribute in an archetype, the tool did not consider this data type to generate GUI. Figure 6 shows in JSON format the elements extracted from archetypes in the REST API.

Using the extracted elements from archetypes, the NoSQL data schema generation is performed as follows: A routine creates a document database and a set of collections to separately store the data attributes, terminologies and constraints. It then inserts archetype elements in their respective collections.

A loop scans the list of archetype elements and adds the code, description and data type in the attribute collection. To maintain the relationship between elements, attribute reference and constraint description codes are stored in the constraint collection. Finally, in case the data attribute has one or more terminologies, the attribute reference code is stored in its collection along with the list of terminologies found.

IV. RESULTS

In this section, we show GUI and data schema generation using archetypes. The tool was installed on Microsoft's Azure cloud solution and an ArangoDB database was used to generate the data schema exemplified in this evaluation. Both archetypes used in this example are shown in Figure 7 and are available in the openEHR repository [18].

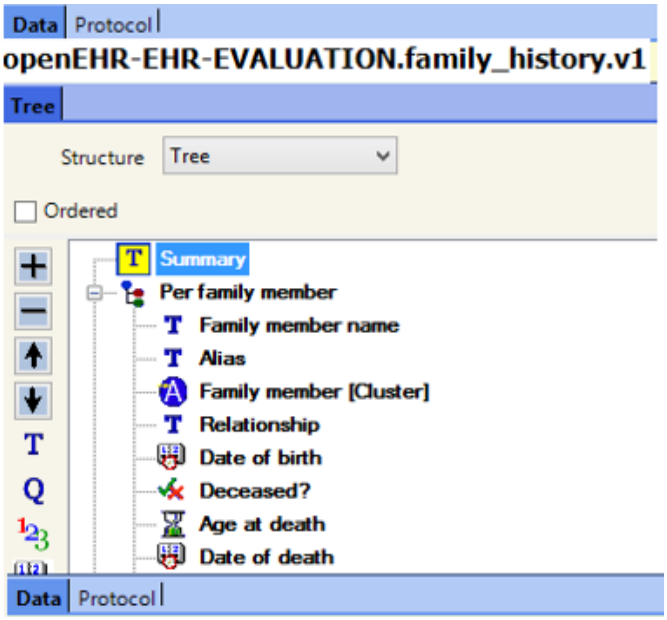

Figure 7. Family history Archetype

The family history archetype models information about significant health-related problems in genetic and non-genetic family members, both alive and deceased. The blood pressure archetype describes systemic arterial blood pressure from any measurement method or physical location.

In order to generate the GUIs and data schema, we import the two archetypes. The GUI generated from the family history archetype can be seen in Figure 8.

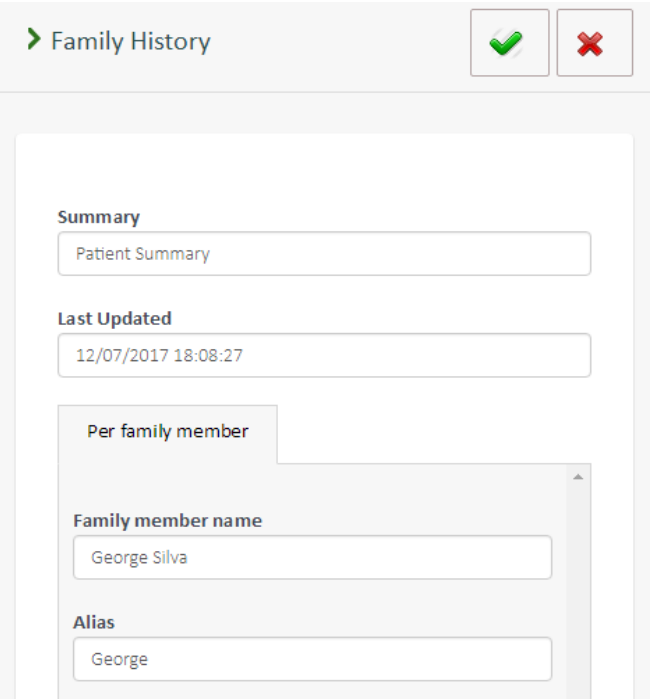

Figure 8. GUI generated from the family history archetype

The tool uses the same elements extracted from the archetype (i.e., data attributes, terminologies, and constraints) to generate both the NoSQL data schema and the GUI. In addition, each generated GUI provides resources for the end user to insert, update, delete, and query data.

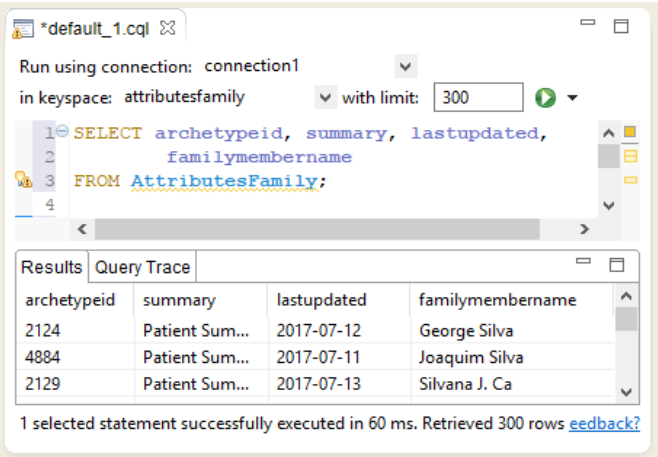

Figure 9. Data schema generated using the family history archetype

Figure 9 shows a data set stored in the database created by the tool. The data entered in the GUI is stored into the database by a REST API. GUI components can be enabled or disabled even when the GUI is in use in the HIS, triggering the data schema to dynamically change.

V. CONCLUSION AND FUTURE WORKS

In this paper, we presented a software architecture to improve Electronic Health Record data management in a legacy Health Information System using polyglot persistence to decentralize its data storage into two database models (i.e., relational and NoSQL). Using a developed tool, we extract attributes, terminologies and constraints from archetypes and use them to generate a Graphical User Interface and data schemas at runtime. The use of archetypes allows users to create new HIS functionalities without the need of a software development team.

In this paper, we limited the scope of research to data schema and GUIs generation. A forthcoming work will address privacy, performance and security issues by presenting an algorithm to encrypt EHR data on a cloud service or local storage. In addition, we intend to implement a Health Level 7 (HL7) messages system to exchange data between health applications.

ACKNOWLEDGMENT

This work was partially supported by Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), under the grants APQ-0173-1.03/15 and IBPG-0809-1.03/13.

REFERENCES

[1] V. Dinu and P. Nadkarni, “Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases,” International Journal of Medical Informatics, pp. 769-779, 2007.

[2] C. C. Martínez, T. M. Menárguez, B. J. T. Fernández, and J. A. Maldonado, “A model-driven approach for representing clinical archetypes for Semantic Web environments,” Journal of Biomedical Informatics, pp.150–164, 2009.

[3] S. Garde, E. Hovenga, J. Buck, and P. Knaup, “Expressing clinical data sets with openEHR archetypes: A solid basis for ubiquitous computing,” International Journal of Medical Informatics, pp. 334–341, 2007.

[4] B. Bernd, “Advances and Secure Architectural EHR Approaches,” International Journal of Medical informatics, pp. 185-190, 2006.

[5] K. Bernstein , R. M. Bruun, S. Vingtoft, S. K. Andersen, and C. Nøhr, “ Modelling and implementing electronic health records in Denmark,” International Journal of Medical Informatic, pp. 213-220, 2005.

[6] K. Kaur and R. Rani, “Managing Data in Healthcare Information Systems: Many Models, One Solution,” IEEE Computer Society, pp. 52-59, 2015.

[7] J. Buck, S. Garde, C. D. Kohl, and G. P. Knaup, “Towards a comprehensive electronic patient record to support an innovative individual care concept for premature infants using the openEHR approach,” International Journal of Medical Informatics, pp. 521-531, 2009.

[8] L. Lezcano, A. S. Miguel, and S. C. Rodríguez, “ Integrating reasoning and clinical archetypes using OWL ontologies and

 ISBN: 978-1-61208-590-6

SWRL rules," Journal of Biomedical Informatics, pp.1-11, 2010.

[9] T. Beale, "Archetypes: Constraint-based domain models for future-proof information systems, " Eleventh OOPSLA Workshop on Behavioral Semantics: Serving the Customer, pp. 16-32, 2002

[10] R. Elmasri and S.B. Navathe, Fundamentals of Database Systems, Addison-Wesley, 6 ed., 1994.

[11] K. k. Lee, W. Tangb, and K. Choia, "Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage," Computer Methods and Programs in Biomedicine, pp. 99-109, 2013.

[12] S. R. Seelam, P. Dettori, P. Westerink, and B. B. Yang, "Polyglot Application Auto Scaling Service for Platform As A Service Cloud," IEEE International Conference on Cloud Engineering, pp. 84-91, 2015.

[13] M. Ellison, R. Calinescu, and R. Paige, "Re-engineering the Database Layer of Legacy Applications for Scalable Cloud Deployment," IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 976-979, 2014.

[14] S. Prasad and N. Sha, "NextGen Data Persistence Pattern in Healthcare: Polyglot Persistence," Fourth International Conference on Computing, Communications and Networking Technologies, pp. 1-8, 2013.

[15] M. B. Späth and J. Grimson, "Applying the archetype approach to the database of a biobank information management system," International Journal of Medical Informatics, pp. 1-22, 2010.

[16] D. Georg, C. Judith, and R. Christoph, "Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience," BMC Medical Informatics and Decision Making, pp. 1-12, 2013.

[17] E. Marco, A.Thomas, R. Jorg, D. Asuman, and L. Gokce, "A Survey and Analysis of Electronic Healthcare Record Standards," ACM Computing Surveys, pp. 277–315, 2005.

[18] Clinical Knowledge Manager. Available from: http://openehr.org/ckm/ 2017.08.10.

 ISBN: 978-1-61208-590-6