pyEHR: a scalable clinical data management toolkit for biomedical research projects

Luca Lianas, Francesca Frexia, Giovanni Delussu, Paolo Anedda, Gianluigi Zanetti
CRS4
Pula, CA, Italy
firstname.lastname@crs4.it

***Abstract*—In this work we describe pyEHR, a new toolkit for building scalable clinical/phenotypic data management systems for biomedical research applications. The toolkit uses openEHR formalisms to guarantee the decoupling of clinical data descriptions from implementation details, and NoSQL technologies, or next-generation SQL ones, to provide scalable storage back-ends.**

I. Introduction

Next-generation sequencing and other high throughput technologies are rapidly transforming life sciences [1], [2]. Their use is now routine in biology labs and is very quickly expanding to clinical research [3] and applications [4]. This transformation has led to an abrupt transition towards a data-intensive domain where sophisticated computational analysis is essential to extract biologically relevant information from the raw data [5]. Consequently, a great effort has been made to develop scalable computational tools able to cope with the current data load and the foreseen, much larger, one that is expected to arise due to the increasing use of these technologies in large-scale clinical studies. However, since most biomedical research is focused on finding correlations between measured biomolecular signals and observed phenotypic traits, the development of novel algorithms and data handling techniques needs to be complemented by robust, scalable, computable, uniform and implementation-independent descriptions of phenotypic data [3]. Here, scalability is meant both with respect to the sheer size of the data and to the evolution of their structure and type. The latter issue is typical of longitudinal biomedical studies, where multiple, heterogeneous data types can become necessary within the time span of the project. Ideally, common solutions based on ad-hoc database tables should be replaced by computable formalisms for the meta-description of structured clinical/phenotypic records. Such systems should easily support operations such as aggregations on sophisticated profile descriptions across all the available records, as well as in-depth navigation of all data related to a specific study participant.

In this paper we describe our ongoing work on pyEHR, a toolkit for the creation of clinical/phenotypic data-management systems for biomedical research that are scalable with respect to the evolution and heterogeneity of clinical data structures and to data volumes compatible with regional-scale studies. We use openEHR [6] — a computable formalism for the meta-description of structured clinical records — as a systematic approach for handling data heterogeneity. Scalability with respect to dataset size, on the other hand, is achieved through a multi-tier architecture with interfaces for multiple data storage systems. Currently, we provide drivers for MongoDB [7] and ElasticSearch [8].

Our motivation — as well as our use cases — for the development of pyEHR comes from our direct experience in providing computational support to a wide range of biomedical research projects, including large-scale genome sequencing studies [9]–[11] and safety assessments of novel gene therapy approaches [12].

The remainder of the paper is structured as follows. Section II provides a brief overview of openEHR, while section III delineates pyEHR's architecture. Section IV illustrates related work on open source openEHR implementations. Lastly, section V is dedicated to the conclusions.

II. Computable formalisms for the meta-description of structured clinical records

There is a large body of research and practical experience in computable formalisms for the meta-description of structured clinical records, HL7v3 RIM and CEN 13606/openEHR being two leading examples. openEHR uses the Archetype Definition Language (ADL) to express constraint-based clinical models, or *archetypes*, and the Archetype Query Language (AQL) to search and retrieve clinical data from archetype-based EHRs. Both languages are designed to express clinical facts and the queries upon them at the semantic level (captured within the archetype description) rather than at the data instance level, and are completely neutral with respect to system implementation and environment. This computability at the semantic level makes it possible, in principle, to guarantee the development of clinical health systems that are robust and future-proof, since their concrete implementation maps to their formal description.

A considerable challenge in health informatics is the harmonization of competing clinical perspectives across multiple clinical sub-domains. For instance, a simple "Family History" archetype, developed for use in general family practice, may lack key attributes expected in more specialist settings, where a full family "pedigree" is required. Nevertheless, both must interact at some level, and therefore be based on a shared model. The "maximal, inclusive" dataset paradigm used in openEHR archetypes addresses this difficulty by including all attributes

978-1-4799-6644-8/14/$31.00 ©2014 IEEE

required to model a well-bounded concept such as "Family History". A further constraints layer — openEHR templates — is then applied to define an agreed minimal standard applicable to a specific care setting, or shared-care environment. This template layer also provides a static, computable formalism from which traditional software outputs can be derived, such as program code and GUI generation, message schema or data dictionary definitions. Although the formalism is terminology-agnostic, external reference terminologies such as ICD-x, LOINC and SNOMED-CT generally play a significant role in the construction of archetype-based semantic models. However, much work remains to be done to effectively employ such terminologies, particularly in the context of terminologies such as SNOMED-CT that rely heavily on post-coordination.

III. Architecture

pyEHR consists of three main modules: the *Data Management System* (Sec. III-A), the *Information Management System* (Sec. III-B) and the *Query Engine* (Sec. III-C); the relationships between these modules and the databases are shown in Figure 1. The software has been written mostly in Python, with services exported through a REST architecture, similarly to the approach adopted to develop the LiU EEE system [13]; in our case, however, particular attention has been devoted to ensuring a degree of flexibility capable of satisfying the scalability requirements described in the previous sections. Specifically, the Data Management System and the Query Engine have been designed to support multiple data management technologies via pluggable drivers, thus making it straightforward to develop extensions for the framework, in order to to achieve the goals of scalability (vertical, horizontal or both) depending on what are the needs of the system that must be implemented.

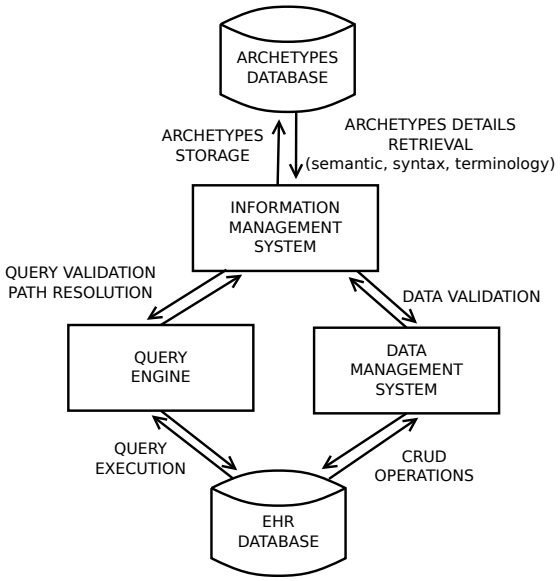

Fig. 1. pyEHR architecture: modules and their interactions

A. Data Management System

The Data Management System (DMS) handles data storage and retrieval. Clinical data records, even when expressed according to a standard like openEHR, can be represented in several different formats. Entity-Attribute-Value [14], for instance, is a good choice for table-based storage systems (e.g., SQL databases, key-value NoSQL databases, HDF5 files); other popular choices include XML [15] and document-oriented databases [16].

To avoid tying our system to any specific storage technology, we split this component into two layers: a service-oriented API for managing clinical data and a multi-driver interface that supports multiple data back-ends (Fig. 2).

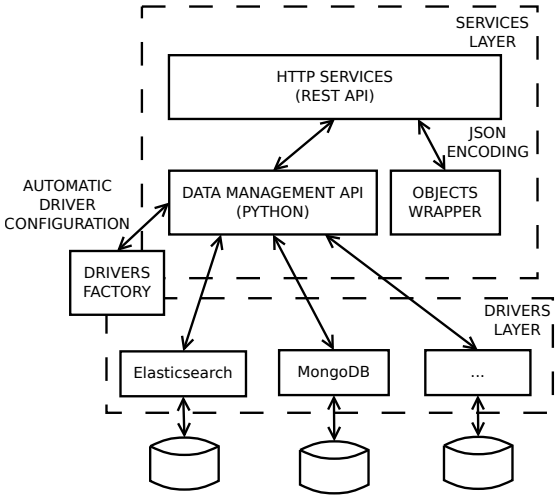

Fig. 2. Architecture of the Data Management System.

The services layer provides the required abstraction for managing data irrespective of the underlying driver. Both clinical and patient records are wrapped into specific objects that all exposed callbacks expect to handle. Clinical records, in particular, are represented as documents in the ADL format, with structures that match those described in the openEHR archetypes. Wrapper objects are converted to and from JSON as needed, so that they can be handled by the REST API.

The drivers layer has been designed to ensure transparent and uniform access to multiple storage technologies, such as the NoSQL-oriented solution proposed by Atzeni et al. [17]; this is achieved by defining a common interface that all drivers must implement and a factory class that performs driver initialization based upon a given configuration. Currently we support two NoSQL database management systems: MongoDB [7] and ElasticSearch [8]. Both systems, having been designed to handle hierarchical sets of key-value items, are easily adaptable to the document-like structure of openEHR data.

Each driver has the following responsibilities:

- manage connections and disconnections to the database;
- provide full CRUD (create, read, update, delete) support;
- handle queries (Sec. III-C);

- encode/decode data to/from the wrapper objects defined in the services layer, automatically converting any special characters;
- create data structures such as SQL tables or folders;
- split or join records as required by the underlying storage system.

B. Information Management System

The Information Management System (IMS) provides openEHR-related features. The DMS, on the other hand, is designed to be agnostic with respect to openEHR: clinical records are stored in an ADL-like format, but no semantic or syntactic check is performed at this level. The IMS acts as an independent framework that exposes a set of services, which implement features like data and query validation. This component is currently under development; Fig. 3 shows the intended architecture.

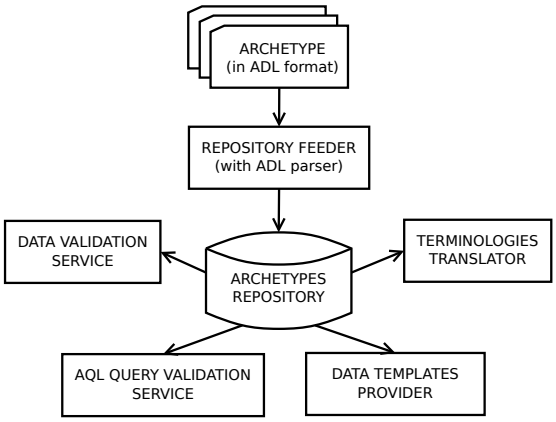

Fig. 3. Architecture of the Information Management System.

The framework's core component is the *Archetypes Repository*, where normalized version of the archetypes are stored in a graph-oriented database, implemented using Neo4j [18]; inclusion relationships between archetypes are represented by graph edges. Access to the repository is mediated by the *Repository feeder*, which parses ADL documents using the official openEHR Java libraries [19]. In addition, the framework provides the following services:

Data Validation Service: performs semantic and syntactic validation of clinical records to be stored in the repository.

AQL Query Validation Service: validates AQL queries, ensuring that any specified paths are correct. This service also makes use of information stored within the graph database to resolve all possible paths that lead to the retrieval of a given record (this ambiguity is due to the fact that an archetype can be saved as stand-alone or as a member of another archetype).

Data Templates Provider: provides empty document templates that can be filled by external systems to populate the DMS (provided that the supplied records pass data validation). This component also provides descriptions that can be used by DMS drivers to build appropriate structures for new data types or by external applications to automatically generate components such as Graphical User Interfaces [20].

Terminologies Translator: maps openEHR paths to labels and descriptions in one of the languages defined in the ontology section of the Archetype's ADL. This component allows to present information in a human-readable format.

C. Query Engine

Effective use of the openEHR formalism requires a query engine that can leverage its archetype-based model to retrieve clinical data from EHRs. pyEHR supports the Archetype Query Language (AQL), formerly known as EHR query language (EQL) [21]. Its main advantage is that it allows to express queries at the archetype level (i.e., the semantic level), rather than at the data instance level, thus allowing to share queries across system or enterprise boundaries.

The main advantage of the AQL is that it allows to express the queries at the archetype level, i.e. semantic level, other than at the data instance level. This is the key in achieving sharing queries across system boundaries or enterprise boundaries. Its main features are:

- the utilization of openEHR archetype path syntax in the query expression;
- the utilization of containment mechanisms to indicate the data hierarchy;
- the utilization of ADL-like operator syntax, such as matches, exists, in, negation;
- a neutral expression syntax. AQL does not have any dependencies on the underlying RM (Reference Model) of the archetypes. It is neutral to system implementation and environment;
- the support of queries with logical time-based data rollback.

The Query Engine consists of the following components:

- the Query Manager Service;
- the Parser;
- the DB Drivers (supporting multiple DBs);
- the Command Line Interface.

The Query Manager Service is the main component; it is responsible for instantiating the Parser and the DB Drivers, coordinating their activities and providing an interface that allows access by external process.

The Parser converts an AQL query string into an object model instance that allows to access the query's structure and data via API calls; this instance is used by other components to perform matches into the database.

A DB Driver (see Sec. III-A) act as an interface to the underlying storage system. Its main responsibility is to translate the object model instance provided by the parser into the specific format required by the storage system, execute the query and return the results as an AQL Results Set — a simple interface that represents query results as sets of columns and rows.

The Command Line Interface, implemented in Python, provides a text-based tool for executing queries and displaying the corresponding results.

The query processing workflow is structured as follows:

1) the Query Manager Service receives a query string;
2) the Parser generates an object model instance of the query, or Query Object Model (QOM);
3) the DB Driver translates the QOM into a language supported by the underlying DB;
4) the DB Driver converts the result to an AQL Result Set;
5) the Query Manager Service returns the result to the user.

IV. Related Work

The development of a comprehensive data infrastructure for the management and analysis of Electronic Health Records based upon the openEHR specifications has been pursued extensively in different contexts and with various goals. Here we provide a short overview of related work on this subject; for a more detailed discussion, see [22].

In [13], Sundvall et al. describe LiU EEE, an educational EHR environment designed to help newcomers and developers experiment with and learn about the EHR model. Liu EEE uses an HTTP-based interface to provide a comprehensive set of features for handling EHR records described as openEHR archetypes and templates, as well as retrieving data using AQL queries. Although the system does provide a rich-featured openEHR-based environment, its intended usage is for educational purpose rather than for large-scale data analysis; in particular, data records are stored as XML documents, an approach that does not scale well to population-wide queries.

ResearchEHR [23] [24], formerly known as LinkEHR, is a platform for the development and application of semantic technologies to the management of existing EHRs. The system, which allows to "describe the semantics of legacy health data in a manner independent of the particular data organization in the underlying data repositories", provides an application builder to easily create and deploy web applications starting from known archetypes, and an integration framework to fetch data from legacy EHR systems and map them to archetypes. ResearchEHR is mainly intended for health professionals, and its focus is on semantic interoperability; in particular, it does not offer research-oriented features such as population-wide queries. Moreover, interaction with the underlying database is not mediated by a sophisticated query service as described in Sec. III-C; rather, interrogations are driven directly by the graphical user interface.

Domain-specific systems that use openEHR are described in [25], which shows a solution for electronic patient records in neonatology, and in [26], which provides guidelines for chemotherapy. Both works show how to apply an openEHR approach to develop a system where patient data can be structured, stored, managed and exchanged in a safe and reliable way between different healthcare providers, and how both existing and new archetypes can be used for data description; however, they do not describe a full working system.

Several solutions have been proposed to develop EHR systems for whole countries, such as Romania [27] [28], Indonesia [29] and Brazil [30] [31].

V. Conclusions and Future Work

We have presented pyEHR, a new toolkit designed to simplify the creation of clinical/phenotypic data management systems for biomedical research. pyEHR is a modular environment that consists of three main components: the Information Management System and the Query Engine provide functionalities to store, retrieve and query data modeled using the openEHR formalism (ADL and AQL), while the Data Management System offers a flexible storage solution that employs a multi-driver layer to select among different scalable storage back-ends.

The current implementation of PyEHR includes: the Data Management System, with storage drivers for MongoDB and ElasticSearch; the Query Engine and the Information Management System, albeit without the ADL validation capability.

PyEHR is distributed as open source and is available on GitHub at https://github.com/crs4/pyEHR

Acknowledgments

One of the authors, Giovanni Delussu, has performed his activity in the framework of the International PhD in Innovation Science and Technology at the University of Cagliari, Italy.

References

[1] C. S. Pareek, R. Smoczynski, and A. Tretyn, "Sequencing technologies and genome sequencing." *Journal of applied genetics*, vol. 52, no. 4, pp. 413–35, Dec. 2011.

[2] W. W. Soon, M. Hariharan, and M. P. Snyder, "High-throughput sequencing for biology and medicine." *Molecular systems biology*, vol. 9, p. 640, Jan. 2013.

[3] C. Chute, M. Ullman-Cullere, and G. Wood, "Some experiences and opportunities for big data in translational research," *Genetics in ...*, vol. 15, no. 10, pp. 802–809, 2013.

[4] R. Simon and S. Roychowdhury, "Implementing personalized cancer genomics in clinical trials." *Nature reviews. Drug discovery*, vol. 12, no. 5, pp. 358–69, May 2013.

[5] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, June 2013.

[6] T. Beale and T. Beale, "Archetypes constraint-based domain models for futureproof information systems," 2000.

[7] MongoDB. [Online]. Available: https://www.mongodb.org/

[8] Elasticsearch. [Online]. Available: http://www.elasticsearch.org/

[9] V. Orrù, M. Steri, G. Sole, C. Sidore, F. Virdis, M. Dei, S. Lai, M. Zoledziewska, F. Busonero, A. Mulas, M. Floris, W. I. Mentzen, S. A. M. Urru, S. Olla, M. Marongiu, M. G. Piras, M. Lobina, A. Maschio, M. Pitzalis, M. F. Urru, M. Marcelli, R. Cusano, F. Deidda, V. Serra, M. Oppo, R. Pilu, F. Reinier, R. Berutti, L. Pireddu, I. Zara, E. Porcu, A. Kwong, C. Brennan, B. Tarrier, R. Lyons, H. M. Kang, S. Uzzau, R. Atzeni, M. Valentini, D. Firinu, L. Leoni, G. Rotta, S. Naitza, A. Angius, M. Congia, M. B. Whalen, C. M. Jones, D. Schlessinger, G. R. Abecasis, E. Fiorillo, S. Sanna, and F. Cucca, "Genetic variants regulating immune cell levels in health and disease." *Cell*, vol. 155, no. 1, pp. 242–56, Sep. 2013.

[10] P. Francalacci, L. Morelli, A. Angius, R. Berutti, F. Reinier, R. Atzeni, R. Pilu, F. Busonero, A. Maschio, I. Zara, D. Sanna, A. Useli, M. F. Urru, M. Marcelli, R. Cusano, M. Oppo, M. Zoledziewska, M. Pitzalis, F. Deidda, E. Porcu, F. Poddie, H. M. Kang, R. Lyons, B. Tarrier, J. B. Gresham, B. Li, S. Tofanelli, S. Alonso, M. Dei, S. Lai, A. Mulas, M. B. Whalen, S. Uzzau, C. Jones, D. Schlessinger, G. R. Abecasis, S. Sanna, C. Sidore, and F. Cucca, "Low-pass DNA sequencing of 1200 Sardinians reconstructs European Y-chromosome phylogeny." *Science (New York, N.Y.)*, vol. 341, no. 6145, pp. 565–9, Aug. 2013.

[11] G. R. Abecasis, A. Auton, L. D. Brooks, M. a. DePristo, R. M. Durbin, R. E. Handsaker, H. M. Kang, G. T. Marth, and G. a. McVean, "An integrated map of genetic variation from 1,092 human genomes." *Nature*, vol. 491, no. 7422, pp. 56–65, Nov. 2012.

[12] A. Biffi, E. Montini, L. Lorioli, M. Cesani, F. Fumagalli, T. Plati, C. Baldoli, S. Martino, A. Calabria, S. Canale, F. Benedicenti, G. Vallanti, L. Biasco, S. Leo, N. Kabbara, G. Zanetti, W. Rizzo, N. Mehta, M. Cicalese, M. Casiraghi, J. Boelens, U. Del Carro, D. Dow, M. Schmidt, A. Assanelli, V. Neduva, C. Di Serio, E. Stupka, J. Gardner, C. von Kalle, C. Bordignon, F. Ciceri, A. Rovelli, M. Roncarolo, A. Aiuti, M. Sessa, and L. Naldini, "Lentiviral hematopoietic stem cell gene therapy benefits metachromatic leukodystrophy," *Science*, vol. 341, no. 6148, p. 1233158, august 2013.

[13] E. Sundvall, M. Nyström, D. Karlsson, M. Eneling, R. Chen, and H. k. Örman, "Applying representational state transfer (REST) architecture to archetype-based electronic health record systems." *BMC medical informatics and decision making*, vol. 13, p. 57, Jan. 2013.

[14] V. Dinu and P. Nadkarni, "Guidelines for the effective use of entity-attribute-value modeling for biomedical databases," *International Journal of Medical Informatics*, vol. In Press, Corrected Proof, p. 1056, 2006.

[15] K. K.-Y. Lee, W.-C. Tang, and K.-S. Choi, "Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage." *Computer methods and programs in biomedicine*, vol. 110, no. 1, pp. 99–109, Apr. 2013.

[16] O. Schmitt and T. A. Majchrzak, "Using Document-Based Databases for Medical Information Systems in Unreliable Environments," in *Proceedings of the 9th International ISCRAM Conference*, L. Rothkrantz, J. Ristvej, and Z. Franco, Eds., 2012.

[17] P. Atzeni, F. Bugiotti, and L. Rossi, "Uniform access to NoSQL systems," *Information Systems*, pp. 1–17, Jun. 2013.

[18] Neo4j. [Online]. Available: http://www.neo4j.org

[19] OpenEHR Java Libraries. [Online]. Available: https://github.com/openEHR/java-libs

[20] T. Schuler, S. Garde, S. Heard, and T. Beale, "Towards automatically generating graphical user interfaces from openEHR archetypes." *Studies in health technology and informatics*, vol. 124, pp. 221–226, 2006.

[21] C. Ma, H. Frankel, T. Beale, and S. Heard, "EHR Query Language (EQL) – A Query Language for Archetype-Based Health Records," in *MEDINFO 2007. Proceedings of the 12th World Congress on Health (Medical) Informatics. Building Sustainable Health Systems*, K. A. Kuhn, J. R. Warren, and T. Y. Leong, Eds., vol. 129, AMIA. IOS Press, 2007, pp. 397–401.

[22] S. Frade, S. M. Freire, E. Sundvall, J. H. Patriarca-Almeida, and R. J. C. Correia, "Survey of openehr storage implementations." in *CBMS*, P. P. Rodrigues, M. Pechenizkiy, J. Gama, R. Cruz-Correia, J. Liu, A. J. M. Traina, P. J. F. Lucas, and P. Soda, Eds. IEEE, 2013, pp. 303–307.

[23] J. A. Maldonado, C. M. Costa, D. Moner, M. Men??rguez-Tortosa, D. Bosc??, J. A. Mi??arro Gim??nez, J. T. Fern??ndez-Breis, and M. Robles, "Using the ResearchEHR platform to facilitate the practical application of the EHR standards," *Journal of Biomedical Informatics*, vol. 45, pp. 746–762, 2012.

[24] M. Robles, J. T. Fernández-Breis, J. A. Maldonado, D. Moner, C. Martínez-Costa, D. Bosca, and M. Menárguez-Tortosa, "ResearchEHR: use of semantic web technologies and archetypes for the description of EHRs." *Studies in health technology and informatics*, vol. 155, pp. 129–135, 2010.

[25] J. Buck, S. Garde, C. D. Kohl, and P. Knaup-Gregori, "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*, vol. 78, pp. 521–531, 2009.

[26] R. Chen, P. Georgii-Hemming, and H. Ahlfeldt, "Representing a chemotherapy guideline using openEHR and rules." *Studies in health technology and informatics*, vol. 150, pp. 653–657, 2009.

[27] O. Stan, D. Sauciuc, and L. Miclea, "Medical Informatics System for Romanian Healthcare System," *Proceedings of the 3rd International Conference on E-Health and Bioengineering*, pp. 24–27, 2011.

[28] ——, "Electronic healthcare record according to general clinical observation file," *Proceedings of 2012 IEEE International Conference on Automation, Quality and Testing, Robotics*, pp. 503–507, May 2012.

[29] A. W. Setiawan, A. Handayani, A. D. Setiawan, G. A. P. Saptawati, A. B. Suksmono, and T. R. Mengko, "A Review of the OpenEHR Implementation in Indonesian National Health Information System: Integrated Health Post," pp. 234–236, 2009.

[30] G. M. Bacelar-Silva, H. Cesar, P. Braga, and R. Guimaraes, "OpenEHR-based pervasive health information system for primary care: First Brazilian experience for public care," in *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems U6*, 2013, pp. 572–873.

[31] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, "A cloud computing solution for patient's data collection in health care institutions," in *Proceedings of the 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine*. IEEE Computer Society, 2010, pp. 95–99.