A Framework for Supporting the Distributed Management of Big Clinical Data

Alfredo Cuzzocrea
ICARCNR and University of Calabria
Rende (CS), Italy
cuzzocrea@si.deis.unical.it

Giorgio Mario Grasso
CSECS Department
University of Messina
Messina, Italy
gmgrasso@unime.it

Andrea Nucita
CSECS Department
University of Messina
Messina, Italy
anucita@unime.it

ABSTRACT

Managing Big Data in distributed environments in a critical research challenges which has driven the attention from the community. In this context, there are several issues to be faced-off, including (*i*) dealing with massive and heterogenous data, (*ii*) inconsistency problems, (*iii*) query optimization bottlenecks, and so forth. Clinical data represent a vibrant case of Big Data, due to both practical as well as methodologies challenges exposed by such data, also dictated by tight requirements of applications which manage them. Following these considerations, in this paper we present an architecture for the storage, exchange and use of health data for administrative and epidemiological purposes, that focuses on the patient, who in a safe and easy way can make use of their data for therapeutic and research purposes. This research is being conducted as part of the CCE Project, in order to experience a new kind of storage architecture and data exchange within the field of clinical oncology.

1. INTRODUCTION

Electronic Health Record (EHR) Systems are widely considered a crucial tool for the excellence in patient care, especially in the context of chronic diseases. Nevertheless, patients often do not have full control on their clinical data, which are generated by different health centers. Moreover, collecting, storing and providing clinical data are intensive tasks for health structures, which frequently suffer for shortage of investments. Indeed, EHR systems are a crucial tool to collect data about the clinical history of a client. These data are collected during encounters with health care providers, for the prevention or during episodes of illness. It is a "health diary", through which doctors can get an overall picture of the history of people's health and can better diagnose a disorder, or decide the most appropriate therapy.

The spread of computers and computer networks seems to be able to meet the growing need for storage, processing and transmission of clinical data in a broader context of computerization of the Health Care Systems. The management and control of health are based on the use, transmission and comparison of a large amount of heterogeneous data, information and knowledge. The need for data exchange has soared, both within a single health care facility (between the different actors and between business units), and between geographically distant facilities.

Moreover, EHR systems allow to provide services to the patients with comprehensive and reliable services, in a efficient and economical fashion. A good care, in fact, also depends on the ability to have a complete history of the patient. One of the main limitations to full scale implementation of a system for the management of clinical data of the patients is the need to build a physical infrastructure to interconnect the different operational headquarters of the public and private health care systems, which can meet the performance requirements in the transport of data and synchronization of information. This limitation is inherent to the classical approach of the client-server systems, including those based on web platforms, where the client is a simple browser. In fact, in this classical scheme, with the increase of the processing and data exchange requirements, computational resources of increasing bandwidth are mandatory. Thinking about the extremely large number of benefits that the health care system already daily provides at provincial level, it is easy to understand the huge amount of resources needed to scale-up the architectures with the spread of digital systems.

In this paper, we present the results from the CCE Project, which proposes a distributed architecture for clinical data, in which the patient is considered the focal point of the process. The patient is the real owner and holder of his/her data, and is provided with a smart card containing all the clinical reports, prescriptions or medical imagery. On the other hand, each clinical center does not have to invest in expensive data centers, because the proposed solution takes advantage of a distributed architecture, which includes the already operating local servers. As a prominent consequence, we expect that the proposed architecture would bring benefits both to patients, giving them the desired centrality in the care process, and to health administration, that could exploit the same infrastructure for better addressing health policies, for example implementing epidemiological and drug safety research.

The system for archiving and distribution of clinical data proposed here, is intended to overcome the limitations im-

BODYNETS 2014, September 29-October 01, London, Great Britain

DOI 10.4108/icst.bodynets.2014.257235

posed by traditional systems of centralized data management, and aims to provide an innovative solution that can significantly reduce infrastructure investment for the Public Administration. In addition, the patient often does not have full control of his/her health information, which typically reside in health facilities that deliver them. This raises issues of privacy. On the other hand, often the current approach consider the patient a mere user of services and not the real owner of the data. This study aims to present an architecture for the storage, exchange and use of health data for administrative and epidemiological purposes, that focuses on the patient, who in a safe and easy way can make use of their data for therapeutic and research purposes. This research is being conducted as part of the CCE Project, in order to experience a new kind of storage architecture and data exchange within the field of clinical oncology.

2. CCE: ARCHITECTURE AND FUNCTIONALITIES

The proposed architecture is an integrated system of management of health services, but also and above all, an interconnection of information systems, which can be scaled up without relevant investments in setting up new infrastructures.

As already mentioned, in the context of the CCE architecture, the patient remains the holder of his/her personal data, even though copies of the data are stored in encrypted form on remote servers for backup purposes. To do so, the patient is provided with a smart card, which will in turn record data of the health services in an encrypted way. This card, which will be presented in the next section, allows the patient to be still in possession of their clinical data, encoded according to recognized standards (HL7 and DICOM). In addition, every structure linked to the project has at its disposal a CCE server, which stores all the health services provided by the health center, encrypted and accessible only through special credentials. Data are subsequently replicated for backup, and a copy of them will reside on two servers in different associated centers. The network architecture and the features of the CCE server will be presented in Section 2.2.

2.1 CCE Client

As already mentioned, within the project CCE, the patients remain not only the owner, but also the real holder of their clinical data. It then becomes essential to use a storage device capable of holding all the clinical data of the patient. Specifically, patients who are considered by the project are cancer patients, and for this specific type of pathology they need frequent visits and diagnostic tests, not necessarily always provided by the same medical facility.

In addition, it is necessary that the data in the possession of patients reside in an encrypted storage devices, so as to ensure access only to the patient or to whom is authorized by him/her. However, the system should also provide a form of data replication to prevent data integrity issues in case of loss or malfunction of the device.

For this purpose, we have considered devices capable of storing a large amount of data (at least 8 GB), integrated with a smart card reader. In this way, the cryptography application stored on the smart card will be able to manage the file system in the flash memory USB device. Precisely, in order to ensure the safety of the device, the application was developed in JavaCard, and resides on the smart card with a PIN key access system, similar to those used in the payment cards. The application includes a private key: once the patient types the PIN code, the key will be sent to the client for the decryption of the file system. The authentication mechanism is shown in Figure 1. Here, the patient types the PIN code and, after the authentication, the file system will be available.

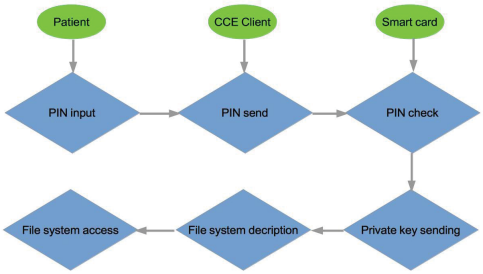

Figure 1: Client authentication mechanism.

The hospital, once the patient has typed the PIN code, gains access to patient data and adds the data on its server, digitally signing them through the device of the patient. The authentication process is shown in Figure 2.

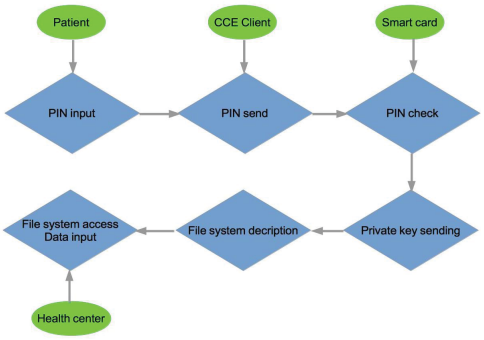

Figure 2: The authentication process for the hospital.

2.2 CCE Network

The proposed architecture uses a peer-to-peer network, where the nodes consist of servers of the health facilities involved in the project. Therefore, an essential component for the implementation of the peer-to-peer network for data exchange is the CCE Server. The CCE Server is supplied to each clinical facility involved, and has been optimized from the point of view of the implementation and management costs, in a perspective of a large scale deployment.

The CCE Server plays a key role because it is the tool through which the data of health services are stored and synchronized for backup, through the peer-to-peer network. One of the key points to ensure the efficiency of the CCE architecture is to be able to guarantee a continuous and reliable functioning of all servers on the CCE network. For this purpose we chose a specialized hardware, together with an operating system dedicated to CCE Server, which is able

to ensure adequate reliability and functionality. Starting from a Linux operating system (Ubuntu Server 12.04 LTS), specific variants have been developed that reduce the unnecessary components and implement specific functionality, such as the synchronization systems between peers in the network.

The peer-to-peer network has been realized through N2N [7], a layer two VPN that allows to take advantage of the characteristics of a P2P network instead of using the application layer; that is, users can get native IP visibility and can be reached through a local IP address assigned by the network. In addition, N2N uses encryption level two: the encryption is performed on the edge nodes using open protocols with encryption keys defined by the users. Each user can belong to multiple networks N2N (eg. communities). Moreover, N2N is transparent to NATs and firewalls.

The architecture of N2N is based on two components: supernode and edge nodes . The supernode is used by the edge nodes on startup, or to reach the nodes that are located behind a firewall. The purpose of the supernode is to keep track of registered nodes to the network and make routing of packets to those nodes that can not communicate directly. The edge node represents the client of the N2N network; virtually, every node creates a tun/tap device which is the entry point to the network N2N (Figure 3).

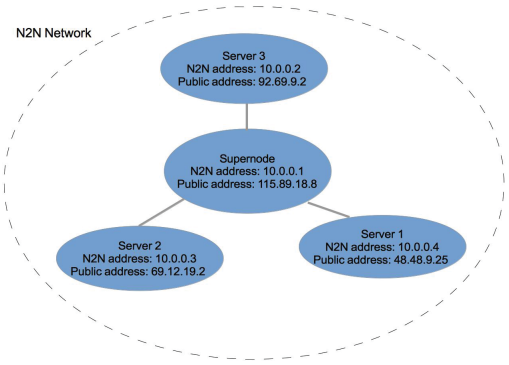

Figure 3: Topography of the N2N network.

The CCE Server software consists of the following modules, whose functionalities are summarized below: EDGE N2N and CCE BACKUP.

The EDGE N2N module is responsible of establishing a connection of all the nodes of the network. The CCE Server launches at startup the edge service of N2N, in order to create a virtual interface to access the private network N2N. In this sense, the server CCE is connected with all other CCE servers in order to be able to exchange information and to make a backup of the sessions. By using the edge service, each CCE server sees the other nodes as if they were in a local area network (LAN), thereby overcoming the limitations and obstacles of the most critical network configurations (Firewalls and NATs), and also in case the CCE Server is placed on a network that has a dynamic IP, thus avoiding the additional installation of a DDNS service (Dynamic DNS). Moreover, the role of supernode is played by the CCE Monitor, that is a node with the following tasks: i) routing packets to those servers that can not be reached directly; ii) collecting metadata about provided services for administrative or epidemiological purposes.

The CCE BACKUP service uses the private network N2N previously described in order to establish a secure connection with other CCE Servers (Figure 4). The backup software is written using the Qt Framework, and permits the use of a communication protocol based on SSL connections. The timing of the sessions between the server CCE occurs in two phases: i) discovery or election of the most updated server for each patient and the ii) updating phase. Each CCE Server, during the discovery phase, reads his patient list and, for each entry, controls in the database the list of servers that need to synchronize. Then, launches a number of threads equal to the number of servers. Each thread connects to a server to get the latest information about the patient to synchronize. Such information includes the total number of sessions for each patient and the last timestamp. Based on the information received, the CCE Server is able to determine if it is the most updated for that patient, and how many CCE server need to be updated. If the server recognizes itself as the most updated node, it launches N threads, for those servers to be updated; each of them initially alerts the Monitor, to inform it that CCE is beginning a process of synchronization, then sends the data to the different servers and then again the CCE Monitor is informed about the ending of the session synchronization process. This procedure is performed for each patient.

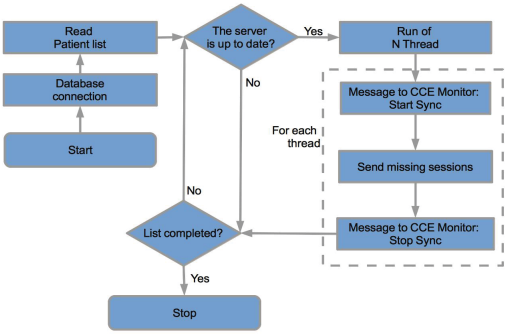

Figure 4: CCE server procedure for patient data backup.

Following this scheme, the synchronization protocol is activated (Figure 5). The first operation performed by the master server is to retrieve the sessions and their hash from the database, in order to create an XML file. Then starts the handshake phase and the master server informs the slave server with the message (CCE_CLIENT_INFORMATION:patient data). The slave server receives the message CCE_CLIEN_-INFORMATION:patient data, retrieves data from the patient and sends the message CCE_CLIENT_INFORMATION_-OK. The master server, after receiving the message CCE_-CLIENT_INFORMATION_OK, sends it to the slave server, the message XML_SESSION_file_start , the XML file of the sessions and finally when the XML file transmission ends, it sends the message XML_SESSION_files_END. The slave server receives this message sequence, and receives the XML file. When it receives the message XML_SESSION_files_-END , reads the XML file and checks which sessions related

to that patient has preserved in the repository, also checking the hash to control if there are indications of manipulation. Subsequently, it keeps in memory a list of files that is should receive, and starts with the first file, sending the message GET_SESSION_File:file name. The master server sends the message sequence SEND_files_SESSION_start: file name and SEND_files_SESSION_END:file name. The slave server receives this message sequence and the file, until it gets the message SEND_files_SESSION_END. Then, it calculates the hash and compares it with the one sent in the XML file. If the two hashes do not match, it sends the message back to the master server GET_SESSION_File:file name, for subsequent resubmission; otherwise it sends the message SEND_files_RECEIVED_OK:file name. The master server, after receiving receiving the message SEND_files_RECEIVED_OK , sends the message SEND_files_RECEIVED_CONFIRM. Finally, when the slave server receives this message, it starts to send to the master the message GET_SESSION_File:file name, to get the other files. When the list of file to be sent is terminated, the slave sends a message END_SYNC to communicate the ending of the synchronization process. The master receives the message END_SYNC, and responds with a message END_SYNC_CONFIRM, closing the socket. When the slave server receives END_SYNC_CONFIRM, also closes the socket and the SSL communication is terminated.

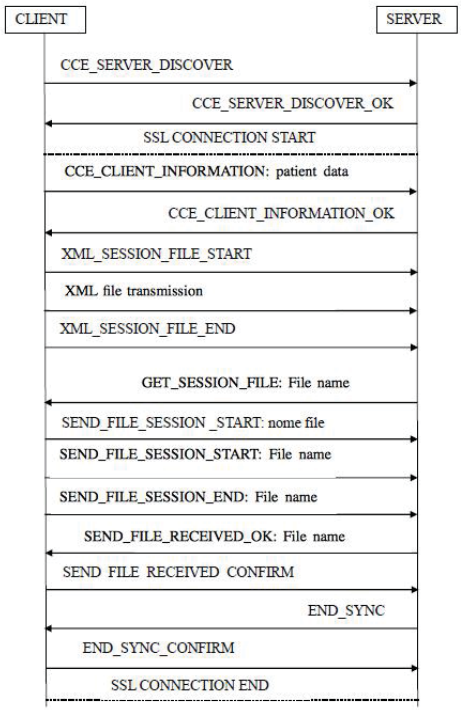

Figure 5: Synchronization protocol.

3. CCE COMMUNICATION PROTOCOL

As we already discussed, in the proposed architecture each patient is provided with a smart card, in which all the clinical data are stored. On the other hand, although each single patient holds his/her copy of the clinical data, the servers of the health facilities store their own copy of the provided clinical services. Hence, it became necessary to design the integration and communication process between the CCE client (i.e. the smart card) and the CCE server of the health structure which in turn is involved in the patient care process.

The phase of integration between the client and the server consists mainly in the use of three communication protocols, in order to be able to exchange messages and data through the local network of the hospital, in which every patient receives a health service. In Figure 6 the interaction between the two components is shown.

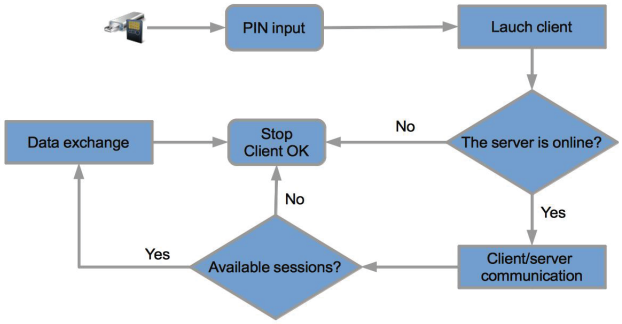

Figure 6: CCE Card-CCE Server Interaction.

In the following, we analyze in detail the three communication phases: Discovery, CCE Sync and CCE Update.

The Discovery phase is used by the CCE client in order to find a server on the local network. The CCE client, after the patient has been authenticated by entering the PIN code, sends a broadcast UDP datagram containing the message CCE_SERVER_DISCOVER to find out the IP address of the CCE server. Each CCE server, however, listens on an UDP port in order to respond to the broadcast requests of CCE clients with the message CCE_SERVER_DISCOVER_OK. The CCE client sends a maximum of four requests in broadcast with a timeout of ten seconds and if there is no response, it returns to its normal activity.

The CCE Sync phase consists of sending sessions from the client to the CCE server, in order to have backup copies of individual sessions. The exchange of data between the two entities is done through a secure channel of communication (SSL). The communication protocol follows the following steps (see Figure 5):

1. the first operation performed by the client, is to create an XML file containing sessions and hashes, so it is possible to know a priori if it has to establish an SSL connection or not. In the case in which there is at least one session to send to the server, the Handshake phase starts and subsequently, the client sends the message (CCE_CLIENT_INFORMATION: patient data).
2. The server receives the message CCE_CLIENT_INFOR-

MATION: patient data, retrieves data from the patient and sends the message CCE_CLIENT_INFORMATION_OK.

3. The client, after receiving the message CCE_CLIENT_INFORMATION_OK, sends to the server the message XML_SESSION_FILE_START, the XML file of the sessions, and finally sends the message XML_SESSION_FILE_END, to inform about the end of file.

4. When the server receives this message sequence, accepts the XML file, and once receiving the message XML_SESSION_FILE_END, reads the XML file, checking which client sessions are stored in the repository, also checking the hash to control if the card has been tampered with. Subsequently, it keeps in memory a list of files that has to receive, and starting from the first, sends the message GET_SESSION_FILE: file name.

5. The server receives this message sequence, and thus receives the XML file. After the message XML_SESSION_FILE_END, the server reads the XML file and checks which client sessions has kept in the repository, also checking the hash to control if the card has been tampered with. Subsequently, it keeps in memory a list of files that it expects to receive, and sends the files, starting from the first, with the message GET_SESSION_FILE: file name.

6. The client sends the message sequence SEND_FILE_SESSION_START: file name, the file to be sent and the message SEND_FILE_SESSION_END: file name.

7. The server receives this sequence of messages, receives the file and the message SEND_FILE_SESSION_END, calculates the hash and compares it with the one sent by the client in the XML file. If the two hashes do not match, the server sends back the message GET_SESSION_FILE: file name, for subsequent resubmission; otherwise it sends to the client the message SEND_FILE_RECEIVED_OK: file name.

8. The client receives the message SEND_FILE_RECEIVED_OK; then, sends the message SEND_FILE_RECEIVED_CONFIRM.

9. Finally, when the server receives this message, it sends to the client the message GET_SESSION_FILE: file name, to get the other files.

10. Once the list of files has been completed, the server sends the message END_SYNC to the client, to communicate the end of the synchronization.

11. The client receives the message END_SYNC and responds with a message END_SYNC_CONFIRM, then closes the socket.

12. When the server receives END_SYNC_CONFIRM, it also closes the socket and the SSL communication is terminated.

The CCE Update phase consists of sending a session from the server to the CCE client, in order to send the latest updates related to health cares carried out in the hospital. The session is generated by a specific software, which interact with the local servers of the health facilities involved in the project, so that this process is completely transparent to the preexisting IT infrastructures or services (e.g. PACSs). Moreover, the exchange of data between the two entities is done through a secure channel of communication (SSL). The communication follows the following steps (see Figure 7):

1. the first operation performed by the client is to send the message CCE_CLIENT_CF:tax code patient to the server.

2. The server receives the message CCE_CLIENT_CF: tax code patient, recovers the tax code and sends the message CCE_CLIENT_CF_OK.

3. The server then controls if a session exists to be sent to the client for that patient. After carried out this inspection, the server sends to the client the message YES_SYNC_SESSION (if the session actually exists) or NO_SYNC_SESSION (if the session does not exist).

4. In the case there is a session to be sent to the client, the server sends the message SESSION_HASH:hash of the session.

5. The client receives the message SESSION_HASH:hash of the session, and the server responds with the message SESSION_HASH_OK:hash received.

6. The server compares the hash sent with the hash received. If the two hash code coincide, the server sends the message SESSION_HASH_CONFIRM, otherwise it resends the message SESSION_HASH:hash of the session.

7. The client sends the message START_SEND_FILE to notify the server that it is ready to receive data.

8. The server sends the message sequence SEND_FILE_SESSION_START:session name, the session data and the message SEND_FILE_SESSION_END:name of the session.

9. The client, after receiving the message SEND_FILE_SESSION_END:name of the session, performs an integrity check on the data received by recalculating the hash and comparing it with the one previously sent by the server; if the comparison is successful, it sends to the server the message SEND_FILE_RECEIVED_OK. In the event of a negative outcome, the client sends again the message START_SEND_FILE, to tell the server to resend the data.

10. If the sending of data is successful, the server responds with the message SEND_FILE_RECEIVED_CONFIRM.

11. The client responds with the message END_SYNC to communicate to the server the end of the communication.

12. Finally, the server sends the message END_SYNC_CONFIRM.

13. In conclusion, both client and server close the socket and the SSL communication is terminated.

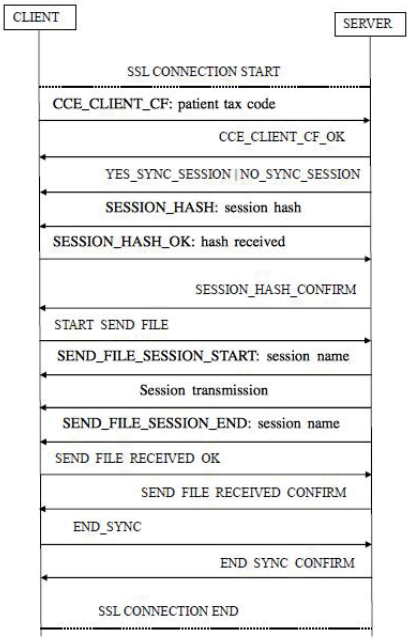

Figure 7: Update protocol.

4. RELATED WORK

The growth of computer systems, applications and infrastructures for the health services poses several issues [20][21], involving many aspects, such as sustainability, data integration or system reliability. Moreover, the centrality of patients within the care services, leads to involving them in the iterative process for the development of software tools as well as of the whole system architecture[22][13][17], or to let the patients participate actively in the data collection for epidemiological purposes [2]. This is mainly true, when dealing with chronic diseases, such as diabetes or cancer. Since the beginning of the therapy, in fact, patients may gather many clinical documents, ranging from medical examinations to clinical imagery.

Moreover, typically different health care structures are involved in the care process, each one with different software systems and data models, hence the need to share and integrate data coming from heterogenous sources [11][12], which is a problem that may require a somewhat complex solution [20][1][19].

As has already been acknowledged, distributed systems can represent a key solution for e-health care services. In [14], authors take benefits from the adoption of the Service Oriented Architecture, together with open standard such as XML and SOAP, to reach service interoperability and giving support to all the possible actors involved during patient care, such as physicians, nurses or pharmacists. In [8], authors propose to use a decision support system architecture, with distributed EHR systems, knowledge bases and data mining algorithm in order to give decision-making support to health care professionals. Moreover, the wide spread of technological devices capable of monitoring health parameters, and the adoption of tele-home care systems, may together prevent unnecessary hospitalization [17] and thus contribute to reduce the overall health care cost.

In our scenario, we observe fragmentation of clinical data. This makes it possible to avoid huge investments in data centers, but it poses several questions about data querying and load balance. In [9], authors present a methodology for query processing over distributed XML databases, underlying that data fragmentation is possible only if there is a transparent way to query the distributed database. Moreover, distributed databases, with the need of frequent data interchange, poses the key issue of security, which is particularly critical when dealing with clinical data [15]. Moreover, it has been noted [21] that the preservation of data relating to electronic health records (EHR) is a critical issue.

On the other hand, the possibility of collecting clinical data, anonymously and automatically, opens the way to the exploitation of these data for epidemiological research purposes [18]. In this sense, as has already been acknowledged [16][10], creating a dictionary of terms used is of crucial importance for the management of the clinical database.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a distributed and patient-centric architecture for electronic health record (EHR) systems. The proposed architecture has been developed within the CCE Project, connecting several public and private heath facilities. The main achievement of our system is that the patient is the real owner and holder of its/her clinical data, which are encrypted and stored in a smart card in the possession of the patient. Moreover, the data are replicated for security purposes, ad the update/synchronization process is completely transparent to the patient and to the health professionals, that do not have to quit using their usual software tools. Moreover, the proposed architecture makes it possible to collect a huge amount of data for epidemiological and administrative purposes, without violating the privacy of the patient, since each patient can digitally sign and send his/her anonymized data to researchers through the client.

As a relevant part of our future work, we identify two different contexts. As regards proper research-oriented aspects, we are considering several challenges: (*i*) dealing with massive clinical data, hence introducing suitable *compression paradigms* (e.g. [6, 3]); (*ii*) dealing with streaming clinical data, e.g. within the context of *body sensor networks*, hence studying *uncertain and imprecise information processing paradigms* (e.g., [4]); (*iii*) dealing with security and anonymity aspects of clinical data, hence focusing on *privacy-preserving models and algorithms* (e.g., [5]). As regards proper system-oriented aspects, we plan to deeper investigate algorithms and optimization mechanism, in order to query the distributed XML database generated by the implementation of the project. We are confident that the intelligent use of the collected data, may lead to better address health policies and would foster epidemiological and drug safety research efforts.

6. REFERENCES

[1] J. Anderson. Social, ethical and legal barriers to e-health. *International Journal of Medical Informatics*, 76(5-6):480 – 483, 2007.

[2] M. Biermann. A simple versatile solution for collecting multidimensional clinical data based on the cakephp web application framework. *Computer Methods and Programs in Biomedicine*, 114(1):70 – 79, 2014.

[3] A. Cuzzocrea. Providing probabilistically-bounded approximate answers to non-holistic aggregate range queries in olap. In *DOLAP*, pages 97–106, 2005.

[4] A. Cuzzocrea. Retrieving accurate estimates to olap queries over uncertain and imprecise multidimensional data streams. In *SSDBM*, pages 575–576, 2011.

[5] A. Cuzzocrea, V. Russo, and D. Saccà. A robust sampling-based framework for privacy preserving olap. In *DaWaK*, pages 97–114, 2008.

[6] A. Cuzzocrea and P. Serafino. *LCS*-hist: taming massive high-dimensional data cube compression. In *EDBT*, pages 768–779, 2009.

[7] L. Deri and R. Andrews. N2n: A layer two peer-to-peer vpn. In *Proceedings of the 2Nd International Conference on Autonomous Infrastructure, Management and Security: Resilient Networks and Services*, AIMS '08, pages 53–64, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] S. H. El-Sappagh and S. El-Masri. A distributed clinical decision support system architecture. *Journal of King Saud University - Computer and Information Sciences*, 26(1):69 – 78, 2014.

[9] G. Figueiredo, V. P. Braganholo, and M. Mattoso. Processing queries over distributed xml databases. *JIDM*, 1(3):455–470, 2010.

[10] T. Hannan, J. Rotich, W. Odero, D. Menya, F. Esamai, R. M. Einterz, J. Sidle, J. Sidle, F. Smith, and W. Tierney. The mosoriot medical record system: design and initial implementation of an outpatient electronic record system in rural kenya. *International Journal of Medical Informatics*, 60(1):21 – 28, 2000.

[11] R. Haux. Health information systems - past, present, future. *International Journal of Medical Informatics*, 75(3 - 4):268 – 281, 2006.

[12] R. Haux. Individualization, globalization and health – about sustainable information technologies and the aim of medical informatics. *International Journal of Medical Informatics*, 75(12):795 – 808, 2006. Jochen Moehr Special Issue.

[13] R. Heeks. Health information systems: Failure, success and improvisation. *International Journal of Medical Informatics*, 75(2):125–137, 2006.

[14] F. Kart, G. Miao, L. E. Moser, and P. M. Melliar-Smith. A distributed e-healthcare system based on the service oriented architecture. *2013 IEEE International Conference on Services Computing*, 0:652–659, 2007.

[15] L. Lhotska, P. Aubrecht, A. Valls, and K. Gibert. Security recommendations for implementation in distributed healthcare systems. In *Proceedings of the 42nd Annual IEEE International Carnahan Conference on Security Technology*, pages 76–83. IEEE, 2008.

[16] C. McDonald, J. Overhage, W. Tierney, P. Dexter, D. Martin, J. Suico, A. Zafar, G. Schadow, L. Blevins, T. Glazener, J. Meeks-Johnson, L. Lemmon, J. Warvel, B. Porterfield, J. Warvel, P. Cassidy, D. Lindbergh, A. Belsito, M. Tucker, B. Williams, and C. Wodniak. The regenstrief medical record system: a quarter century experience. *International Journal of Medical Informatics*, 54(3):225 – 253, 1999.

[17] S. Nourizadeh, C. Deroussent, Y. Q. Song, and J. P. Thomesse. Author manuscript, published in "mobihealth 2009 (2009)" a distributed elderly healthcare system.

[18] A. Nucita, G. Bernava, P. Giglio, M. Peroni, M. Bartolo, S. Orlando, M. Marazzi, and L. Palombi. A markov chain based model to predict hiv/aids epidemiological trends. In A. Cuzzocrea and S. Maabout, editors, *Model and Data Engineering*, volume 8216 of *Lecture Notes in Computer Science*, pages 225–236. Springer Berlin Heidelberg, 2013.

[19] F. Oliver and A. R. A. B H C Sputh, R Acharya. A pervasive design strategy for distributed health care systems. *The Open Medical Informatics Journal*, 2(1):58–69, 2008.

[20] C. Pelletier, T. Chaussalet, N. Szirbik, and C. Vasilakis. Integration of data on long-term care from heterogeneous sources for research purposes. In *Proceedings of the MEDICON'04*, Ischia, Italy, 2004.

[21] R. Scott. e-records in health–preserving our future. *International Journal of Medical Informatics*, 76(5-6):427–431, 2007.

[22] Y. Sumitaa, M. Takata, K. Ishitsuka, Y. Tominaga, and K. Ohe. Building a reference functional model for ehr systems. *International Journal of Medical Informatics*, 76(9):688–700, 2007.