35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Testing of Electronic Healthcare Record Images and Reports Viewer

Rita Noumeir, *Member, IEEE*, Jose Rose

Abstract— **Electronic Health Record (EHR) is a distributed system that results from the cooperation of several heterogeneous and autonomous subsystems. It improves health care by enabling access to prior diagnostic information to assist in health decisions. We focus on the image and imaging report visualization component that needs to interoperate with several other systems to enable healthcare professionals visualize a patient's imaging record. We propose and describe an environment that has been built and used to facilitate the development of the viewer component. This environment has also been used to test and verify the interoperability of the viewer component with other EHR components in accordance with the Integrating the Healthcare Enterprise (IHE) technical framework. It has also been used to demonstrate functionalities, to educate end users, and to train maintenance and test engineers. Moreover, it has been used for acceptance testing as part of an EHR deployment project. We also discuss the challenges we faced in constructing the testing data and describe the software developed to automatically populate the test environment with valid data.**

I. Introduction

Interoperability testing is crucial to Electronic Health Record (EHR). EHR improves the quality of care by enabling access to prior diagnostic information in order to assist in health decisions. Information includes observations, laboratory results, imaging reports, drugs, discharge summaries and allergies. This access is achieved independently from the institution where the information was initially gathered. EHR also improves productivity and reduces duplication of tests. EHR is not a single system that can be provided by a single manufacturer. It is a system that results from the cooperation of several heterogeneous distributed systems. Interoperability is therefore essential. Achieving interoperability requires, in addition to using communication standards, defining requirements and conducting thorough testing to eliminate and reduce technical, semantic, functional, quality and logistical impediments [1].

In this paper, we are interested in the EHR component that is responsible for visualizing medical images and imaging diagnostic reports. This component needs to interact with several other EHR components to enable the user search a specific patient's history, display specific patient's documents, visualize and process patient's images while ensuring information security and confidentiality. We briefly describe the Cross-Enterprise Document Sharing Integration Profile (XDS) that lays the infrastructure for image sharing between multiple care delivery systems [2]. This architecture allows sharing of patient's information in the form of documents. Medical images, important information of the patient health record, are shared with XDS. XDS has been adopted as a standard to share documents in many countries, including Canada and the United States; several projects are being deployed around the globe [3-5].

We focus our interest on one main component of this architecture: the image viewer; it is used by healthcare professionals to access patient's record, and view imaging reports and medical images. The viewer needs to successfully interact with all other components to support the healthcare process. After describing the process flow of the image and report viewer, we concentrate on the testing environment and testing data that have been created to test the interoperability of the EHR image viewer.

The EHR image viewer interacts with several other EHR components using different standards: the Digital Imaging and Communications in Medicine (DICOM) standard [6] which defines communication protocols and encoding of images, imaging reports and other imaging information; the Health Level Seven (HL7) which specifies the exchange and encoding of patient's information in several functional areas, such as patient registration, and delivery of diagnostic observations; standards initially developed for non-medical domains such as ebXML that have been adapted to the medical domain by the Integrating the Healthcare Enterprise (IHE) [7]. To develop and test EHR image viewer, all other peers are needed. However, these peers are usually provided by different vendors. We therefore propose and describe an environment that is based on open-source software to simulate other peers. We also describe how testing data has been created and published to testing peers. Creating the testing data is not an easy task. Data known to different peers need to be related to support the testing of a specific healthcare process. After describing how the environment is assembled, we present how testing data has been constructed and published to the various peers.

II. Architecture

A. Document sharing

The XDS architecture enables patient information, encoded as a document, to be shared between multiple care systems. A central registry maintains metadata describing published documents. It does not store the document, but answers queries about documents meeting specific criteria. The metadata includes information about the location from which documents may be retrieved. Documents are stored in a repository. Systems that generate patient's care information, such as radiology reporting systems, publish information as documents to the registry. Systems interested in accessing the

R. Noumeir is with Dept. of Electrical Engineering, Ecole de technologie supérieure (ETS), Montreal, Canada, e-mail:rita.noumeir@etsmtl.ca
J. Rose., is with Softmedical, Montreal, Canada, e-mail: jose.rose@softmedical.com

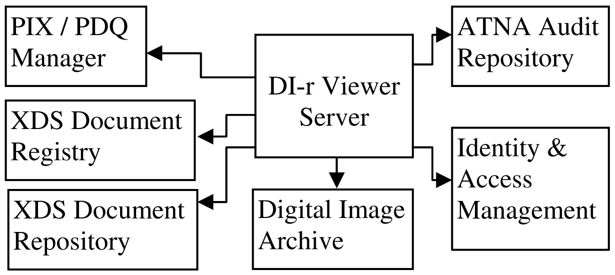


Figure 1. Viewer component and peers

patient's record query the registry for documents using query keys such as patient`s name or date of study. In the response to a query, the registry includes a reference to the document address enabling the consumer system to retrieve the document from its repository. The communication with the Registry/ Repository is done using the ebXML standard over SOAP. Fig. 1 shows the Viewer in the role of a document consumer; it interacts with the registry and repository in accordance to the IT Infrastructure (ITI) XDS integration profile [2, 7].

B. *Patient Identification*

The architecture is based on normalized patient's identifiers (ID): there is a single assigning authority (a Patient Identity Source Actor) that provides a unique ID for each patient. The Viewer component uses the normalized patient's identifier to communicate with the registry. This is done by querying a Patient Demographics Query (PDQ) Supplier to retrieve the normalized patient ID using demographic information, as shown in Fig.1. For example, the PDQ query is accomplished using a combination of patient's name, birth date, and /or ID as known to a specific institution. The query results consist of a list of patients whose demographic match the query. Each returned patient record contains also the normalized patient's ID.

C. *Security*

To achieve a minimum level of security the viewer implements the IHE Audit Trail and Node Authentication (ATNA) profile. This profile specifies the content of audit logs for all activities or transfers related to personal health information. Audits are communicated using the Syslog communication protocol to an Audit Repository (Fig 1).

D. *Sharing of imaging information*

To share a set of images, a manifest encoded as a DICOM key Object Selection (KOS), that contains references to those images is published. With this solution, the KOS is stored in the Document repository and the image are archived in the digital image Archive system that keeps the referenced images available to be retrieved as shown in Fig 1. Medical images are encoded in conformance with the DICOM standard. The manifest is another type of DICOM object; it contains the Universal Identifiers (UID) of the images that are referenced. Fig. 1 shows that the Viewer interacts with the image archive in order to retrieve the images.

III. IMAGE VIEWER SYSTEM

A. *Viewer process flow*

The image visualization application, or the viewer, is a system that is interested in retrieving and rendering published images and reports. The viewer process flow is depicted in Fig. 2. This is a complex system that has to interoperate with all other systems part of the XDS architecture. As a PDQ consumer, it starts by querying the PDQ supplier for patients whose demographic information matches the query criteria such as the name, and/or the date of birth, and/ or the patient`s ID. The PDQ Supplier returns a list of patients from which the user chooses one patient. Fig. 2 shows one audit message sent by the Viewer to the Audit Repository to inform it about the PDQ query that took place. The user chooses one patient and decides to query for published documents for that patient.

A query is then sent to the XDS Registry using the universal patient's ID that is retrieved from the PDQ response. The XDS Registry returns a list of documents for that specific patient. The Viewer sends an Audit message to the Audit Repository to inform it about the XDS query. This audit message is not shown in Fig. 2. From the list of documents, the user chooses one to be retrieved. If the selected document is an imaging report, it is retrieved from the XDS Repository and visualized. An imaging report is encoded as an HL7 Clinical Document Architecture (CDA); it is in XML format that can be transformed to be directly rendered in a browser. If the selected Document is an image manifest, it is retrieved from the XDS repository and decoded. The manifest content is a list of imaging series, where each series has a modality type and contains a certain number of images. The images themselves are not part of the manifest. Only their UID are encoded with the DICOM Application Entity from where they can be retrieved. When a document is retrieved from the XDS Repository, the viewer sends an audit messages to the Audit Repository (not shown in Fig. 2 for clarity). The user may select one or multiple series to be visualized. The images are retrieved using either DICOM C-Retrieve or Web Access to DICOM Persistent Objects (WADO) transactions.

B. *Authentication and access control*

The viewer consists of two components: a web client and a server. The web client runs in a browser; it enables the user to enter query parameters and select an item from a query result list; it also displays the images and reports. The server is responsible for the communication with all the peers; it runs behind a firewall. Between the client and the server, there is a reverse proxy (Fig. 3) whose main responsibility is to perform user's authentication: it intercepts the request to the viewer server; if the user is not already authenticated, it requests the user to enter username and password; the proxy authenticates the user by communicating with an authentication service; after the user is successfully authenticated, the request is forwarded to the viewer server; the request header is modified by injecting into it the user information including the user's role; the viewer server reads the user's injected information and uses it to enforce role

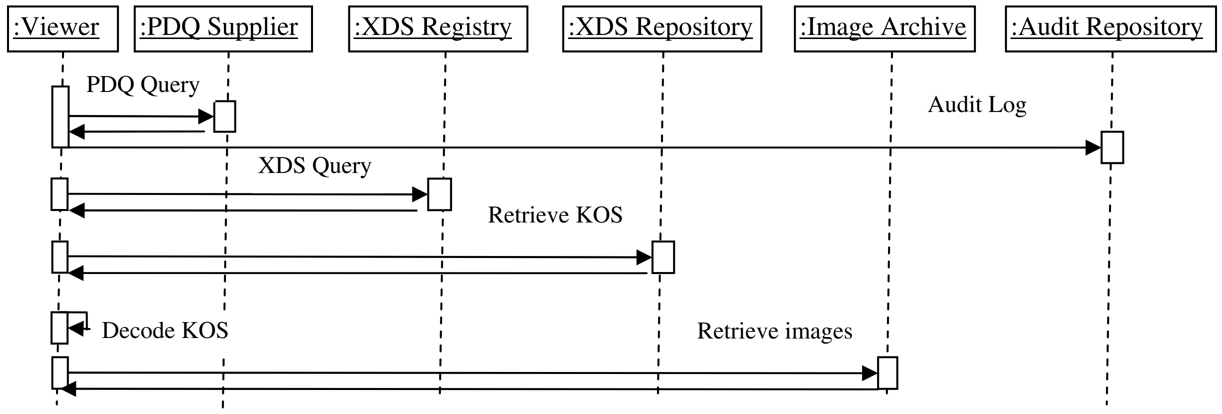


Figure 2. Viewer process flow

based access control; the server's response is sent back to the proxy who forwards it to the viewer client.

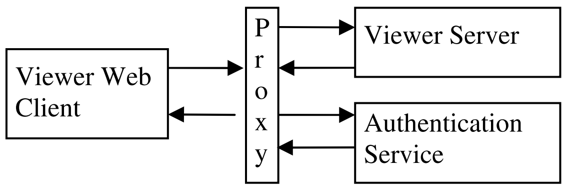


Figure 3. Viewer components and reverse Proxy

IV. Development and Testing environment

A. Software components and architecture

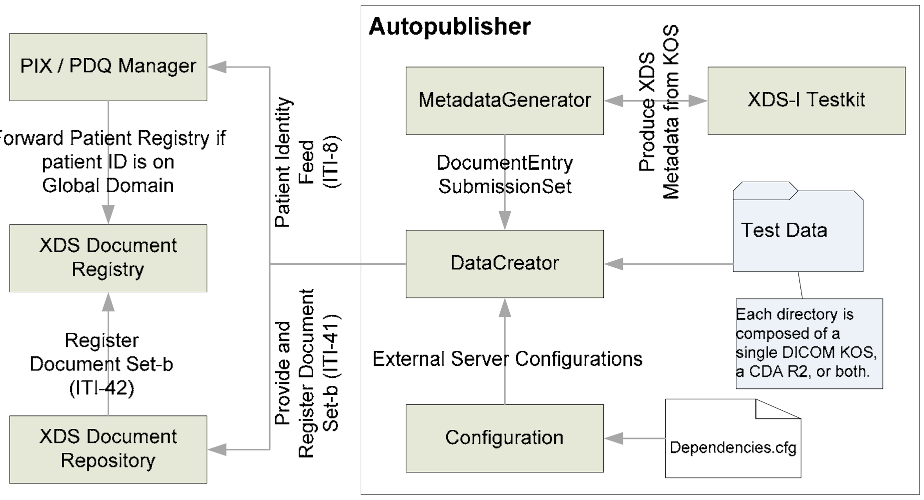
In order to test the viewer during development or during integration testing, an environment that simulates all communication peers is developed. The testing environment comprises: a PDQ supplier and PIX manager that are provided by the OpenPIXPDQ software [8]; an XDS registry and an XDS repository that are provided by the OpenXDS software [9]; an audit log repository provide by OpenATNA software [10]; a DICOM image archive provided by DCMTK software [11]; a WADO server provided by XDS-I Testkit software [12]; and a reverse proxy provided by the open source Web Scarab software [13].

These components have been configured to function as follows: the WADO server retrieves the requested images and reports from the DICOM image archive using a DICOM C-Retrieve transaction. The PIX manager forwards a Patient Identity Feed Transaction to the XDS registry when a patient is registered with a normalized ID. The XDS repository forwards a Document Register transaction to the XDS registry when it receives a Document through a Provide and Register transaction. The reverse proxy intercepts POST http requests from the web client and injects the user's identity into the http header before forwarding the request to the viewer server.

B. Testing data creation and consistency challenge

The major difficulty is the testing data consistency: 1- Each patient needs to be registered with the PDQ supplier with at least one ID that is not normalized in addition to the normalized one. This is to ensure that the exchange of information between the viewer and the PDQ supplier, as depicted in fig 2, can be done using the non-normalized patient`s ID, where the query response will always include the normalized ID. 2- For each patient, a KOS or CDA needs to be published to the XDS registry/repository. This is to ensure that the exchange of information between the viewer and the XDS registry returns at least one document.3- the images that are referenced inside the KOS need to be available for retrieve from the DICOM archive. This is to ensure that viewer can retrieve the images from the archive. 4- The AE title encoded inside the KOS needs to point to the DICOM archive. This is to ensure that the viewer can map the AE title decoded from the KOS to the address of the image archive peer.

To guarantee the consistency of data between the various peers, we have automated the publishing process by developing an auto-publisher software application. Based on the simple publisher from [12, 14] and on Open Health Framework (OHT) [15], the software creates registry metadata using either the KOS or the CDA, registers the patient with the PDQ supplier and publishes the documents to the XDS registry. In order to run various test scenarios, a single patient needs to be registered with multiple IDs, and multiple KOS or CDA documents are published for a specific patient. Therefore, the group of transactions to register a patient may be repeated; likewise, the group of transactions to publish documents for a single patient may also be repeated. For that reason, the auto-publisher prepares in memory a structure of the patients' data and documents before starting to populate the peers. The data flow diagram of the auto-publisher is depicted in Fig. 4. The input to the auto-publisher is an external XML configuration file that describes the data to be published, such as patients and their documents. When reading a KOS, the publisher generates an XDS document entry matching the KOS DICOM header. Likewise, the auto-publisher uses the information in the CDA header to generate a document entry, when there is no KOS for that patient. The KOS and the CDA are loaded into their respective document entries and added to the submission set. The auto-publisher registers a patient in PIX manager using the patient information from the KOS, or from the CDA if KOS is not available. The PIX manager is in charge of forwarding the patient register transaction to the XDS patient registry. At end, the auto-publisher publishes the documents



Autopublisher
PIX / PDQ Manager
Forward Patient Registry if patient ID is on Global Domain
XDS Document Registry
Register Document Set-b (ITI-42)
XDS Document Repository
Patient Identity Feed (ITI-8)
Provide and Register Document Set-b (ITI-41)
MetadataGenerator
Produce XDS Metadata from KOS
XDS-I Testkit
DocumentEntry SubmissionSet
DataCreator
Test Data
Each directory is composed of a single DICOM KOS, a CDA R2, or both.
External Server Configurations
Configuration
Dependencies.cfg

Figure 4. The auto-publisher data flow

to the Document repository that forwards the transaction to the Document registry.

C. *Benefits and usage scenarios*

The EHR viewer holds no patient's information; it depends on all other peers to operate. Therefore the testing environment is essential not only in the development process of the viewer but also for testing, demonstration and training. Although IHE provides testing for specific integration profiles such as PDQ or XDS, the proposed environment complements IHE testing by allowing quality impediments to be tested [1]. By populating the environment with large image sets, the viewer can be tested for memory consumption, timeliness, reliability and availability. Moreover, by simulating faulty situations, such as the non-availability of specific peers, the quality of service and the security impediments [1] can be assessed and accounted for. Moreover, during the lifecycle of the viewer, regression tests are conducted using all data deployed in the test environment. The data covers all possible information objects that can be encountered in reality such as many image types, reports, and presentation states. Furthermore, because the viewer holds no data, its functionality cannot be demonstrated without its peers. The testing environment is populated with de-identified data, so it is used for demonstration and for training.

V. Conclusion

We have described how the EHR images and reports visualization component interacts with various peers. In order to enable its development and interoperability testing, we presented how we assembled an environment that simulates the various peers. The major challenge encountered was the creation of testing data. Not only testing data needed to be carefully designed to enable various testing scenarios, it needed to be synchronized between various information objects and multiple systems. To overcome this challenge we have developed automatic publishing software to populate the testing environment with consistent testing data. The environment has been used to test the interoperability of the viewer component. It has also been used to demonstrate functionalities, to educate end users, to train maintenance and test engineers, as well as for deployment acceptance testing.

References

[1] Noumeir R., Requirements for Interoperability in Healthcare Information Systems, Journal of Healthcare Engineering 3, no. 2, p.323-346, 2012

[2] Noumeir R., Sharing Medical Records: The XDS Architecture and Communication Infrastructure, IEEE IT Professional, v 13, n 4, p 46-52, 2011

[3] EHR Progress in Canada- Canada Health Infoway, [Online] Available https://www.infoway-inforoute.ca/index.php/progress-in-canada, Feb 3rd, 2013

[4] Nationwide Health Information Network (NHIN) Overview, [Online] Available, http://www.healthit.gov/policy-researchers-implementers/nationwide-health-information-network-nwhin, Feb 3rd, 2013

[5] Where in the World is CDA and XDS, [Online] Available http://www.google.com/maps/ms?gl=us&ie=UTF8&oe=UTF8&msa=0&msid=110535847732151766411.00047b0b46314e91435c9, February 3rd, 2013.

[6] The Digital Imaging and Communications in Medicine (DICOM) standard, [Online] Available http://medical.nema.org, Feb 3rd, 2013

[7] IHE Technical Framework and supplements, [Online] Available http://www.ihe.net/Technical_Framework/index.cfm, Feb 3rd, 2013

[8] OpenPix/PDQ, [Online] Available projects.openhealthtools.org/sf/projects/openpixpdq, Feb 3rd, 2013

[9] OpenXDS, [Online] Available projects.openhealthtools.org/sf/projects/openxds, Feb 3rd, 2013

[10] OpenATNA, [Online] Available projects.openhealthtools.org/sf/projects/openatna, Feb 3rd, 2013

[11] DCMTK DICOM Toolkit, [Online] Available http://dicom.offis.de/dcmtk, Feb 3rd, 2013

[12] IHE-XDS-Imaging, Testing Software Source Code, [Online] Available http://sourceforge.net/projects/ihe-xds-imaging, Feb 3rd, 2013

[13] Web scarab, [Online] Available http://sourceforge.net/projects/owasp/files/WebScarab/, Feb 3rd, 2013

[14] Noumeir R., Bérubé R, IHE cross-enterprise document sharing for imaging: interoperability testing software, Source Code Biol Med., vol. 5, no 9, sept. 2010.

[15] Open Health Tools Java IHE API, [Online] Available projects.openhealthtools.org/sf/projects/iheprofiles, Feb 3rd, 2013