An Interoperability Test Bed for Distributed Healthcare Applications

Robert Snelick
National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, USA
robert.snelick@nist.gov

Abstract. Standards provide the foundation for ensuring interoperability, but if they are not implemented correctly or consistently their value is diminished leading to problematic installations and higher costs. Conformance and Interoperability testing is essential for ensuring standards are implementable and implemented correctly; however, limited budgets often preclude adequate attention to this testing during the product development life cycle. Automated testing can help on both fronts. We propose an Internet-based interoperability test bed that extends a testing infrastructure and conformance testing framework. The operational aspects of the architecture are presented against the backdrop of distributed health information technology applications and a representative case study. Although the concepts and methods are applied to the healthcare domain, they have broad applicability.

Keywords: conformance, data communication standards, healthcare, interoperability, messaging systems, testing

1 Introduction

A major challenge for the healthcare industry is achieving interoperability among proprietary applications marketed by different vendors. Each healthcare entity may have to use multiple applications to capture and share administrative and clinical data. Seamless and reliable exchange of information is difficult to attain. Recent mandates in the United States have ignited a renewed push towards interoperable healthcare information systems based on standards. Specification and wide-spread use of standards provide the foundation for ensuring system capabilities and the ability to exchange information reliably; however, standards alone are not enough. Testing and certification programs are necessary to assess implementations for conformance and interoperability. In prior work [1,2,3], we presented the testing infrastructure and testing tools we developed for conducting conformance testing. These tools have been used extensively in the US as part of

the health information technology (HIT) certification program related to meaningful use of electronic health record (EHR) systems [4,5,6]; internationally at the Integrating the Healthcare Enterprises (IHE) testing events [7,8]; and at production sites. We now present an extension of this work by describing the fundamental components of the testing infrastructure and operational facets of an interoperability test bed (ITB). A case study for patient identification cross-referencing is employed to describe a representative use case and to aid in explaining the ITB capabilities and operation. An overview of the applications involved, the roles the applications play, and the interactions between the applications is presented. We then describe how the interactions are tested in the operational environment. Central to the design of the ITB is that it must provide Internet-based, on-line, continual access to conformance and interoperability testing services.

The Health Level Seven (HL7) Version 2 messaging standard [9] is the focus for discussing the aspects of the ITB. HL7 is a data exchange standard for transmitting clinical and administrative information between healthcare applications [9]. Typical HL7 messages include admitting a patient to a hospital or requesting a laboratory order for a blood test. Conformance message profiles are used to constrain the set of data that is exchanged between applications that support the HL7 standard; these message profiles are the source of the conformance requirements that verify correct data exchange [2].

2 Testing Infrastructure

The central design principle behind the construction of a test system is a modular and flexible approach afforded by a testing infrastructure. The testing infrastructure provides reusable components with well-defined interfaces that can be utilized in a test system as needed. The obvious advantage of this approach is that the user can combine the components of the test system into whatever configuration is most suitable for solving the problem at hand. The system is also easily extensible since each test system component runs independently, and adding new services to the test system can be accomplished readily. The test systems described and referenced in this work, including the ITB, adopt this approach.

The testing infrastructure is composed of three types of high-level components: the services, the test harness (services composition), and network functions. Additionally a test management system may be employed, but it is not central to the test infrastructure—it is likely a separate system. Services provide the testing functionality. It is apparent that for this model to be used effectively in conducting testing, it will be necessary to facilitate the interactions between the user and the supported testing services. A test harness is necessary to orchestrate the services to conduct a test. A network is utilized to route messages and may include logging and proxy capabilities. A test management system (e.g., IHE Gazelle [11]) can be used to assign, manage, and track each set of tests. A certification testing lab may utilize such a system in their process.

Services provide a portfolio of testing-related capabilities that are specified to perform unique functions within the testing infrastructure. Each service has well-

defined responsibilities and authority, and they work together in collaboration with other services to support the execution of test cases. The generation service creates a message instance based on a message profile (template) and a data set. The validation service evaluates a message instance against requirements stated in the standard. Data content also can be evaluated to support application functional testing. Test agents (also referred to as simulators) are implementations of actors (or applications) that support the functionality of the underlying specification of the actor and need only support the functionality of the actor to support testing of applications. The resource repository contains the artifacts necessary for conducting tests and facilitating test execution processing. Artifacts that support test case execution typically include the test execution script, test case descriptions, test data specifications, test data, and test specifications (e.g., conformance profiles). The testing infrastructure is not bound to a defined set of services; rather, the set is determined by the objectives of the testing goals and system.

3 Test Environments

Recognizing that testing is a complex, multidimensional, and often incremental process leads us to consider the use of multiple environments for conducting testing. In an earlier work [12], we identified three distinct environments and described the testing activities that can be performed within each environment. These environments include the data instance test environment (not discussed here), the isolated system test environment, and the peer-to-peer system test environment. The delineation of environments and their testing capacity is intended to facilitate a more structured approach to testing in which the relationship between test requirements and testing, along with an understanding of the capabilities and limitations of testing tools, is more clearly defined.

In the isolated system testing environment a test is conducted with the SUT and a test tool. Conformance testing, including data exchange and functional behavior, is the main objective in using this model. Functional behavior assessment is achieved with a test scenario in which a sequence of orchestrated transactions is composed to probe certain requirements. The test tool includes functionality of an application (i.e., test agents) that an SUT would typically interact with in an operational environment. Isolated system testing typically accounts for the majority of testing that is conducted. Once a system has successfully undergone conformance testing, interoperability testing usually proceeds more easily.

Peer-to-peer system testing is designed to test interoperability among one or more systems and is the focus of the ITB. The peer-to-peer system testing environment poses different and significant challenges in testing from that of isolated system testing. In this environment, data exchange occurs among a group of systems, and the testing tool no longer has direct interaction with the systems under test. Here an intermediary or a proxy can be employed to intercept, log, and route messages to their intended destination. Peer-to-peer system testing may include some or all of the conformance testing described for isolated system testing. When conformance testing is conducted in advance, peer-to-peer testing specifically targets both syntactic interoperability and semantic interoperability

testing. The conformance test cases that were developed for isolated system testing can be leveraged in peer-to-peer testing. The abstract test cases could be identical; however, execution of the test steps, configuration requirements, and assertion assessment will differ. By ascertaining that the conformance requirements are now met in an environment where the SUTs are interacting, we can make a declaration of the interoperability capabilities of the systems.

IHE testing involves a number of incremental testing steps. First, vendors conduct conformance testing of their products. This task involves testing in isolation to determine if the system implements the requirements specified in the standard. These tests are labeled *pre-connectathon* tests and correspond to the isolated system test environment. The NIST IHE Patient Identification Cross-referencing (PIX)/Patient Demographics Query (PDQ) [7,8] and Patient Care Devices (PCD) pre-connectathon test tools are examples of production implementations of the isolated system test environment. Interoperability testing (peer-to-peer testing) is conducted at an event called a *connectathon* in which scores of vendors bring their products to a central site and live-monitored tests are performed over a period of a week [10]. Such concentrated events are useful as vendors can interact with many other vendors in a short period of time; however, connectathons occur infrequently (once a year in the United States and Europe) and are costly. One objective of the ITB is to provide an intermediate format in the form of an on-line *virtual connectathon* in which interoperability testing is *always* available. In the virtual connectathon environment, participants indicate what they want to test and publish their availability. Once testing partners reach agreement, their systems are configured in the test bed and they can proceed with testing. The ITB leverages the testing infrastructure and implements the peer-to-peer testing environment.

The ITB isn't fundamentally that different from the test system for the pre-connectathon isolated test environment. The main divergent point is testing multiple *real* systems instead of one, i.e., replacing some or all of the test agents in the isolated system environment with vendor products. This difference does, however, present noteworthy technical challenges including the scheduling of participants, sequencing of events and notification to the participants, and capturing/forwarding messages and then mapping the messages to the corresponding test case interaction.

4 Case Study: Patient Identification Cross-referencing

In this section, we describe an example case study and associated workflow that demonstrates typical transactions among disparate healthcare information technology systems. IHE publishes integration profiles that describe many healthcare workflows [7]. We describe a typical workflow of cooperating patient identifier and document management systems. The data exchange standard involved in this use case is HL7 V2. For the purposes of demonstrating the test execution of the ITB the scenario focuses on the HL7 V2 messaging PIX aspects.

Our example examines a healthcare system made up of a Patient Identifier Cross-referencing (PIX) Domain and a Cross-Enterprise Document Sharing (XDS)

system. We examine a typical PIX domain made up of three disparate actors: a PIX Source, a PIX Manager, and a PIX Consumer. A PIX Source is used for adding and modifying patient demographic data; a PIX Manager is used for managing and cross referencing patient identifiers from different domains; and a PIX Consumer is used for querying a PIX Manager for patient identifiers. All communications among the actors are accomplished through the exchange of HL7 V2 messages. An XDS system supports registering and retrieving documents across enterprises within an administrative domain.

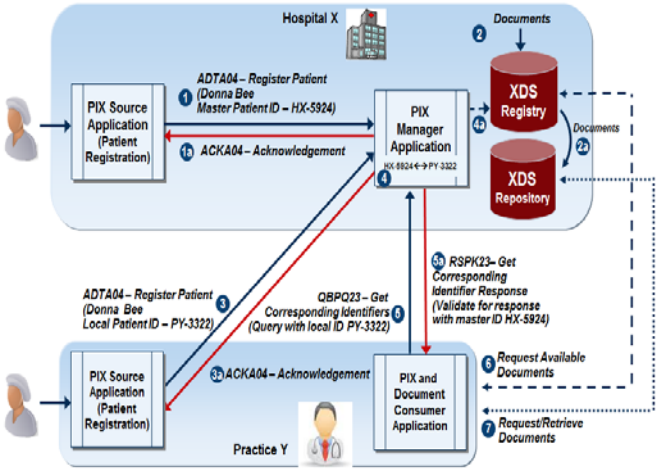

Fig. 4.1. Patient Identification Cross-referencing Workflow

Healthcare systems can be divided into various administrative domains, each responsible for managing a set of patient information. Patients may require services provided in different healthcare domains. When this situation occurs, different electronic health records for the same patient may exist in more than one domain. It is clearly desirable to be able to recognize when multiple records exist belonging to the same patient. IHE has addressed this problem by delegating the responsibility for determining when two patient identifiers belong to the same patient, and hence there are two records belonging to the same patient, to the PIX Manager actor. Our case study (**Figure 4.1**) examines some of the data points the PIX Manager must consider; for example, when it is determined that two patients are actually the same patient, how will the information be propagated throughout the healthcare continuum? The resolution is important since a single patient identifier is typically used to retrieve documents about a patient from a repository.

The scenario (workflow) is predicated on a family doctor at Practice Y seeking pertinent medical documents for patient Donna Bee. Practice Y relies on Hospital X's patient management and document repository systems; however, before the doctor can retrieve the documents from the repository he must obtain the patient identifier used by Hospital X's document repository. Practice Y acting as a PIX Consumer queries (via HL7 V2 message) the PIX Manager using its patient identifier for Donna Bee (PY-3322) and requests the master patient identifier in Hospital X's domain. The PIX Manager returns an HL7 response message

containing the patient identifier (HX-5924). Once the patient identifier is acquired, it can be used to query for available documents in the registry and retrieve these documents from the repository. This scenario assumes that the relevant documents have already been uploaded into the repository via a document source actor.

The three steps below show a test case derived from the above scenario. Although the test case is simple, it provides a representative set of applications and steps that have to be accounted for in testing and supported by the ITB constituent components. One patient (Donna Bee) is registered in different domains. The registration messages are sent to a PIX Manager. A query is sent to resolve a reference to patient Donna Bee who is expected to be found.

Step 1: Hospital X PIX Source sends a registration message (ADT^A04) to register patient Donna Bee in domain HOSP-X. Patient ID is HX-5924. The PIX Manager shall register the patient and send an Acknowledgement (ACK) message back the PIX Source.

Step 2: Practice Y PIX Source sends a registration message (ADT^A04) to register patient Donna Bee in domain PRAC-Y. Patient ID is PY-3322. The PIX Manager shall register the patient and send an ACK message back to the PIX Source.

Step 3: Practice Y PIX Consumer sends a query message (QBP^Q23) to ask for Donna Bee's ID in domain HOSP-X using their ID PY-3322 in domain PRAC-Y. The PIX Manager is expected to respond to the query with Donna Bee's ID HX-5924 in domain HOSP-X.

Successful completion of the case study requires that each application involved in the process correctly performs certain tasks that can be measured based on the application's externally observable behavior. The requirements on the application's external behavior can be formulated as a set of conformance and interoperability testing requirements. Although no explicit interoperability requirements have been defined, successful completion of all of the steps in the workflow provides a prima facie demonstration of interoperability among the systems.

5 Interoperability Test Bed

The interoperability test bed supports the peer-to-peer testing environment that targets multiple systems under-test. **Figure 5.1** depicts a scaled-down design of the architecture that employs a set of fundamental components and operating procedures necessary to conduct testing. A description of each component and its capability is described followed by an abbreviated step-by-step test flow.

5.1 Operational Functions

The interoperability test bed has a number of logical operational divisions. A ***scheduling system*** is required to coordinate the vendors and match up interest in test selection, actors, and time availability. Vendors input their testing interest and capabilities along with time availability, and the scheduler will pair up common

requests and will notify the participants. The ***configuration utility*** records the connection and addressing information required for communicating with each participant. This information is necessary for the ITB to prime the proxy to handle intercepting and forwarding messages. ***Test setup*** includes informing each of the participants of the test instructions associated with the test case. These include their roles, actions required, data requirements, and system configuration (e.g., loading test data in the data base). The ***test manager*** controls the overall operation of the ITB. These activities include managing simultaneous test executions and initiating test case instance executions.

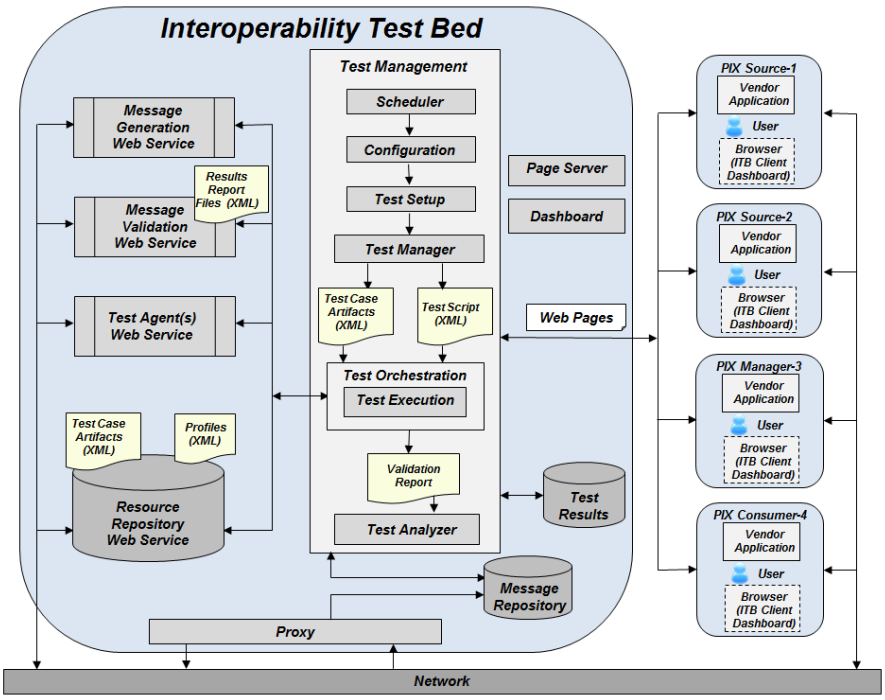

Fig. 5.1. Interoperability Test Bed Architecture

The test management system relies on a ***test orchestration*** component to organize and coordinate the collective activities of the vendors and the test system. Information (e.g., instructions and status) is disseminated through a common shared user interface and a participant-perspective user interface. A test script encapsulates the choreographed steps and participant actions. The ***test engine*** is a workflow management tool that directs the execution of defined test cases through the orchestration of testing services. The test execution operates at a level below (within) the test orchestration and coordinates the measurement aspects (e.g., message validation) of the test case instance. The test execution is independent of the test orchestration since its scope is coupled to the test case. Therefore the same test execution script is applicable to isolated system testing. For each task that is performed either by the test system or participant, a status update is ***broadcast*** (served pages) to the participants. This includes a human test manager observing

the execution of the test instance from a test management dashboard. As messages progress through the system they are captured, stored, analyzed, and forwarded. The ***proxy***, ***message database***, and ***validation*** provide these capabilities. For each interaction (message), the test execution assembles the various artifacts and submits them to the validation service for evaluation. A machine-processable report is returned and rendered to the participants. A test case instance consists of numerous steps for the applications being tested and creates many individual validation reports. A ***test analyzer*** is necessary to make a collective determination of the test results. The test analyzer can use the test script for navigating, linking, and analyzing the array of validation reports. Upon completion of the test instance execution, the test results are stored for analysis and auditing purposes.

The ITB includes the capability to supply a vendor system(s) when not all the participant systems needed are available or not available at the desired time. These features include message generation and test agent services. Having one SUT and all test agents is essentially equivalent to isolated system testing provided by the pre-connectathon testing tools. The ITB encapsulates this model.

5.2 Test Flow

The test flow describes each of the actions required by the ITB and participants to perform our test case instance. The test case calls for four applications or application modules. There are two PIX sources (PS-1 and PS-2), a PIX Manager (PM-3), and a PIX Consumer (PC-4). Our focus is on the operational aspects of the test instance. **Figure 5.2** illustrates the events as they progress through the ITB for the transaction 1, interaction1 described below.

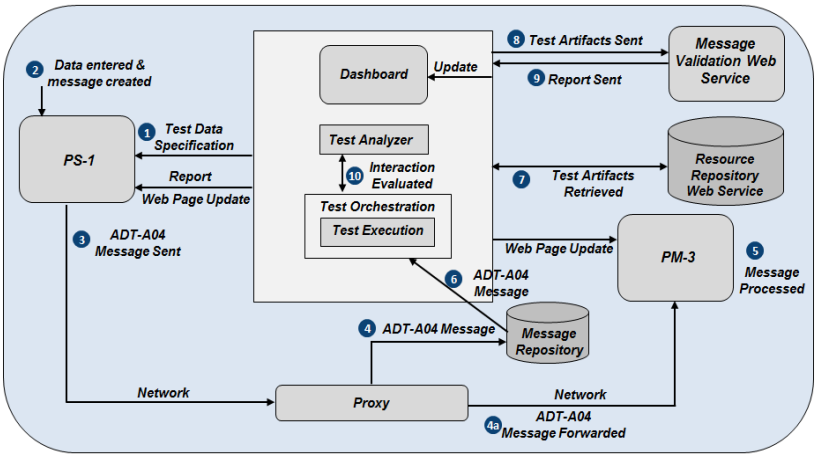

Fig. 5.2. Test Flow for Transaction 1, Interaction 1

ITB Actions: PS-1 is provided a test data specification that includes data (e.g., patient name and DOB) associated with the test story and consists of typically available information in the administrative and clinical setting. Together, the test story and the test data specification provide sufficient information to be entered into the SUT for a particular test case such that a message can be generated.

PS-1Actions: PS-1 loads the test data provided in the test data specification; typically, loading is a manual process performed using the user interface (UI) capabilities of the SUT. Once all relevant test data are loaded into the PS-1, a message is generated and sent. Based on the configuration, the message (intended for the PM-3) is sent to the ITB proxy. PS-1, via the dashboard indicates that it has sent the message; this controlled execution eases the test instance navigation for the test execution engine.

ITB Actions: The ITB proxy captures and stores the message in the message repository along with pertinent message meta-data sufficient for mapping the message to the interaction. The ITB test engine retrieves the message from the message repository and the conformance validation artifacts from the resource repository and invokes the validation service to validate the message. The ITB test analyzer evaluates the interaction along with validation results to determine if any errors detected would prevent the test from continuing. Note that inconsequential conformance issues are not pertinent at this point; the focus is on the delivery of the message and key data elements that might prevent other participants from completing their tasks. If a severe fault is detected, the test orchestration notifies all participants accordingly via the common shared display capability and then ends the test instance. The ITB proxy forwards the message to the PM-3. The test orchestration reports that interaction 1 is completed and the common user interface display is updated accordingly. The validation report for the interaction is made available to the participants. The ITB now instructs PS-2 to begin interaction 2. Note that the ITB also updates the status at intermediate events, for example, when the message is received by the proxy or forwarded.

Upon completion of the test case instance, the test analyzer evaluates the collective results and determines the outcome of the test instance. This information, along with the individual interaction validation reports, is provided to the participants. The vendors can use these test results, as well as logging information and the observed behavior of their system to further assess the performance of their product. A test manager can review the reports to make an assessment of the test instance. All related test instance data are stored in the test management system for auditing purposes if desired.

Although we have presented here a simplified portrayal of the architecture and case study, it is still evident that the ITB provides the capabilities to conduct conformance and interoperability testing in support of distributed healthcare applications for a broad assortment of test cases essential for comprehensive testing. For example, an interoperability test may stipulate incorporation and display of laboratory results into an EHR. In this case, completely automated black box testing is unattainable; however, an inspection document based on the test data is dynamically created and provided for this test step. Screen shots of the vendor's display and/or data base can be captured and submitted to the test system for visual evaluation by a monitor. A monitor will use the inspection document to verify expected behavior.

6 Conclusion

Interoperability between disparate applications can be achieved better through the use of standardized interfaces. Even if the same standard is implemented in the applications, interoperability is not assured due to two primary reasons: (1) the same sets of options allowed by the standard are not implemented by the developers, a problem that can be addressed with conformance provisions offered by the standard; (2) the standard is implemented incorrectly in the applications. These issues are addressed through conformance testing. Applying conformance processes and successfully conducting conformance testing will not ensure interoperability, but these actions will increase the likelihood of applications interoperating. Beyond conformance testing, interoperability testing is employed. Interoperability testing has been challenging because assembling even a minimum number of application vendors in one place at the same time to conduct this type of testing is difficult. For years, the IHE connectathon has provided an invaluable venue for massive interoperability testing; however, this event occurs only once a year. The proposed *always-available* Internet-based interoperability test bed seeks to fill this gap and complement connectathon events.

7 References

[1] *A Framework for testing Distributed Healthcare Applications*. R. Snelick, L. Gebase, G. O'Brien. 2009 Software Engineering Research and Practice (SERP09), WORLDCOMP'09 July 13-16, 2009, Las Vegas, NV.

[2] *Towards Interoperable Healthcare Information Systems: The HL7 Conformance Profile Approach*. R. Snelick, P. Rontey, L. Gebase, L. Carnahan. Enterprise Interoperability II: New Challenges and Approaches. Springer-Verlag, *London Limited 2007 pp*. 659-670.

[3] NIST Laboratory Results Interface (LRI) EHR Meaningful Use Conformance Testing Tool; http://hl7v2-lab-testing.nist.gov/mu-lab/

[4] NIST Resources and Tools in Support of Health IT Standards and the ONC Meaningful Use certification program. http://healthcare.nist.gov/

[5] Centers for Medicare & Medicaid Services (CMS) EHR Incentive Program http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/index.html

[6] ONC Certification Programs; http://www.healthit.gov/providers-professionals

[7] Integrating the Healthcare Enterprises Technical Framework; http://www.ihe.net

[8] NIST PIX/PDQ Pre-connectathon Conformance Testing Tool; http://pixpdqtests.nist.gov:8080/

[9] Health Level 7 (HL7) Standard Version 2.7, January, 2011; http://www.hl7.org.

[10] IHE Connectathon; http://www.ihe.net/Connectathon/

[11] Gazelle Test Management Framework; http://gazelle.ihe.net/

[12] "*Testing Environments for Assessing Conformance and Interoperability*" L. Gebase and R. Snelick. 2010 Software Engineering Research and Practice (SERP10), WORLDCOMP'10 July 12-15, 2010, Las Vegas, NV.