A Framework for Testing Distributed Healthcare Applications

R. Snelick[1]**, L. Gebase**[1]**, and G. O'Brien**[1]
[1]National Institute of Standards and Technology (NIST), Gaithersburg, MD, State, USA

Abstract - *We present an architectural framework for supporting conformance and interoperability testing among distributed healthcare systems. Healthcare organizations such as hospitals employ information systems that are composed of multiple, distributed applications. These applications need to interoperate seamlessly. This presents a challenging testing problem. We describe in detail a typical healthcare workflow—presenting the applications, their roles, and transactions. We then examine the testing requirements that emanate from the workflow. Next we present a test system and describe how the system can be used to satisfy the testing requirements. Finally, we present a generalized approach based on a service oriented architecture that supports testing of a broad range of healthcare workflows that employ multiple messaging and document data exchange standards. We assert that the proposed testing framework is applicable for a broader class of applications.*

Keywords: Automated Testing; Conformance; Interoperability; Service Oriented Architecture; Testing Framework; Workflow.

1 Introduction

Healthcare organizations such as hospitals employ information systems that are often composed of heterogeneous applications provided by a variety of vendors. Market fragmentation is an artifact of the vast scope of healthcare services; no one organization can develop systems to support all services. The result is that vendors build specialized products—for example, lab results reporting or a patient registration system. In a healthcare organization, these systems need to exchange information seamlessly and reliably. Standards based systems are the foundation for achieving this goal. Applications communicate via widely used healthcare clinical data and imaging exchange messaging standards and clinical data document standards. A comprehensive information healthcare system will incorporate data from a variety of disparate, heterogeneous applications. For example, the system may include Health Level Seven (HL7) data, Digital Imaging and Communications in Medicine (DICOM) data, and Clinical Document Architecture (CDA) documents. Each standard is used for a specific purpose; testing support is needed for each.

HL7 is a data exchange messaging standard for moving clinical and administrative information among healthcare applications [1]. Typical HL7 messages include admitting a patient to a hospital or requesting a lab order for a blood test. Conformance message profiles are used to constrain the set of data that is exchanged among HL7 applications; message profiles provide the conformance requirements that verify correct data exchange [9]. DICOM is a standard for handling, storing, printing, and transmitting information in medical imaging systems [2]. DICOM enables the integration of scanners, servers, workstations, printers, and network hardware from multiple manufacturers into a picture archiving and communication system (PACS). DICOM devices come with conformance statements which clearly describe the classes they support. DICOM has been widely adopted by hospitals and is making inroads in smaller applications like dentists and doctors offices. CDA is an XML-based markup standard intended to specify the encoding, structure, and semantics of clinical documents for exchange [13]. CDAs can be specialized; for example a Continuity of Care Document (CCD) component was developed to facilitate the communication of patient summary information between electronic systems. The specification for CCD supports conformance testing of such documents.

The use of standards is essential for building a healthcare system made of subsystems and multiple applications coming from different vendors. New applications cannot easily be incorporated into such a system without each new application exhibiting behavior that is in strict compliance with an agreed upon standard. Assuring that applications are compliant then is an important task and the focus of conformance testing. We present a conformance testing framework for supporting the evaluation of a collection of independent healthcare systems. We will examine the problem by illustrating a scenario describing patient identification and document management. We provide an overview of the applications involved, the roles the applications play, and the interactions between the applications. For a representative set of transactions we describe the requirements that must be satisfied to successfully complete the transaction. We then present a testing strategy to evaluate adherence to the requirements that were formulated from examining the workflow scenario. This process leads to the introduction of a service oriented architecture (SOA) model that can be applied to the testing of a broad spectrum of healthcare applications.

The framework defines a set of services to support testing. The key services include test agents (implementations of application functionality to the extent necessary to support testing), validation services (e.g., HL7 messages and CDA documents), test data generation, logging, and repositories (e.g., for maintaining testing artifacts). Additional services are also available but are not discussed here.

An important characteristic of the testing framework is its flexibility to support multiple configurations. The test framework, like the healthcare system it is designed to test, is made up of a collection of discrete tools and applications, each providing one or more testing services. Evaluating application compliance may in one case require only evaluating correct message construction. In this case, the test system may be configured as a single application providing a message validation service. In another case, evaluation of multiple healthcare subsystems made up of multiple healthcare applications may require deploying a testing configuration consisting of the entire array of tools available to the system. 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. In addition to the flexibility offered by the system, it is also easily extensible. Since test system components run independently, adding new services to the test system can readily be accomplished. Adding a new service requires little more than defining the services the application will be required to support and the mechanisms the application will be required to use for interactions with other components of the system. The framework, through various service compositions affords the user with the flexibility to select the approach, resources, and environment in which they want to conduct testing. In addition to being flexible and extensible, the testing framework also scales well since its resources requirements can be distributed across multiple discrete systems. Finally the framework supports rapid test suite development since they can be built independently and simultaneously. Stakeholders, such as certification bodies or vendors testing in-house, can build the testing system that meets their requirements.

2 Example Healthcare Workflow

We describe an example workflow (also referred to as a use case) that demonstrates typical transactions among disparate healthcare information technology systems. We first give an overview of the applications involved in the example, including the functional role played by the application. An application is generally made up of a single actor, but may include multiple actors. The Integrating the Healthcare Enterprise (IHE) organization defines an actor based on the functional components supported by the actor [3]. IHE publishes integration profiles that describe many healthcare workflows. In our use case example an actor can be equated with a healthcare application such as the patient identifier cross-referencing manager (PIX Manager). Note that a PIX Manager actor may exist within a hospital's healthcare information technology system. We describe a possible workflow of cooperating patient identifier and document management systems. The scenario is supported by the IHE Information Technology Infrastructure Integration Profile [4]. The data exchange standards involved in this use case include HL7 messaging and clinical documents (CDA).

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 and data. All communications among the actors is accomplished through the exchange of HL7 messages. An XDS system supports registering and retrieving documents across enterprises but within an administrative domain. XDS exchanges patient identification information via HL7 messages and exchanges XML based documents associated with patients in the form of a CDA.

Healthcare systems can be divided into various administrative domains, each responsible for managing a set of patient information. Patients, though, may require services provided in differing healthcare domains. When this occurs, 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 the records belonging to the same patient, to the IHE PIX Manager actor. Our use case examines some of the data points the PIX Manager must consider; for example when it is determined that two patient's match, how will the information be propagated throughout the healthcare environment. This is important since a single patient identifier is typically used to retrieve documents about a patient from a repository.

The workflow illustrates the integration not only among applications within a healthcare information technology system, but also between systems. This example assumes that appropriate information sharing policy agreements exist and that appropriate security and privacy protocols are used. Refer to Figure 1, Example Workflow.

1. Hospital A has a patient administration system (ADT) that includes a PIX Source actor. When a new patient, John Doe, is registered through the system, the PIX Source generates a patient identifier (HA5882). Using the identifier and demographic data, an HL7 message (Patient Identify Feed—PIF) is constructed and sent to the PIX Manager. The PIX Manager is part of Hospital A's information technology system.

2. The XDS system, also part of Hospital A's information technology system, is notified that a new patient has been registered. The same HL7 PIF message is sent to the XDS Registry actor. The XDS Registry now knows about patient John Doe.

3. A clinical document (e.g., a patient summary document; a CCD) following the CDA for patient John Doe is created by Hospital A. The summary document is registered with the XDS system using a Provide and Register message. Hospital A's Document Source actor sends the document to the XDS Repository actor.

4. The same John Doe visits Clinic Z which maintains its own patient registration system (ADT) that has a PIX Source actor. It too now registers a patient, John Doe, and assigns the patient an identifier, CZ-7441. The patient identifier along with other patient demographic data is used to construct a message PIF. The PIF is then sent to Hospital A's PIX Manager. Clinic Z relies on Hospital A's patient identification cross-referencing management system.

5. The Pix Manager on receiving the message from Clinic Z for John Doe applies its cross referencing algorithm to compare the message data with the patient information it is already maintaining for John Doe. After applying the algorithm, it determines that the John Doe in its database and the John Doe being registered for Clinic Z is the same patient. The PIX Manager links the patient identifiers for John Doe.

6. Clinic Z collects and produces medical information about John Doe during his visit and creates two documents. The documents include a patient summary (CCD) and a lab report document. Clinic Z relies on Hospital A's patient management and document repository systems. In order for Clinic Z to store the document in Hosipital A's repository it must obtain the patient identifier (called the affinity domain identifier) used in the XDS Registry. Therefore Clinic Z must Query the PIX Manager.

7. Clinic Z submits the documents to the XDS Repository using a Provide and Register message.

8. John Doe now visits Medical Center B for a medical procedure. Medical Center B has a patient registration system (ADT) that includes a PIX Source actor and a PIF message is sent to the PIX Manager. John Doe's patient identifier at Medical Center B is MCB3319. As before, the PIX Manager determines based on demographics data that this is the same John Doe. The patient identifiers are linked in the PIX Manager.

9. A doctor at Medical Center B wants to obtain pertinent medical documents for patient John Doe. Medical Center B relies on Hospital A's patient management and document repository systems. However, before the doctor can retrieve the documents from the repository it must obtain the patient identifier (called the affinity domain identifier) used in the XDS Registry. Medical Center B now acting as a PIX Consumer queries the PIX Manager using its patient identifier for John Doe (MCB3319) and requests the patient identifier in Hospital's A domain. An HL7 query message is used to perform this task. The PIX Manager returns an HL7 response message containing the patient identifier (HA5882).

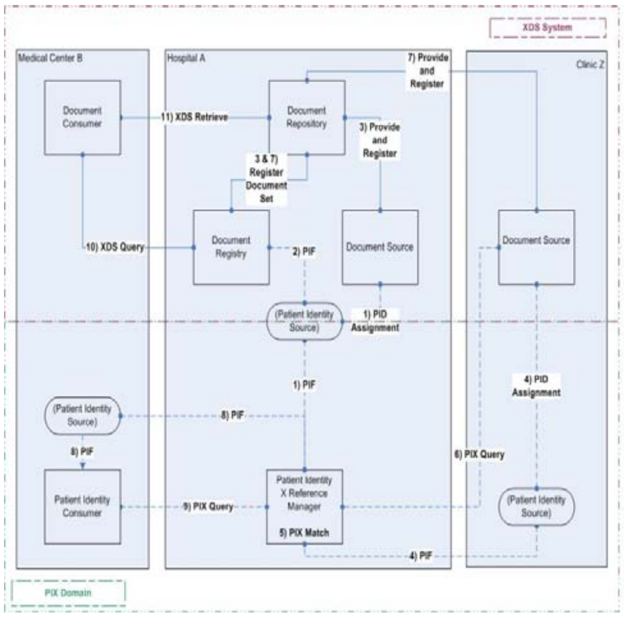

Figure 1: Patient Identification and Document Management Workflow

10. Medical Center B can now use the patient identifier HA5882 to query the XDS Registry. The registry doesn't return John Doe's clinical documents; it returns metadata about the available documents, including the identifiers to retrieve them. Note that the registry manages documents created in Hospital A and Clinic Z.

11. The doctor at Medical Center B reviews the information and can use the references to retrieve the desired CDA documents from the XDS Document Repository.

3 Testing Requirements

The workflow describes a number of applications (actors) and the transactions among them; the integration is not only among applications within a healthcare information technology system, but also between systems. Successful completion of the above use case 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. They include message validation, document validation, actor behavior validation, and protocol conformance. Each system involved in a message exchange must send messages that adhere to the message construction requirements defined by the associated standard. Additionally, HL7 messages are further constrained and must also validate against the corresponding message conformance profile [9]. Clinical documents must be created so as to conform to the general rules defined by the CDA standard and the specific rules for the document type being created. Actor behavior conformance is an assessment of the actions taken by the actor in response to a stimulus. The behavior is assessed by examining the cumulative responses of the actor. Protocol conformance is an assessment of the protocol level standards. Message exchanges among HL7 applications must conform to the message exchange rules defined by the HL7 Minimum Lower-Layer Protocol (MLLP). CDA documents must be constructed and sent in accordance with the document exchange rules defined by ebXML SOAP or SOAP with attachments. CDA documents must be retrieved in accordance with the HTTP document retrieval rules. 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.

4 Architecture

In the previous section we described a moderately complex scenario. For the scenario to be carried out to its completion, each actor/application will have to take the correct actions in carrying out each step of the scenario. Clearly interoperability among the applications will be necessary if the scenario is to be carried out to completion, but the likelihood of applications interoperating will be increased if each application behaves correctly, i.e., correctly implements the standard and behaves in compliance with any constraints that have been imposed on the application. Thus, an architecture designed to facilitate the measurement of an application's compliance, while providing the infrastructure that enables the necessary interchanges to take place when done correctly, would provide the necessary framework for testing. Since measurements will be required that evaluate different aspects of an application's behavior against different standards, a modular architecture with independently functioning components will also be necessary; these characteristics are also necessary to ensure scalability.

The essential idea behind a Service Oriented Architecture (SOA) is that functionality is aggregated into groups for the purpose of providing a collection of well defined services. These services are made available to users over a network, allowing the services to be combined in ways that best suit the user's development requirements. This model provides an architecture for development that scales well and is both modular and flexible.

The requirements we identified above imply that an architectural model to support our testing requirements will have to include the following services: a message validation service, a document validation service, and a service for measuring conformance to the required protocols. The provision of message and document validation services is fairly straightforward. For both, a web service interface is defined. For message validation, functionality is defined that allows the user to deliver messages through the interface to an application for evaluating message conformance. The result of the evaluation is then returned to the user through the interface. The provision of a document validation service is accomplished in a similar manner; providing a service for measuring conformance isn't as straightforward.

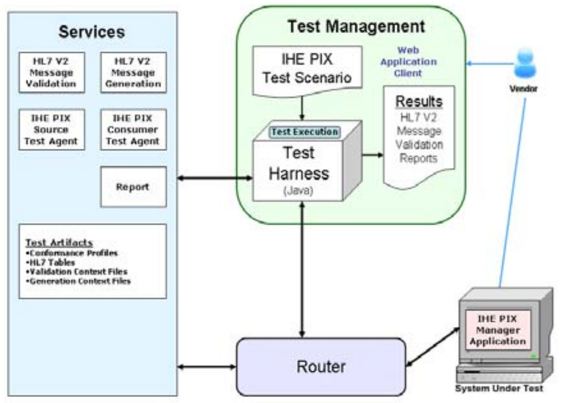
Services
HL7 V2 Message Validation
HL7 V2 Message Generation
IHE PIX Source Test Agent
IHE PIX Consumer Test Agent
Report
Test Artifacts
•Conformance Profiles
•HL7 Tables
•Validation Context Files
•Generation Context Files
Test Management
IHE PIX Test Scenario
Web Application Client
Test Execution
Test Harness (Java)
Results
HL7 V2 Message Validation Reports
Vendor
Router
IHE PIX Manager Application
System Under Test

Figure 2: Design to Test PIX Manager Application

Measuring conformance means evaluating an application's externally observable behavior for compliance to a set of rules that are determined by a standard and possibly other constraints. HL7 defines a protocol for message exchanges among applications, so adherence to the protocol rules must be evaluated. This requirement disallows use of the simple model where a user delivers the necessary data to an application through a well defined web interface. To address this problem, applications are added for the specific purpose of testing support. The testing applications are called *test agents* because they are designed to simulate the actions of the application that they will be used to test. The applications being tested are referred to as Systems-Under-Test (SUTs).

For the use case described, many testing approaches exist varying in breath and depth. At one end of the spectrum we may desire testing to support simply the validation of a CDA document only; at the other end we may seek a system that is capable of all-at-once testing; examining each transaction and aspect in the use case for every application involved (e.g., IHE Connectathon) [6]. A middle ground testing approach might be a vendor wanting to test their application in isolation. For the purpose of illustration, we consider the latter for introducing and describing a proposed test system design. We first provide this concrete example and then extrapolate the concepts to a generalized approach.

Figure 2 illustrates a system to test a PIX Manager actor—SUT. Following the concepts of an SOA the system consists of a set of services and a composition of the services. The services include message generation, message validation, and test agents; in this case a PIX Source and a PIX Consumer. The composition of the services is implemented as a web browser client controlling an underlying test harness. We have built a prototype system described in Figure 2 [5].

This system provides the capabilities to evaluate the PIX Manager's behavior. The evaluation capability is provided as well defined services that can be incorporated into the SOA model. For test agents we do this by defining a web service for accessing and controlling the actions of the test agent. In the case we our examining, the web service simply needs to support a method for deploying the test agent. Once the test agent is deployed, it can be directed to perform a given action. For example, for the first step in our use case the PIX Source test agent can be directed to send a registration message to the PIX Manager SUT. The PIX Manager SUT, in accordance with its specification will return an acknowledge message. We can evaluate that message using the HL7 message validation web service. Of course, the response that we're looking for depends on the message content we sent; this is indicated by the test script.

For each step along the way the test harness directs the test agents to interact with the SUT and can use the validation services to evaluate responses. Message generation services can be utilized to create valid messages the test agents sends to the PIX Manager SUT. The test script indicates the content of the data given to the generation service. Later in the workflow, testing of the PIX Manager will employ a PIX Consumer test agent. As described, the PIX Consumer queries the PIX Manager with one patient identifier requesting a patient identifier for the same patient in another domain. The test harness will create the query message, with assistance from the generation service. It then directs the PIX Consumer test agent to send the query message to the PIX Manager. The PIX Manager is expected to send a response message back to the PIX Consumer. The test harness now can request the message from the PIX Consumer so that it can be evaluated. The test harness has context of the use case which aids in evaluation assertions (i.e., it will know exactly the patient identifier that should be returned). The results of the evaluations can be displayed to the user via the browser interface. Following our method for evaluating the PIX Manager SUT, it is relatively clear that the approach can be used to evaluate the other actors described in the workflow. It should also be noted that the selection of the web browser client and test harness model for controlling the test is just one choice of many possible service compositions.

By using the SOA approach for providing testing services, it is possible to test each component that must be tested to complete the use case example outlined above. Figure 3 shows an architecture that can be used for solving general problems of this type.

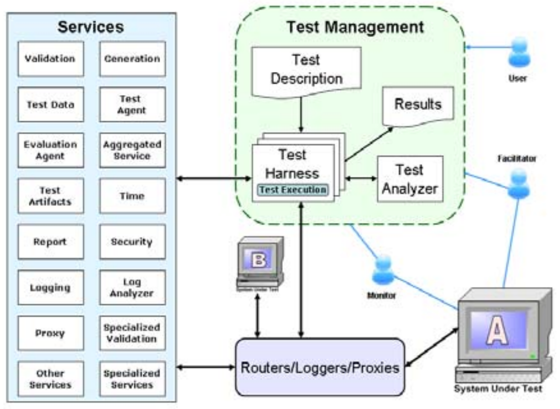

Figure 3: Testing Framework Functional Model

The architecture 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 architecture—it is likely a separate system. Services provide the testing functionality. In an SOA design, services are autonomous and hide implementation details. 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. Many instantiation choices are available—ranging from an ad hoc test driver to powerful process execution languages. The framework employs a network to route messages and may add logging and proxy capabilities. Optionally a test management system can be used to assign, manage, and track a set of tests. A certification body may utilize such a system in their process. Another important aspect of the design is the development of interfaces that connect these components. For example, a common interface between the services composition and the test agents is specified. The design supports additional test agents seamlessly through the common interface. Below we further elaborate on the architecture components.

4.1 Services

Test Agents: Test agents are implementation of actors (or applications) that support the functionality of the underlying specification of the actor. The test agent does not have to be a complete implementation; it needs to only support the functionality of the actor to support testing of applications. However, over time a test agent can become a reference implementation. Potentially hundreds of actors will exist in this framework—although for any one given test, typically five or fewer will be deployed. The distributed nature of the SOA allows for test agents to be built independently and maintained at different locations.

Generation Services: Generation services support the creation of test material such as messages and documents. For example, generation of HL7 message instances can be generated [5, 10, 11].

Validation Services: A validation service provides an evaluation function for a given protocol. For example, validation of an HL7 message instance against an HL7 conformance profile [5].

Registry/Repository Services: A repository contains testing artifacts needed in testing. The registry provides mechanisms to organize, maintain, and retrieve the artifacts. The Registry/Repository maintains the current specifications, schematron rules, profiles, test messages, vocabulary, etc. to support testing.

Additional Services: The framework also includes other services that provide various testing functionality. Additional services include logging, log analyzers, security, reporting, and more. We will elaborate further on these services in companion publications.

4.2 Test Harness

An important aspect of the SOA is the composition of the services into executable process. Services are assembled in a particular order and follow a set of rules to carry-out a business process—a test scenario in our example. Arbitrary composition of services supports broad range of test requirements in a straightforward and flexible manner. Additionally the complexity of the services composition is at the discretion of the user of the services. Modifications to test execution are handled easily in this model—allowing for quick adaptation to changing requirements in test scenarios.

The *test harness* orchestrates services to produce an executable implementation of the testing workflow. A test management control system may provide input (test scripts) that directs the actions of the test harness. The test harness carries out the instructions by employing the services. A *test analyzer* may be used to assist the test harness.
There are many choices available for implementing the test harness. These may include powerful process execution languages such as BPEL (Business Process Execution Language) [8], a web browser client with scripting language support, or a high-level programming language such as Java. The framework supports any number of test harness implementations. Operationally, multiple-independent instances of test harness implementations can execute simultaneously.

The choice of the test harness will depend on the needs of the user. For example, an IHE connectathon [6] is an event that tests interoperability among groups of applications. The tests are composed of a choreographed sequence of steps; BPEL may be an appropriate technology choice for this testing need [7, 8]. BPEL is an XML-based scripting technology for generating a visual representation and driving a business process. It can orchestrate services to produce an executable implementation of a testing process. Another case is testing in isolation; a vendor may want to beta test an application on-site. This testing can be conducted over the Internet and driven simply by ad hoc Java code. A more sophisticated user interface may employ a browser-based client that can direct the test, either in a well-defined test sequence or in an ad hoc manner.

Services may be aggregated. For example, an implementation of a workflow may become a service itself. Services may use other services—test agents may choose to invoke a validation service directly. Other utilities for the support of test harness instances include templates—for example a web browser client. This will allow for uniform creation of web sites for a particular set of tests; for example IHE pre-connectathon tests. This could be applied at the organization level or at a testing event level.

4.3 Network Functions

Network functions include routing messages and documents between the test agents and SUT. Additional capabilities may include logging and a proxy or a set of proxies that can be used to further facilitate the capturing and analysis of test data. We are currently exploring where logging and proxy capabilities fit best in the architecture—either as services or as network functions, or possibly both depending on the testing requirements. The use of a proxy or a related NIST design concept, an *evaluation agent*, is necessary when conducting interoperability testing—i.e., concurrently testing a group of communicating SUTs. We are currently exploring the design alternatives.

4.4 Test Management Systems

As mentioned, the test management system is not part of the testing architecture but is an important ancillary system. The management system is responsible for setting up test cases and reporting and maintaining results. Auditing services can also be included. Certification and testing bodies such as the Certification Commission for Healthcare Information Technology (CCHIT), Nationwide Health Information Network (NHIN), and IHE will likely employ a test management system.

5 Testing Framework

It is clear that the architecture described is extensible and flexible providing a framework that can support an expansive set of testing requirements. The user is afforded the flexibility to select the interface and environment in which they want to conduct testing. It is extensible since services can be added and the composition of services can be built independently. Additionally the framework scales well since the set of services are independent and services composition and test execution can occur spatially.

The test framework supports rapid test suite development which can be built independently and simultaneously. Stakeholders, such as certification bodies or vendors testing in-house, can build the testing system that meets their requirements. Pre-built public domain test harnesses will be made available; these provide "*off-the-shelf*" testing tools for well-known test scenarios. For example, IHE pre-connectathon PIX actor tests; tests that must completed before vendors attend the connectathon event. Templates will also be provided to accelerate the development of third party implementations of test harnesses. A template will provide the infrastructure and an example so that similar test harnesses can be built. For example, a web browser client template test harness will be provided to support HL7 V2 actor pre-connectathon tests. The template will have the same look-and-feel interface; a library of tests can be built for each of the many IHE domains.

To elaborate further, a browser-based testing client template can be built for all IHE connectathon PIX tests, or CCHIT lab testing, or IHE patient care devices (PCD) pre-connectathon testing. Each group could develop its own test harness. However it would be best if they use a common look and feel, therefore templates should be used.

6 Conclusion

We have presented a testing architecture based on the SOA model that allows the user to employ and combine testing services in arbitrary ways. This approach gives the user the capability to formulate problem solutions based on the most effective use of the available services. We have examined in some detail a specific use case. By developing an evaluation strategy based on the SOA model, we were able to demonstrate how the systems and applications employed in the use case example could be effectively evaluated. Furthermore, we have left open the possibility of combining these services in different ways to formulate other solutions to the problem. In our future work, we plan to further investigate the effectiveness of the solution proposed in this paper and to explore other approaches, possibly expanding the set of services supported to include new services that have not yet been considered in our examination thus far of the problem of evaluating multiple, interconnected disparate applications.

7 References

[1] Health Level 7 (HL7) Standard Version 2.5, ANSI/HL7 V2.5-2003, June 26, 2003, http://www.hl7.org.

[2] Digital Imaging and Communication in Medicine (DICOM); http://medical.nema.org.

[3] Integrating the Healthcare Enterprise (IHE); http://www.ihe.net.

[4] IHE IT Infrastructure (ITI) Technical Framework Integration Profile, December 12, 2008.

[5] NIST HL7 V2 Testing Tools; http://hl7v2tools.nist.gov

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

[7] Gazelle Testing Framework Project. Gazelle is a work-in-progress system targeted for the IHE connectathon testing. Managed by Steve Moore (Washington University of St. Louis-MIR) and Eric Poiseau (INRIA). MIR, INRIA, NIST and others are developing the system in a joint effort.

[8] BPEL http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[9] *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.

[10] "*Dynamically Generating Conformance Tests for Messaging Systems*" R. Snelick, L. Gebase, S. Henrard. 2006 Software Engineering Research and Practice (SERP06), WORLDCOMP'06 June 26-29, 2006, Las Vegas, NV.

[11] "*Conformance Testing and Interoperability: A Case Study in Healthcare Data Exchange*" L. Gebase, R. Snelick, M. Skall. 2008 Software Engineering Research and Practice (SERP08), WORLDCOMP'08 July, 2008, Las Vegas, NV.

[12] The Clinical Document Architecture (CDA), ANSI/HL7 May 2005; http://www.hl7.org.