Conformance Testing and Interoperability: A Case Study in Healthcare Data Exchange

L. Gebase[1], **R. Snelick**[1], **and M. Skall**[1]
[1]National Institute of Standards and Technology (NIST), Gaithersburg, MD, State, USA

Abstract - *Correct data exchange is critical for ensuring reliable healthcare systems. Standards based systems are the foundation for achieving this goal. However, standards alone are not enough to ensure this promise; conformance and interoperability testing are essential. We present and compare conformance testing strategies for a widely used healthcare clinical data exchange messaging standard. We discuss in detail an actor-based testing framework and give insight on the approach used in developing the framework. We present an architecture that extends this framework to support testing of integrated healthcare systems using multiple messaging and document data exchange standards.*

Keywords: Conformance; Interoperability; Messaging Systems; Testing Framework; Test Strategies.

1 Introduction

A major challenge for the healthcare industry is achieving interoperability among proprietary applications provided by different vendors. Each hospital department may use multiple applications to share clinical and administrative data. Interoperability can be better achieved through the use of standardized interfaces. Even though the applications may implement the same standard, there is no assurance of interoperating. There are two primary reasons for this problem. One is that the applications don't implement the same set of options allowed by the standard. This problem can be addressed with conformance provisions offered by the standard. The second problem is that applications implement the standard incorrectly. This is addressed with conformance testing. Applying conformance processes and successfully conducting conformance testing will not ensure interoperability, but they will increase the likelihood of implementations interoperating. Employing a comprehensive testing program at the onset of an implementation leads to more reliable systems, and ultimately, reduced costs.

We propose and examine testing strategies for the Health Level Seven version 2.x (hereafter HL7) messaging standard [1]. HL7 is a widely used standard for the exchange of clinical and administrative data among healthcare applications. HL7 provides an interesting testing challenge due to the wide array of options allowed by the standard. To reduce the number of choices implementers are confronted with and increase the likelihood of different implementations interoperating, the HL7 standard has introduced a conformance section that allows implementers to support a subset of the functionality offered by the standard. By reducing the large set of options allowed by the standard, implementers are able to significantly increase the likelihood of interoperating. The principle mechanism to constrain the allowed set of options is a message profile. Message profiles not only aid interoperability, they also enhance the capabilities and effectiveness of conformance testing and the overall testing process.

We are interested in establishing conformance metrics for HL7 implementations and evaluating vendors' adherence to those metrics in a pragmatic environment designed to simulate a real world environment that does not require changes to the vendor implementation. We examine two approaches for evaluating conformance. One approach employs an Upper Tester, which sits above the application being tested and makes use of whatever user interface—possibly an application programming interface (API)—that the application provides, along with a Lower Tester which acts as a peer application and drives the testing. The second approach we examine employs actors to interact with the application being tested. Actors are autonomous, relatively small modules, generally run on separate execution threads that support a well defined subclass of the total functionality defined by the standard. Finally, while our initial focus is on HL7 testing, our objective is to develop tools and methodologies that can readily be applied more broadly to environments outside of HL7.

2 Conformance and Interoperability

Standards, no matter how good they are, are just pieces of paper. They are a means to an end. The goal of any standard is the eventual *binding* of the requirements in the standard into correct, reliable software. To accomplish this goal, the standard must be a clear, precise, unambiguous, complete and testable enumeration of detailed requirements. Using the English language to provide this detailed specification is a challenge in itself because English is not a precise language and lends itself to ambiguities. There are, however, principles that one can implement to help ensure a precise, testable standard. A good standard should 1) define what/who needs to implement the standard, 2) distinguish between normative (mandatory) and informative sections of the standard, 3) use universally accepted key words to specify requirements, 4) be modular with minimal redundancy, 5) be adaptable as things change, and 6) be technology and design-independent. If a standard encapsulates these principles it stands a good chance of being implemented correctly. However, in order to substantially increase the likelihood of developing correct implementations, tests need to be developed to determine conformance.

Conformance is defined as the fulfillment of a product, process, or service of specified requirements [6]. Conformance is essential to any standard because it specifies who needs to conform to the standard and what they need to do to claim conformance.

Conformance testing is a way to determine directly or indirectly that all relevant requirements in a standard have been implemented correctly. Conformance testing is ***black box*** testing. In black box testing, the tester does not have knowledge of the implementation's internal structure or have access to the source code. A tester examines that requirements have been met by probing the implementation through a series of test cases comprised of both valid and invalid input and examines the output for correctness, as defined by the standard. This is contrasted with ***white box*** testing where the internal structure of the code is known to the tester. With white box testing, the tester chooses inputs that exercise paths through the code in order to determine if the implementation is working correctly.

There is a relationship, much like a three-legged stool among standards, implementations, and conformance testing. If one leg of our stool does not work correctly, we will not have confidence that our requirements have been faithfully implemented. The implementation is tested (via conformance testing) against the requirements in the specification to determine if all requirements are met. There are only two possible outcomes. If any of the tests result in at least one error, then we know to a certitude that the implementation does not conform. However, ironically, if the implementation passes all of the tests, we don't know anything for certain. Either the implementation does indeed conform or the tests are not comprehensive enough to find the non-conformity. This is another way of stating that conformance testing can never be exhaustive. Conformance testing can only prove the presence, not the absence, of errors.

The goal of interoperability testing, on the other hand, is to ensure that diverse systems can "work together" and thus interoperate. In the world of messaging standards, interoperability will result in implementations reliably exchanging messages without error. Note that conformance testing is a pre-requisite for interoperability testing since we need to ensure that the information being exchanged is the correct information. However, interoperability testing requires another layer of testing *after* conformance has been ascertained. Interoperability plus conformance ensures that both systems are *speaking* the same language. Systems can send and receive messages and respond with appropriate messages and ultimately incorporate the information into their systems and workflow.

Conformance testing is often (but not always) performed by testing laboratories, with a resulting issuance of a certificate to implementations that pass all the tests. This process is called certification. However, even if certification is not the goal, conformance testing is still necessary, since it is the only way to ascertain if requirements in the standard have been correctly implemented. Additionally, conformance testing serves as a communication between buyers and sellers allowing sellers to substantiate their claims and buyers to increase their confidence in the product.

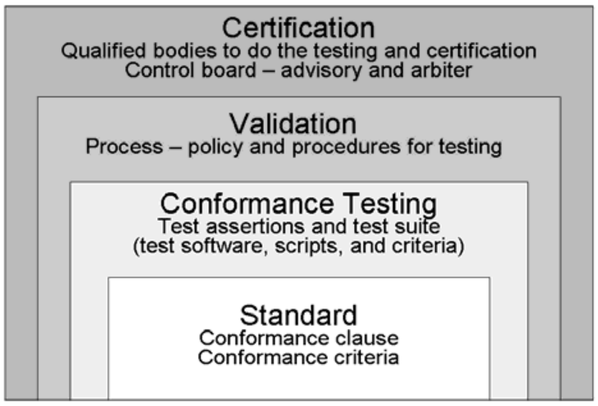
Certification
Qualified bodies to do the testing and certification
Control board – advisory and arbiter
Validation
Process – policy and procedures for testing
Conformance Testing
Test assertions and test suite
(test software, scripts, and criteria)
Standard
Conformance clause
Conformance criteria

Figure 1: Certification Building Blocks

There is a relationship among the standard, conformance testing, conformity assessment and certification as a set of inter-connected building blocks much like a Russian nested doll with the standard as the inter shell (see Figure 1). None of the higher-level blocks can be performed unless the box beneath it has been completed. Thus, conformance testing can not be performed unless the standard (with its conformance clause and clear, testable requirements) has been completed. Conformity assessment (processes and policies for testing) can not be implemented until the standard and the conformance testing test suite are in place. Finally, certification can only be accomplished when all of the three lower level building blocks are in place. Also, one can stop anywhere along this spectrum. Many standards exist without conformance testing or certification. Some standards have associated conformance tests but no certification regime while some standards contain all the components all the way up through certification.

3 Testing HL7 Healthcare Systems

Typical healthcare organizations have many proprietary heterogeneous information systems that must exchange data reliably. Not only are the systems heterogeneous but standards for exchanging data among them are different. Seamlessly sharing data among systems and testing them is complex. In this section we focus on homogeneous systems for the exchange of clinical data using the HL7 messaging standard. We provide an overview of HL7 version 2 and our conformance approach, an analysis of testing strategies, and tools that facilitate testing and interoperability. We then focus on an actor-based testing framework. In the section that follows, we extend the framework to a heterogeneous healthcare system that uses multiple message types and employs more than one document exchange standard.

3.1 HL7 and Conformance

The Health Level Seven (HL7) version 2.x 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. HL7 messages are structured hierarchically, but the hierarchy is limited to exactly four levels and composed of building blocks generically called *elements*. These elements are *segments*, *fields*, *components*, and *sub-components*. Each element has associated attributes that may constrain it. These include the degree of options allowed, repeatability, value set, length, and data type attributes. Segments can contain additional elements, fields and components can contain additional elements or be primitive elements; sub-components are strictly primitive elements. Primitive elements are those that can hold a data value and have no descendant structure. Additionally, a container element called a *group* can be used to group a related collection of segments.

This four-tiered hierarchical structure appears simple enough, but the real complexity in the message structure is revealed when the possible sequence of segments and fields making up a message is examined [8]. Every HL7 message can be identified by its message type. The type limits the segments allowed in the message, but generally, even for a specific message type, a great deal of variation is possible. Segments may be designated as required or optional; required segments may also be allowed to repeat an arbitrary number of times, or they may be required to repeat a specified number of times. Optional segments may be absent or they may be present and repeat an arbitrary number of times. For any message, a segment present in one instance of the message may not be present in another; repeating segments may occur multiple times in one instance and not at all in another. The message content is further complicated by the fact that the sequence of fields making up each segment may themselves be optional or required and also may or may not repeat. An application capable of processing messages of one type may be incapable of processing messages of a different type, and an application capable of processing a specific message type may not be able to process all instances of the type. Clearly the realm of message possibilities is large and for applications to have a reasonable chance of interoperating, the spectrum of possibilities has to be constrained. The conformance section of the HL7 standard has defined a *message profile* (also commonly referred to as conformance profiles or profiles) for precisely this purpose.

```
…
<Segment Name="PID" LongName="Patient identification"
Usage="R" Min="1" Max="1">
   <Field Name="Set ID - PID" Usage="R" Min="1" Max="1"
Datatype="SI" Length="4" ItemNo="00104">
   </Field>
…
    <Field Name="SSN Number - Patient" Usage="X" Min="0"
Max="*" Datatype="ST" Length="16" ItemNo="00122">
   </Field>
   <Field Name="Driver's License Number - Patient" Usage="R"
Min="1" Max="1" Datatype="DLN" Length="250"
ItemNo="00123">
      <Component Name="Driver's License Number" Usage="R"
Datatype="ST" Length="100">
      </Component>
      <Component Name="Issuing State, province, country"
Usage="R" Datatype="IS" Length="10" Table="0333">
      </Component>
      <Component Name="expiration date" Usage="R"
Datatype="DT" Length="30">
      </Component>
   </Field>
…
```

Message profiles define processing rules and provide an unambiguous description of HL7 messages. Vendors agreeing to a common profile are more likely to interoperate. Furthermore, the profile provides a measure for evaluating the validity of the messages exchanged among vendors. Vendors may employ tools specifically designed to facilitate message validation. This may be the first step in the overall process of evaluating conformance. A message profile can be represented as an XML document (see the profile snippet shown). The document includes each element allowed in the message along with its associated attributes. For a more detailed description of a message profile refer to version 2.5 of the HL7 standard [1].

Profiles reduce the number of possibilities to a manageable set, and their use helps to ensure that systems attempting to communicate with each other implement compatible sets of possibilities. A profile defines a set of constraints on the options allowed by the standard. When the profile is specified in XML, it also may be machine processed, thereby greatly facilitating the effort required to produce an implementation and reducing the likelihood of errors.

3.2 HL7 Conformance Testing

Conformance testing focuses on evaluating an implementation's external behavior and assessing its adherence to the standard. For HL7 implementations, assessing external behavior generally encompasses determining the implementation's state and evaluating the content of the data it transmits in its current state. Messaging protocols are typically stateless, but nevertheless an HL7 application can be treated as simple state machines. Initially a responding HL7 application is in a wait state, ready to receive HL7 messages from an initiating application. On receipt of a message, it transitions to a send state in which an acknowledgement message must be returned to the initiating application. An initiating application reverses the responding application's state transitions and is initially ready to send and then transitions to a wait state.

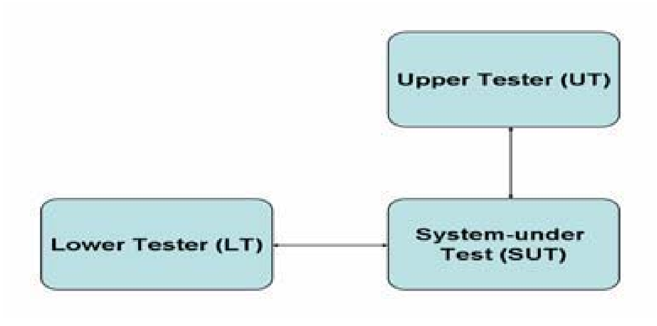

Figure 2: Classic Test System

To evaluate a responding application, it is necessary to measure the content of messages it receives and the acknowledgement message it returns. To ensure the content of sent messages is correct, the messages must first be validated. Validation encompasses message parsing to ensure correct structure and syntax and semantic checking to ensure values are correctly constrained. Criteria that must be satisfied by the returned message can be established from the message that was sent. However, the content of the returned message may also depend on the state of the responding application's database. To track the content of the database, it is necessary to control the initialization of the database and then track changes to it.

Evaluating the behavior of an initiating application largely reduces to validating the messages sent by the application. But when testing both initiating and responding applications, the ability to measure the content of messages sent by an initiator may be limited. In general the exact content of the message cannot be determined without knowing the user request that triggered the sending of the message, but without access to the application user's interface, this is often not possible. Nevertheless, in the case of HL7, message validation with some limitations can still performed. This is possible if the HL7 message profile is used to enforce proper message structure and syntax and to enforce value constraints. It is also possible to conduct more robust testing by constructing messages with invalid values, or with an invalid structure, and evaluating an implementation's reaction to the receipt of the invalid messages.

3.3 Testing with a Lower and Upper Tester

One commonly employed approach to black box testing is to place the implementation being tested—commonly referred to as the system-under test (SUT)—between a so called Upper Tester (UT) and Lower Tester (LT). Figure 2 illustrates the approach.

The approach has been widely employed in conducting protocol testing. With this approach, the SUT communicates with the LT via the protocol defined by the specification and the UT takes the place of the user or the business application supported by the SUT. No additional requirements are placed on the SUT to enable communications with the LT, since the environment does not differ from the operational environment the SUT would otherwise function in. The LT drives the testing. Acting as an initiating application, the LT sends messages to the SUT and evaluates the SUT's behavior based on the acknowledgment messages returned from the SUT. The UT may be used to evaluate the SUT's service interface. The LT may also direct the UT to issue requests to the SUT and in this way the SUT's role as an initiating application can be evaluated.

This approach allows for effectively evaluating the externally observable behavior of the application, but places no requirements on the internal structure of the application or the methods it uses to satisfy the requirements.

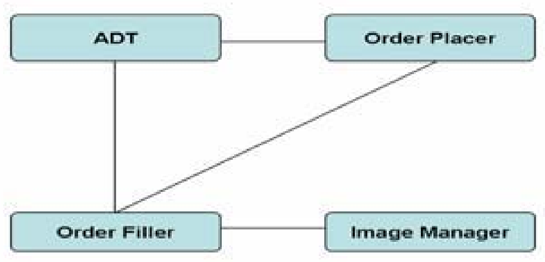

Figure 3: Typical HL7 Environment

Employing a LT to replace a peer application in this environment can usually be achieved without difficulty, but the same is not true for employing an UT. An UT must typically be deployed outside the tester's local environment, in the environment running the SUT. This places requirements on the SUT that may not be easy to achieve, particularly when a standardized user interface for the SUT is not defined, as is often the case. Testing can still be conducted without the use of an UT, but it does place some limitations on the capabilities of the test system and what can be tested.

A further limitation of this form of black box testing is that it cannot be applied in an environment made up of multiple communicating applications, nor can it be used if there are multiple systems to be tested simultaneously.

3.4 Actor Based HL7 Test Framework

A typical HL7 environment is made up of many communicating HL7 applications. A representative environment is show in Figure 3.

Actors are autonomous implementations, typically running on separate execution threads that the testing framework employs in place of HL7 applications needed to simulate the operational environment in which the SUT functions. In general HL7 actors can serve in place of any HL7 system. This enables the construction of an environment completely controlled by the test system that mimics the real world operational environment in which the SUT operates. In this environment the test system can track and monitor all message exchanges with the SUT. The actor based architecture is shown in Figure 4. To the SUT and any other HL7 implementation that is part of the testing environment, the actors are indistinguishable from HL7 applications they might interact with in an operational environment. The actors are distinguishable from other HL7 systems only in that they are driven by a test script and provide complete logging of all activities.

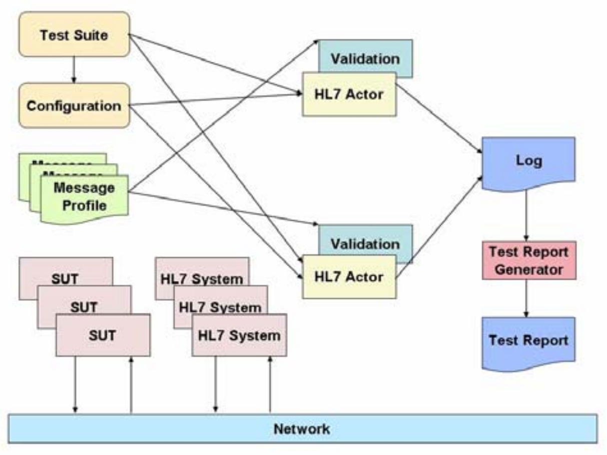

Figure 4: Actor Based Testing Environment

Figure 5 depicts an example environment in which an HL7 Order Placer System is to be tested. The operational environment of the Order Placer includes ADT, Order Filler, and Image Manager HL7 systems. The testing environment consists of the SUT and three actors that carry out the same functionality generally supported by the other systems making up the operational environment. Figure 5 shows a scenario in which message transactions begin with the ADT actor sending a message to the Order Placer. The receipt of this message by the Order Placer SUT is expected to trigger a series of message exchanges. The Order Placer must acknowledge the message sent by the ADT actor, and in addition it is expected to send a message to the Order Filler. But there are no differences between the Order Filler actor and the Order Filler itself that are apparent to the SUT. Thus, no changes are necessary to test the Order Placer.

The messages exchanged in this environment are constrained by a message profile that all participating entities agree to. The profile restricts the set of messages exchanged from the broad spectrum allowed by the standard to a manageable set.
This actor based methodology to testing offers some advantages over the Upper-Lower Tester approach. It is easily extensible; regardless of how many applications are employed in the operational environment, actors can always be employed to replace them in the testing environment. The approach also lends itself better to deployment in more complex environments, such as the environment depicted in Figure 4 where more than one SUT is being tested simultaneously. Actors and applications may be mixed arbitrarily in this environment, which cannot be readily accomplished in the Upper-Lower Tester environment.

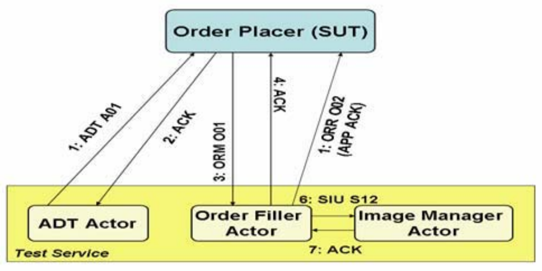

Figure 5: Typical HL7 Testing Scenario

Although message validation is not strictly required prior to performing conformance testing, validating messages in advance can greatly facilitate the process. If it is not performed in advance, it must be conducted simultaneously with the testing. Evaluating message content is critical to accurately conducting conformance testing. If the test system is to evaluate the message responses returning from an HL7 system, it cannot do so accurately unless it is certain of precisely what the content of each message is.

The testing framework does not vary depending on whether or not message validation has been done in advance. However, message validation can be turned off if it was done previously, and, if it is employed, testing should proceed smoothly when validation has been done in advance.

As with the black box testing described above, testing with actors is used to evaluate only external behavior. The approach does not employ any counterpart to an UT and this places some limitations on what can be tested; messages sent and received can be evaluated, but evaluating the functionality provided to the user or the business application cannot be done. Without a counterpart to an UT, triggering an HL7 implementation to initiate a message exchange can be problematic; it may be possible as the result of receiving a message, but short of this some means of accessing the application's service interface is necessary.

3.5 Message Profiles and Interoperability

A message profile applies implementation specific constraints to the standard that eliminate the potential ambiguities that the standard permits as implementation alternatives and thus increase the likelihood of implementations interoperating [3]. Message profiles are an integral part of a testing framework. They provide the message template upon which better test message generation and message validation can be performed.

A desktop tool for creating and documenting message profiles in a common format is the Messaging Workbench (MWB) [2]. The MWB supports all the HL7 version 2.x artifacts in the form of libraries that are readily available within the tool for use in message profile composition. An XML representation of the message profile is an important output of the tool.

An important aspect for achieving interoperability is determining if communicating applications have correctly implemented an interface based on a message profile. To achieve this, the existence of a well-defined and extensive set of test messages is paramount. At NIST, we have developed utilities for message generation that can be incorporated into a testing environment. The utilities are delivered as a collection of APIs, web services, and as a desktop application called Message Maker [3,4,5].

A critical function of testing is message validation utilities. NIST has developed tools to validate messages instances based on a message profile and has extended functionality to support content testing based on a given test scenario. A Java message validation API provides the core functionality. Additionally the functionality has been built into a desktop application, web services [9], and a web application [10]. The APIs and web service APIs can be used to integrate validation services into a testing framework.

4 Extending the Testing Framework

We have described an approach for assessing the conformance and interoperability of HL7 healthcare systems utilizing a set of actors designed to simulate HL7 application behavior. However, healthcare organizations exchange data using a number of messaging and document standards—since there is not a single standard to cover all aspects of data exchange among healthcare systems. Some standards are needed for moving clinical data, others for medical images, and yet others for personal health records, for example. A testing framework can be used to evaluate systems with complex integration requirements. The *Integrating the Healthcare Enterprise* (IHE) initiative [7] has defined numerous integrated test scenarios for various healthcare domains (e.g., radiology). IHE hosts an annual connect-a-thon event [12] where numerous vendors implementations are evaluated using the testing scenarios. The IHE Gazelle project [13] is an effort to develop a testing framework to automate the testing of the integrated test scenarios. The Gazelle framework extends the actor based testing approach to a heterogeneous environment.

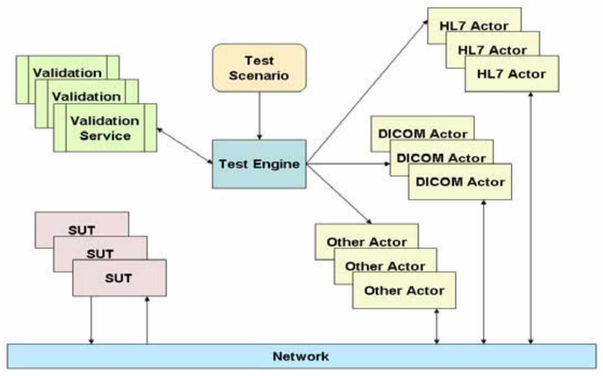

Figure 6: Gazelle Testing Architecture

The IHE environment is a heterogeneous environment designed for exchanging different message types in an environment employing multiple messaging protocols. The Gazelle architecture is shown in Figure 6. The core of the Gazelle test system is the test engine which is responsible for orchestrating message exchanges among the diverse messaging systems. The Gazelle system aggregates the multiple homogeneous systems into a single, complex whole heterogeneous system. A set of HL7 actors make up one homogeneous system, DICOM [14] actors can make up another, and other healthcare data exchange protocols can make yet others. The SUT may be an HL7 system, or it may be a DICOM system; it's possible to test both HL7 and DICOM systems in this environment. Message exchanges within the homogeneous set of HL7 actors proceed as they do when deployed in a strictly HL7 environment; DICOM exchanges proceed in a similar way, but exchanges may also be required that bridge the two protocols. This requires actors capable of processing both HL7 and DICOM messages and presents a challenging problem in itself. The solution employed hinges on implementing a satisfactory translation between DICOM and HL7 messages.

Since the Gazelle system is made up of multiple diverse systems with no common protocol, the test engine is faced with the problem of how to communicate with each system without requiring support for a separate communications technique for each system that is part of the environment. To deal with this problem, a web service interface is employed. Each actor in the system supports a web service interface that the test engine uses to communicate with the actor. The challenge in defining the web service interface is to define a suitable interface that is common to all actors, rather than employing different definitions for each type of system. Since the common thread running through each homogeneous system is that it is actor based, it is possible to abstract the requirements so that a single interface will serve the requirements for all systems. Clearly doing so not only simplifies development of the test engine, it also means that it can readily be extended to incorporate any number of new systems.

5 Conclusions

We have analyzed two techniques for conducting conformance testing, one employing an Upper Tester and Lower Tester, the other actor based. We have shown that the Upper-Lower Tester method can be effectively utilized in conducting conformance testing, but that the method does not scale well to environments incorporating multiple applications or requiring multiple systems to be tested simultaneously. For these systems we have shown that actors—autonomous systems providing limited, but well defined functionality—can be effectively employed for testing. Moreover, we have shown that the actor based approach can be readily extended beyond the HL7 environment. We have also shown that properly conducting conformance testing requires careful evaluation of message content and that while conformance testing cannot ensure that implementations that undergo successful conformance testing will interoperate, it will increase the likelihood of them doing so.

6 References

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

[2] Messaging Workbench (MWB). Developed by Peter Rontey at the U.S. Veterans Administration (VA) in conjunction with the HL7 Conformance Special Interest Group; http://www.hl7.org.

[3] *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.

[4] Message Maker; Developed by the National Institute of Standards and Technology (NIST) in conjunction with the HL7 Conformance Special Interest Group; http://www.nist.gov/messagemaker.

[5] "Dynamically Generating Conformance Tests for Messaging Systems" Robert Snelick, Len Gebase, Sydney Henrard. 2006 Software Engineering Research and Practice (SERP06) conference proceedings, WORLDCOMP'06 June 26-29, 2006, Las Vegas, Nevada.

[6] ISO Reference - ISO/IEC 17000 Conformity assessment - Vocabulary and general principles, first edition 2004-11-02.

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

[8] "Selecting Effective Test Message" Len Gebase, Roch Bertucat, Robert Snelick. 2006 Software Engineering Research and Practice (SERP06) conference proceedings, WORLDCOMP'06 June 26-29, 2006, Las Vegas, Nevada.

[9] NIST HL7 Message Validation Web Services. http://hl7v2tools.nist.gov.

[10] NIST HL7 Message Validation Web Application. http://hl7v2tools.nist.gov.

[11] ISO Reference - ISO/IEC 17000 Conformity assessment - Vocabulary and general principles, first edition 2004-11-02

[12] IHE Connect-a-thon http://www.ihe.net/Connectathon/index.cfm

[13] Gazelle Testing Framework. A collaboration effort led by S. Moore (Washington University of St. Louis) and E. Poiseau (INRIA) to build a testing framework to support IHE test scenarios.

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