Towards Interoperable Healthcare Information Systems: The HL7 Conformance Profile Approach

R. Snelick[1], P. Rontey[2] L. Gebase[1] L. Carnahan[1]
[1] Information Technology Laboratory, National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899, USA
{rsnelick,lgebase,lisa.carnahan}@nist.gov
[2] VHA Office of Information, Messaging and Interface Services, U.S. Veterans Administration (VA), Albany NY 12208, USA
peter.rontey@va.gov

Keywords: conformance testing, healthcare, interoperability, message profile, messaging systems

1 Abstract

Typical healthcare organizations have many proprietary heterogeneous information systems that must exchange data reliably. Seamlessly sharing information among systems is complex. The widely adopted HL7 version 2 messaging standard has helped the process of systems integration. However, using the HL7 standard alone does not ensure system interoperability. The HL7 standard offers a wide range of options. Trading partners, without prior agreement, are not likely to implement options that are compatible. As a result, interoperability is stifled and organizations are left to employ their own ad hoc solutions. Conformance message profiles provide the solution to this problem. Message profiles define a standard template that provides a precise definition of the data exchanged between applications in a common format. Defining a set of message profiles for controlling message exchanges establishes a well defined communications interface among organizations and facilitates interoperability. We present a methodology based on message profiles for defining, implementing, and testing HL7 interfaces. We demonstrate how the use of message profiles improve system interoperability and present a collection of tools that we have developed to facilitate the use of message profiles.

2 Introduction

A major challenge for the healthcare industry is achieving interoperability among proprietary applications provided by different vendors. For example, each hospital department may use one or more applications to share clinical and administrative information. Each application may support multiple communications interfaces that must be modified and maintained. This is a difficult way to achieve interoperability. Alternatively, interoperability can be achieved through the use of standardized interfaces. Standardized interface definitions can remove the cost of building a separate interface for each associated application. Application developers can build applications that conform to the standardized interface definition, increasing the likelihood of interoperability and reducing cost. Maintenance cost is also reduced because the number of interfaces to maintain decreases.

The Health Level Seven (HL7) *Application Protocol for Electronic Data Exchange in Healthcare Environments* Version 2.x standard is the de facto standard for moving clinical and administrative information between healthcare applications [1]. The standard is based on the concept of application to application message exchange. An HL7 *message* is an atomic unit of data transferred between systems [1]. Typical HL7 messages include admitting a patient to a hospital or requesting a lab order for a blood test. HL7 describes an *abstract message definition* for each real world event (*e.g.*, admitting a patient). The abstract message definition is comprised of a collection of segments in a defined sequence. Rules for building an abstract message definition are given by the HL7 message framework. The message framework is hierarchical in nature and consists of building blocks generically called *elements*. These elements are *segment groups*, *segments*, *fields*, *components*, and *sub-components*. Each element has associated attributes that may constrain it. These include optionality, repeatability, value set, length, and data type attributes. Segments and segment groups 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.

When originally developed, HL7 was designed to accommodate the many diverse business processes that exist in the healthcare industry. This universal design was necessary to gain broad industry support. However, such broad accommodations resulted in a standard with many optional components—aligning interface implementations was difficult.

HL7 applications are generally connected in two ways, point-to-point and via middleware as typically supplied by interface engine products. Point-to-point entails connecting each pair of applications independently of other applications. In the interface engine approach, all applications are connected to a centrally located message broker. A set of HL7 message definitions specifies the requirements between the communicating applications. Although the message definitions are very specific there are many ways to specify a given HL7 transaction. In practice, the vendor-provider specifications may not quite match. The differences need to be accounted for in each connection. In point-to-point architectures each new combination will require a separate implementation. In interface engines, a new

mapping transformation definition needs to be defined. In both cases, the breadth of the specification leads to cumbersome and ad hoc interface implementations. System implementations such as these are prone to error, difficult to maintain, and do not scale easily.

To help alleviate shortcomings, the standard through the work of the HL7 Conformance Special Interest Group (hereafter Conformance SIG) introduced the concept of *conformance message profiles* (also commonly referred to as conformance profiles, message profiles, or profiles) in version 2.4. Message profiles define processing rules and, by defining exactly which optional components in the standard a message might include, provide an unambiguous description of HL7 messages. Message profiles also document how messages are used by identifying use case scenarios and as of version 2.6 the coded vocabulary used with many message elements.

We present a methodology based on message profiles for defining, implementing, and testing HL7 interfaces. We demonstrate how the use of message profiles improve system interoperability and present a collection of tools that we have developed to facilitate the use of message profiles.

3 HL7 Conformance Profile Defined

Message profiles constrain HL7 message structure and vocabularies and define dynamic interactions. Message profiles document a use case model, a static definition, and a dynamic definition for a message. The use case model provides a description, defines actor responsibilities, and describes a sequence of actions performed by the sending and receiving applications. The dynamic definition describes the interaction between the sender and receiver in terms of the expected acknowledgments. The static model provides a precise definition of the message structure. A message profile can be represented as an XML document, see the profile snippet shown in figure 1. Each element in the message profile is listed along with its associated attributes. For a more detailed description of a message profile refer to version 2.5 of the HL7 standard [1]. It is important to note that the attributes and the constraints a profile places on a message provides a clear and unambiguous definition, thereby, facilitating the design, implementation, and testing of interfaces. Next a summary of the message profile structure and constraints is presented.

The rules for constructing a message are described by the message framework [1]. In addition, for each real world event, for example “Admitting a Patient”, a specific abstract message type (in this case ADT_A04) is defined. The message type defines a template or structure that the message must follow; it defines explicitly the elements and the order the elements must appear in a message instance. For example, in Figure 1, the “PID” segment contains the field “Set ID – PID”, and so on. The usage attribute refers to the circumstances in which an element appears in a message [1]. For example, the “Driver’s License Number” component in the profile snippet is required (Usage=”R”) and must be present in a valid message instance. Cardinality refers to the minimum and maximum number of repetitions an element may have [1]. An example of an element cardinality is

[0..1]; the element may not appear in the message instance, but can only have one occurrence if it does. A table of allowable values can be defined and associated with a certain element. For example, see the “Issuing State, province, country” component in Figure 1; this element must be populated with a data value that is defined in table 0333. The length attribute defines the maximum allowable length a value can have for a particular element. The data type defines the allowable data values an element can contain. For primitive data types, such a numeric (NM) interpretation is straightforward and requirements for each data type are specified in the standard [1]. Complex data types, such as the Extended Person Name (XPN), may be composed of primitive types or other complex data types. For example, an XPN contains a family name (FN), which itself is a complex data type that is composed of five primitive elements, all of type string. All complex data types are ultimately composed of primitive data types. A proposal for constraining data content is under consideration.

```
…
<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">
     <Reference>3.4.2.1</Reference>
   </Field>
…
    <Field Name="SSN Number - Patient" Usage="X" Min="0" Max="*" Datatype="ST" Length="16"
ItemNo="00122">
     <Reference>3.4.2.19</Reference>
   </Field>
   <Field Name="Driver's License Number - Patient" Usage="R" Min="1" Max="1" Datatype="DLN"
Length="80" ItemNo="00123">
     <Reference>3.4.2.20</Reference>
     <Component Name="Driver's License Number" Usage="R" Datatype="ST" Length="40">
     </Component>
     <Component Name="Issuing State, province, country" Usage="R" Datatype="IS" Length="2"
Table="0333">
     </Component>
     <Component Name="expiration date" Usage="R" Datatype="DT" Length="30">
     </Component>
   </Field>
…
```

Fig. 1. Snippet from a Message Profile

A message profile is distinguished from a specification in that by application of the conformance rules, the ambiguity permitted by the base standard is removed to such an extent that the interface specified by the profile may be directly implemented. The HL7 standard provides a large number of ways to define an interface, 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. It is important to recognize that profiles don't eliminate possibilities allowed by the standard; they select a specific group from the total set of those allowed. In this regard, a profile comprises a constraint on the standard, such that the resultant constrained specification may be used to

implement the interface. The profile also imposes a discipline upon the interface partners that ensures harmony in the actual implementation.

4 Achieving Interoperability with Conformance Profiles

A message profile applies implementation specific constraints to the standard that eliminate the potential ambiguities that the standard permits as implementation alternatives. Irrespective of the underlying architecture, be it point-to-point or brokered through middleware, profiles give organizations a better way to manage system integration. Profiles can be used to directly implement an interface. All participants in the interface view the message profile as a contract specifying the exact behavior of each participant in the conversation. The basic steps that an analyst will perform to design and implement an interface are summarized below. Message profiles can help with every step.

- Analyze the interface needs to determine requirements, including the use case, the static definition of the message, the dynamic interactions, and the vocabulary.
- Document the interface in a standard way, i.e., conformance profiles.
- Implement the interface.
- Devise a testing plan; including the generation of test messages, test cases, test suites; the test plan should account for syntactic, semantic, use case scenario testing, and the handling of error conditions.
- Execute the test plan.

A key development for promoting interoperability was the codification of a means to express message profiles in a standardized way. While natural language documentation of a message profile acceptably facilitates interoperability at the message implementation level, the standardization of the message profile documentation itself adds a new dimension to the promotion of interoperability.

The standardized conformance profile is an XML document specified in terms of both a normative DTD and schema. It is in effect a document that can be used to consistently understand messaging specifications in an automated fashion. Use of such a common format document enables interoperability among messaging tools of various makers, which in turn ensures effective communication of specifications. When used in conjunction with centralized profile registries, conformance profiles offer a reliable means of comparison and differencing for consumers. In addition the profile may be used as a common basis for exposing vendors' value added product features such as profile directed code generation, and profile directed message automation. Another important outcome of the development of the standard conformance profile is the ability to use the profile directly in message instance validation.

Because of the specificity of the conformance profile its use in validation is an important consideration. Further developments of the Conformance SIG such as the incorporation of code sets (known as "tables") within the profile will enhance

the value of the profile as the basis for assessing message instance validity. This and other features under consideration by the Conformance SIG such as the inclusion of regular expressions as a means of evaluating content, promise to make the profile an even more valuable vehicle for message validation in the future.

Capabilities such as these can also have significant impact in terms of how validation is accomplished. Code set validation and content validation has historically been the purview of the individual applications. The use of message profiles in validation to accomplish this can offload some of the burden to the messaging system, potentially reducing both the computational and development effort devoted to validation in endpoint applications. Reassigning such validation duties to the messaging system also ensures consistency of the outcome, therefore enhancing the realization of effective interoperability among applications sharing the same messages.

The standard allows for localizations (called "Z" elements) that give users the ability to extend the standard in a way that satisfies site-specific needs. Prior to message profiles, documentation of localizations was ad hoc. With the use of profiles, localizations can be clearly documented and therefore tested.

5 Tools for Supporting Conformance Profiles in Practice

To realize the benefits of conformance message profiles, tools are needed to support their use. We present our collection of freely available tools and describe how they can be used to improve interoperability among healthcare systems. The Messaging Workbench can be used to build and document profiles in a common format. Message Maker creates test message instances and the NIST HL7 test framework can be used to evaluate HL7 interface implementations.

5.1 Messaging Workbench

The Messaging Workbench (MWB) [2] is a multifaceted productivity tool for HL7 messaging professionals. It was conceived at the VA initially as a tool for messaging specification developers, which remains a fundamental focus of the tool. This capability has been enhanced over six plus years of development effort driven to a large extent by association with the Conformance SIG. Currently it supports the HL7 version 2 family of specifications. It incorporates all the version 2.x artifacts in the form of libraries that are readily available within the tool for use in specification composition, reporting, message instance decomposition, reverse engineering, validation and for test message generation.

The MWB is a Windows platform based GUI tool. MWB profile building is facilitated by the incorporation of HL7 2.x version specific artifact libraries. Building a message profile from scratch is accomplished quickly by selecting a particular message from the message library list and compiling the message structure against the version specific segment library. These steps result in the appearance of a hierarchical representation of all elements in the message in the form of a message tree. Selection of individual message elements in the tree permits each to be constrained and annotated as required by the particular

implementation. The addition of use case information and diagrams using the built in diagram editor typically follows the constraint work. Additionally the developer may constrain the profile vocabularies by selecting a subset of tables and table elements from the master table file library, which completes the constrained profile definition.

Another important part of the profile building process is the introduction of localized structures into the message. The HL7 standard promotes this aspect of messaging via the use of "Z" elements. The MWB supports the creation of and incorporation of localized structures for all of the version 2 artifacts: message, segments, fields, data type, and tables.

Once the profile is created it must be communicated to the interface partners. The MWB native profile format is a proprietary file structure that may be shared among other users of the MWB. Additionally though, the MWB provides a number of reports for alternate expression of the profile and message artifacts. The most important of these is the normative HL7 conformance profile, which is an XML document that validates against the HL7 DTD and schema. This format is especially important because it imparts interoperability among tools that are capable of importing this normative document.

While the MWB makes the initial profile construction a simple point and click operation, the effort required to adequately constrain and localize a profile is considerable. To ensure that these efforts may be conserved and therefore leveraged in future implementations, the MWB allows the developer to save the constrained artifacts in the form of libraries that may be reused and invoked in the same manner as the standard libraries.

The MWB may be used to receive and validate message instances. It will optionally reply with an HL7 ACK message describing the result (including errors) for each validated message. It also has the capacity to decompose received message instances making them available for individual analysis and for reverse engineering. The reverse engineering aspect is especially valuable for creating or updating the documentation of operating interfaces that may have little or no documentation. The MWB has the capability of generating semantically correct test messages. It also has a mechanism to ensure data consistency among related message elements for generated messages, which facilitates development and implementation of rational test sets.

Over the years, the MWB has grown together with the SIG in terms of developing the means to promote interoperability among interface participants. The MWB maintains its mutually productive association with the Conformance SIG. Together they continue to advance the cause of messaging interoperability and consequent implementation integrity. In this relationship, the MWB often serves to prototype SIG pursuits as it did with the normative conformance profile. Currently it supports the inclusion of tables or code sets into the profile as mentioned above, which will be officially introduced into the standard in HL7 version 2.6. It also supports regular expression assessment of content, a capability still under consideration. The overall MWB goal within the context of promoting interoperability among interface partners continues toward providing a light-weight, inexpensive means of interface development and testing at the desktop level.

5.2 Message Maker

An important aspect for achieving interoperability is determining if communicating applications have correctly implemented an interface based on a message profile. The existence of a well-defined and extensive set of test messages is paramount. Message Maker [3,4] developed at NIST is a GUI-based tool that primarily focuses on the generation of test message instances. It supports the full range of possible messages derivable by the profile definition.

Profiles are only fully defined at the implementation level and offer unmatched requirement definitions and flexibility. However, this presents a challenge for testing, as each site can potentially define their own set of unique profiles and therefore will require a set of test messages for those profiles. Message Maker produces self-adapting test messages based on a given message profile. The messages are automatically and dynamically created and factor in unique characteristics of the profile definition.

The constraints defined in the message profile provide the parameters that can be varied to construct test message instances. Message Maker can create messages that may be valid or invalid and contain variation from message to message. An example of an invalid message is a missing data item for a required field. A number of test options settings control the variation in the construction of a message. These may include segment and field cardinality, the usage of certain primitive fields, value sets, data content, and more. Message Maker can produce automated message sets or the user can craft specific messages. Message Maker provides a repository of data values and supports access to the repository that allows the user control over the content of element values. Messages are created with associated metadata that is useful during testing. Message Maker supports both XML [5] and ER7 message encodings and provides a utility to browse and edit messages. See the Message Maker User's Guide for a complete description on all the features and how to use the tool [3].

5.3 NIST HL7 Test Framework

To facilitate interoperability, the NIST HL7 test framework can be employed to evaluate implementations of the HL7 standard. The test system can be used without a profile to establish basic message exchange capabilities, but for a more complete assessment of an implementation's behavior, use of a profile is necessary.

The test framework can be used to conduct dynamic testing designed to evaluate implementation behavior in a real-time, operational environment. When message sending is initiated by the implementation being tested, the test system can analyze received messages, examine message content, validate messages, conditionally send follow up messages, and return acknowledgement messages. Alternatively, message sending can be initiated by the test system. In this case, the system looks for and examines acknowledgement messages returned from the implementation.

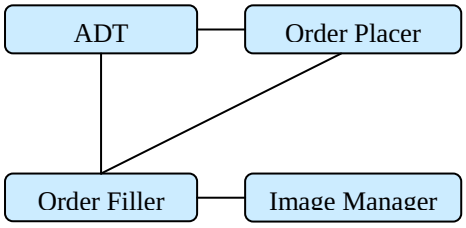

Fig. 2. Example HL7 Operational Environment

The user is given full control over messages sent by the test system and the capability of imposing conditions on received messages that must be satisfied. Nevertheless, to effectively utilize the test system as a tool for facilitating interoperability, a test plan should be employed that enables a methodical examination of the implementation to be tested. In particular, when profiles are employed as part of the testing plan, the likelihood of achieving interoperability is significantly increased. Employing profiles allows the test system to validate messages and determine if an implementation's behavior is consistent with the message constraints imposed by the profile. When a profile is employed, Message Maker can be used to produce a wide range of messages that can be utilized for testing. The likelihood of interoperability among implementations successfully tested utilizing an extensive range of messages and a comprehensive testing plan is significantly increased.

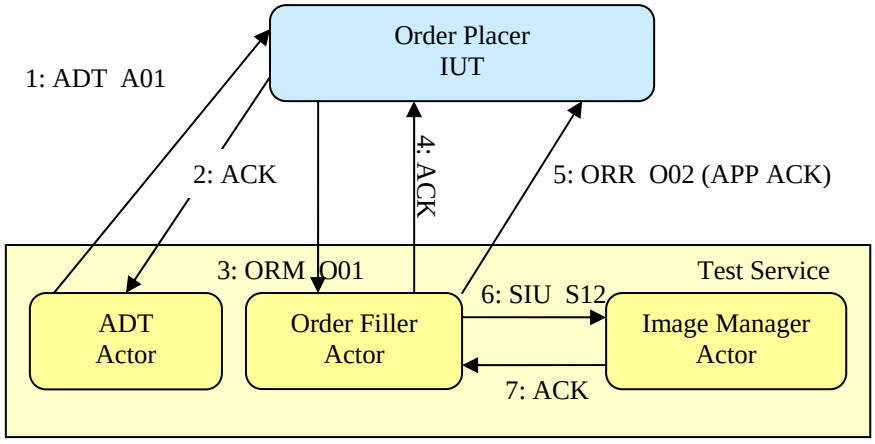

Fig. 3. Example HL7 Testing Environment

The NIST test system architecture is based on a set of actors that support a set of test services for evaluating implementations of HL7 messaging systems. Actors are independent threads of executing Java code that can be configured to act in place of any arbitrary HL7 application.

```
<!-- Example test suite. -->
<testSuite name="Example Test Suite" testReport="nameOfTestReport">

  <testCase name="Patient Registration">
   <actor namespaceID="ADT">
   <!-- send a message to the IUT-->
   <action type="Send" value="ADT_A01Message">
    <to namespaceID="orderPlacer"/>
   </action>
   <!—look for an Ack from the IUT. -->
   <assert value="receive">
    <from namespaceID="orderPlacer"/>
    <messageType value ="ACK"/>
   </assert>
  </actor>
 </testCase>

 <!-- run this test case only if the Patient Registration test case completed successfully -->
 <testCase name="Order Filled" assert="Patient Registration">
  <actor namespaceID="orderFillerName">
   <!-- wait to receive a message from the IUT -->
   <assert value="receive" id=”assertion1”>
    <from namespaceID="orderPlacer"/>
    <messageType value="ORM_O01"/>
   </assert>
   <!-- if the assertion passed, send app acknowledgement message to IUT -->
   <action type="Send" value="ORR_O02Message" assert=”assertion1”>
    <to namespaceID="orderPlacer"/>
   </action>
   <!-- also send a message to the Image Manager -->
   <action type="Send" value="SIU_S12Message" assert=”assertion1”>
    <to namespaceID="imageManagerName"/>
   </action>
   <!-- make sure we get back an Ack -->
   <assert value="receive">
    <from namespaceID="imageManagerName"/>
    <messageType value ="ACK"/>
   </assert>
  </actor>
 </testCase>
</testSuite>
```

Fig. 4. Example Test Script

Before any interactions between test system actors and external applications can begin, the test system must be configured. Test system configuration is a relatively straightforward task designed to construct a test environment that mimics the operational environment of the HL7 system to be evaluated. For example, the user may plan to deploy an Order Placer as one component of an operational

system that also includes an ADT, an Order Filler, and Image Manager applications. The example operational system is depicted in Figure 2 [6]. To evaluate the Order Placer (i.e., the implementation under test (IUT)), actors are configured to replace each of the remaining applications. The resulting environment, depicted in Figure 3, should be indistinguishable to the Order Placer from the operational environment.

Once configuration has been completed, the user can initiate an exchange of messages. The user might, for example, direct the ADT actor to send a message to the IUT. The ADT actor can notify the user when an acknowledgement message is received. At the same time, the user can direct the Order Filler to listen for incoming messages from the IUT and conditionally send subsequent messages to both the IUT and the Image Manager. Generic actors support a general acknowledgment scheme defined in the standard, but do not support the application specific acknowledgement scheme that the standard also defines. To effect application acknowledgement mode, the user has to direct an actor to send the appropriate acknowledgement message. If general acknowledgment mode is also being used, then a test script similar to the one shown in Figure 4 could be used to effect the appropriate actions; ORR_O02 is the application acknowledgement message that should be returned when the ORM_O01 message is received.

The sequence of events resulting from the above test suite is depicted in Figure 3. The test system logs each action and all messages sent and received by each actor. A test report documenting actions and results is also generated. If all actions complete and all assertions are satisfied, the test case outcome will be successful.

6 Conclusion

The ability to share relevant information among diverse healthcare systems and provide consistent data across applications will help improve the quality of care. It will also improve patient safety and reduce the cost of healthcare. HL7 defines the interfaces that allow centrally located and distributed information systems to communicate. The standard establishes rules for building interfaces and provides many optional features to accommodate the disparate needs of the healthcare industry. However, for interfaces to be reliably implemented, a precise and unambiguous specification must be defined. HL7 introduced the concept of message profiles that state precisely the structure and constraints of a message. The use of profiles promote interoperability by providing trading partners a common format for documenting interface specifications.The MWB tool can be used to build profiles to assure that correct message structures and constraints are properly documented. An important end product is an XML representation of the message profile in a standardized form.

To ensure interoperability among healthcare systems, installations must be implemented correctly—conformance testing is essential. Employing a comprehensive testing program at the onset of an implementation leads to more reliable systems, and ultimately, reduced costs. Message profiles provide the mechanisms that promote better testing of implementations. However, message profiles don’t eliminate the complexity of the standard. At any given HL7

installation, many interfaces (specified by message profiles) will be defined and need to be tested. For a given profile, an extremely large number of unique test messages can be constructed [7]—a functional subset must be obtained. Hand-crafting this set of test messages becomes a daunting task; in many cases is cost prohibitive. Automatic and dynamic testing tools are essential. Message Maker is a productivity tool for constructing test messages for any given message profile. The test messages are an integral part of a conformance testing system. The NIST HL7 testing framework will provide services for test setup, sending/receiving messages to/from the implementation under test, executing the tests, evaluation of the response, and other testing activities. The test framework can be used to assess the overall conformance of implementations. Employing an implementation and testing strategy based on message profiles and the tools to support them will improve interoperability among healthcare systems.

7 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] 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.

[4] "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.

[5] HL7 Version 2: XML Encoding Syntax, Release 1. ANSI/HL7 V2 XML-2003 ; June 4, 2003.

[6] Integrating the Healthcare Enterprise (IHE). IHE Technical Framework Volume II Transactions (Scheduled Workflow); http://www.ihe.net/tf/.

[7] "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.