An HL7 v2 Platform for Standards Development and Testing

S. Martinez[1] **and R. Snelick**[1]
[1]National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA

Abstract – *Development of healthcare data exchange standards has long been problematic, plagued with ambiguous and inconsistent requirement specifications. This situation leads to potential misinterpretation by implementers, thus limiting the effectiveness of the standard and creating artificial and unnecessary barriers to interoperability. Likewise, the ability to test implementations effectively for conformance to the standards is hindered. The current approach of standards development and test plan creation relies on word processing tools, meaning implementers must read and interpret the information in these documents and then translate it into machine-processable requirements and test assertions. This approach is error prone—a better methodology is needed. We present a set of productivity tools in an integrated platform that allow users to define standards and test plans that result in machine-processable artifacts. A testing infrastructure and framework subsequently uses these artifacts to create conformance testing tools automatically. We present and demonstrate the utility of a platform for developing standards, writing test plans, and creating testing tools. This end-to-end methodology is illustrated by describing a case study for the HL7 v2.6 Vital Records Death Reporting Implementation Guide.*

Keywords: Healthcare Data Exchange Standards; Healthcare Information Systems; Interoperability; Standards Development; Testing.

1 Introduction

For 30 years, HL7 Version 2 (v2) has been the predominant standard used for the exchange of healthcare administrative and clinical data. Healthcare information systems use the HL7 v2 protocol to develop standardized interfaces to connect to and exchange data with other systems. HL7 covers a broad spectrum of domains including Patient Administration, Laboratory Orders and Results, and Public Health Reporting. The base HL7 v2 standard [1] is a framework that contains many message events, and for each event it provides an initial template (starting point) that is intended to be constrained for a specific use case. The application of constraints to a message event is referred to as profiling [2,3]. For example, the ADT (Admit, Discharge, Transfer) A04 (Register a Patient) message event is a generic template for communicating information about a patient. The base message template is composed of mostly optional data elements. For a given use case, e.g., Vital Records Death Reporting (VRDR) [4], the message template is "profiled". That is, elements can be constrained to be required, content can be bound to a set of pre-coordinated codes, and so on. The base message event (e.g., ADT A04) that has been constrained for a particular use (e.g., VRDR) is referred to as a conformance profile. An implementation guide is a collection of conformance profiles organized for a particular workflow (e.g., report, revise, or cancel a death report). In this example, three conformance profiles exist each with different message events, one for report, revise, and cancel. To date, HL7 v2 implementation guides have been created using word processing programs, which has resulted in ambiguous and inconsistent specification of requirements. This practice has hindered consistent interpretation among implementers, which has created an unnecessary barrier to interoperability.

We present an end-to-end methodology and platform for developing standards (implementation guides), writing test plans, and creating testing tools in the HL7 v2 technology space [3,5]. The platform includes three key foundational components:

- A tool to create implementation guides and conformance profiles
- A tool to create test plans, test cases, and associated test data
- A testing infrastructure and test framework to build testing tools

A key to the approach is that the "normal" process of creating implementation guides, test plans, and testing tools is "reversed". Instead of creating requirements using a natural language and subsequently interpreting the requirements to create test plans and test assertions, the requirements are captured with tools that internalize the requirements as computable artifacts.

Figure 1 illustrates the methodology. Domain experts develop use cases, determine the message events that correspond to the interactions in the use cases, and then proceed to define the requirements. Using the NIST

Martinez, Sandra; Snelick, Robert.
"An HL7 v2 Platform for Standards Development and Testing."
Paper presented at HIMS'17 - The 3rd International Conference on Health Informatics and Medical Systems,
Las Vegas, NV. July 17, 2017 - July 20, 2017.

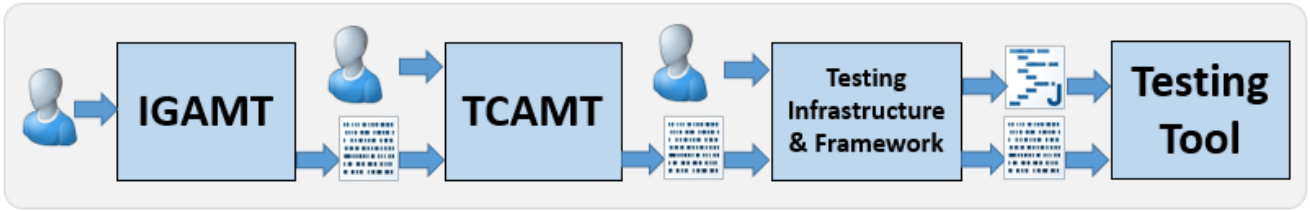

Fig. 1. NIST HL7 v2 Standards Development and Testing Platform

methodology, they accomplish these tasks by entering this information into the Implementation Guide Authoring and Management Tool (IGAMT). During this process, the domain experts constrain the message events according to the requirements needed by the use case. Section 2 will elaborate more on this process and on the details of how the requirements are constrained. The output of IGAMT is a set of artifacts that are represented in Word, HTML, and XML formats. The complete implementation guide, including the narrative and messaging requirements, can be exported in Word or HTML. Such formats are suitable for ballot at standards development organizations such as HL7 or IHE (Integrating the Healthcare Enterprises). Each conformance profile can be exported as XML. The XML format contains all of the messaging requirements in a machine processable representation, which is the most important aspect of IGAMT since the XML conformance profiles have many uses including message validation, test case and message generation, and source code generation.

The XML conformance profile and/or the internal IGAMT model are imported by the Test Case Authoring and Management Tool (TCAMT). TCAMT is used to create targeted test cases for interactions (profiles) defined in the implementation guide. The output is an additional set of constraints in an XML format. The entirety of the output generated from IGAMT and TCAMT is called a "resource bundle".

The NIST platform includes a testing infrastructure of common utilities used for testing, such as a message validation engine, along with a testing framework that provides various testing tool components, such as a communication framework and a profile viewer. Testing Tool instances are then created using the testing infrastructure and framework components as well as the resource bundle output generated from IGAMT and TCAMT.

The NIST platform in essence allows end users to create conformance testing tools by means of a set of productivity tools. This streamlined approach can greatly reduce today's problems with conformance test tools for standards. These problems include: there are too few of these tools, they are expensive to build, they are not dynamic for local refinements, and their time to market is protracted. Additionally, the platform provides value through enforcing consistent and rigorous rules for requirements specifications.

The remainder of this paper explains the NIST platform in more detail in the context of a real-world case study. The layout of the VRDR implementation guide is presented, and we describe how IGAMT is used to capture the messaging requirements. Next, an explanation of how a set of targeted test cases are created in TCAMT is provided. Finally, the resulting VRDR test tool is presented. The goal is to inform the reader about the ease with which HL7 v2 implementation guides, test cases, and testing tools can be created using the NIST platform compared to the current laborious methods used today.

2 IGAMT

IGAMT is a tool used to create HL7 v2.x implementation guides that contain one or more conformance profiles. The tool provides capabilities to create both narrative text (akin to a word processing program) and messaging requirements in a structured environment. Our focus in this paper is on the messaging requirements.

IGAMT contains a model of all the message events for every version of the HL7 v2 standard. Users begin by selecting the version of the HL7 v2 standard and the message events they want to include and refine in their implementation guide. For example, the message events ADT^A04, ADT^A08, and ADT^A11 are used to create 14 conformance profiles in the VRDR implementation guide. Each message event is profiled (constrained) to satisfy the requirements of the use case.

Rules for building an abstract message definition are specified in the HL7 message framework, which is hierarchical in nature and consists of building blocks generically called elements [1]. These elements are segment groups, segments, fields, components, and sub-components. The requirements for a message are defined by the message definition and the constraints placed on each data element. The constraint mechanisms are defined by the HL7 conformance constructs, which include usage, cardinality, value set, length, and data

type. Additionally, explicit conformance statements are used to specify other requirements that can't be addressed by the conformance constructs. The process of placing additional constraints on a message definition is called profiling. The resulting constrained message definition is called a conformance profile (also referred to as a message profile). An example of a constraint is changing ***optional*** usage for a data element in the original base standard message definition to ***required*** usage in the conformance profile.

IGAMT provides, in a table format user interface, the mechanisms to constrain each data element at each level in the structure definition. The rows of the table list the data elements according to the structure being constrained (segments, fields, and data types). The columns list the conformance constructs that can be constrained for a data element, including the binding to a value set.

One key philosophy of IGAMT is the capability of creating reusable building blocks. These lower level building blocks can be used to efficiently create higher level constructs. The building blocks include data type flavors, segment flavors, and profile components. A base data type can be constrained for a particular use; the resulting data type is called a data flavor (or data type specialization). A given base data type may have multiple data type flavors. These flavors can be saved in libraries and reused as needed. A similar process applies to creating segment flavors.

A profile component represents a subset of requirements that can be combined with other profiling building blocks. One such example is the definition of a profile for submitting immunizations. The CDC creates a national level profile. However, individual states may have additional local requirements that can be documented in a profile component. Only the delta between the national and local requirements is documented in the profile component. Combining the national level profile and the state profile component yields a *complete* (implementable) profile definition for that particular state. This design provides a powerful and effective approach to leveraging an existing profile.

A utility for creating and managing value sets is also provided. Specific value sets can be created and bound to data elements. Value set libraries can also be developed for reuse.

3 VRDR Use Cases

The Vital Records Death Reporting (VRDR) Implementation Guide (IG) [4] was developed to support the transmission of death-related information from the health care provider's electronic health records (EHR) to the jurisdictional vital records offices (JVRO) and to the National Statistical Agency (NSA) [6]. Five use cases/workflows are identified to describe the transmission of data: Provider Supplied Death Information (PSDI), Jurisdiction Death Information (JDI), Void Certificate Reporting (RVCA), Coded Cause of Death (CCODA), and Coded Race/Ethnicity (CREIA) [4]. The use cases require three message events: ADT^A04, ADT^A08 and ADT^A11. A given use case has more than one interaction; in total, 14 interactions are needed.

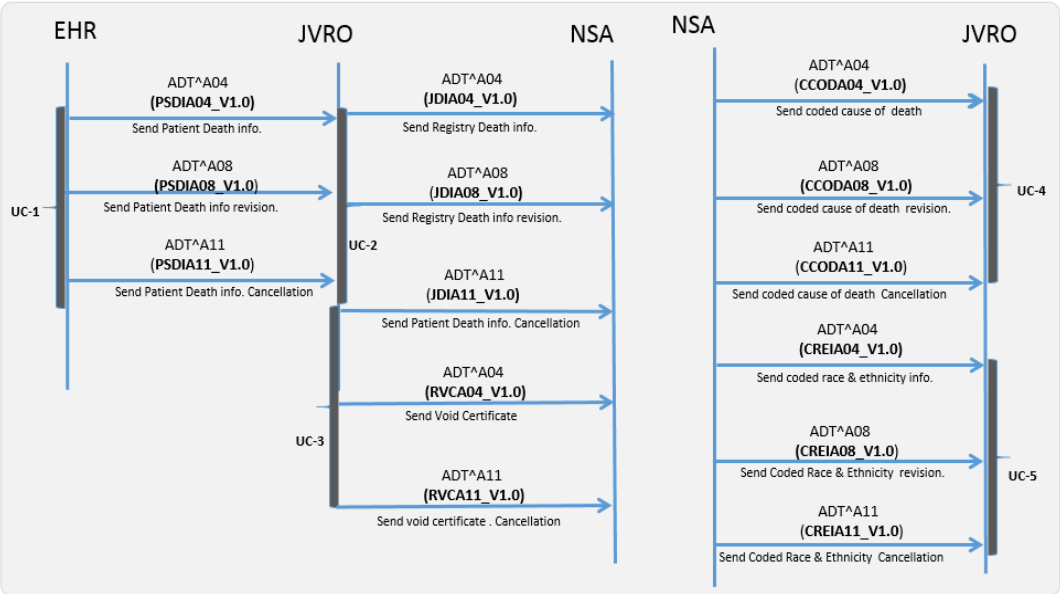

Fig. 2. Vital Records Death Reporting Interactions

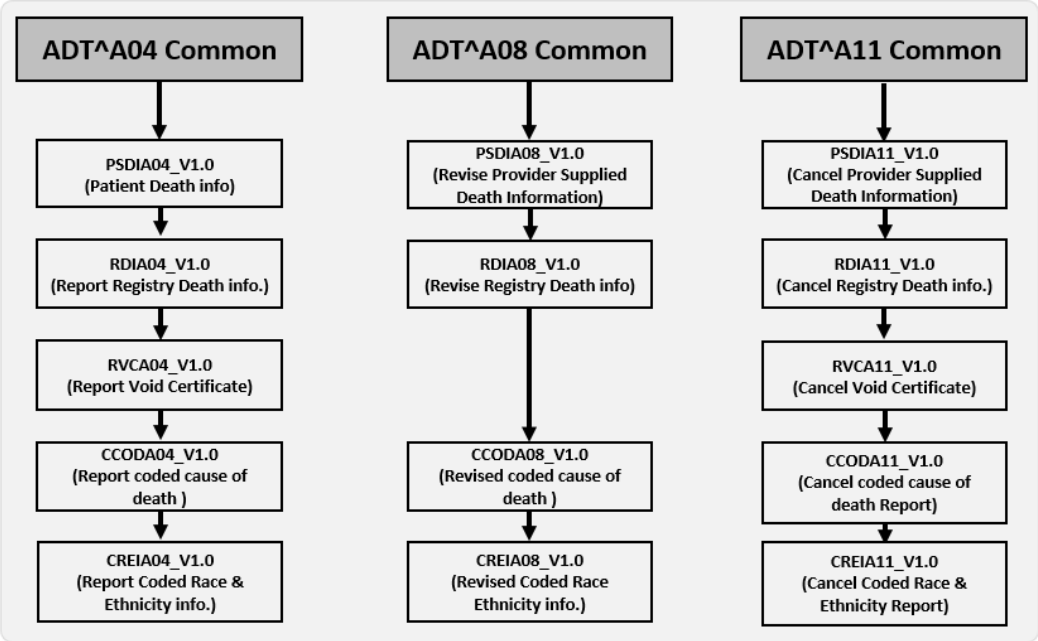

Fig. 3. VRDR Profiles and Design

Figure 2 shows the 14 interactions supporting the VRDR use cases. The type of information being exchanged determines how each interaction (message) is constrained. For the PSDI use case, the ADT^A04 is constrained for reporting about a person's death, the ADT^A08 message is constrained for updates to the report, and the ADT^A11 message is constrained to cancel the report.

Each interaction is assigned a unique profile identifier; e.g., "PSDIA04_V1.0" is a profile identifier for the "Send Patient Death Information" interaction (ADT^A04 interaction in Figure 2). The same three message events (ADT^A04) are employed across various use cases, however, the context in which they are used is different; therefore, the set of constraints applied are different, each resulting in a unique conformance profile (for the same base ADT^A04 event). The content is defined by the set of initiating and responding systems.

Figure 3 shows the conformance profile-building approach for the VRDR profiles and the group of profiles that share the same trigger event. The A04 message event is loaded and constrained to create the common A04 profile. From there, the PSDIA04_V1.0 profile is created. The PSDIA04_V1.0 profile is used in building all of the profiles associated with the A04 event, and the corresponding message-level constraints are added to each profile. Constraints at the message level include segment and group usage, cardinality, and any additional requirement in the form of conformance statements.

The approach used in the development of the A04 profiles is followed in the creation of the A08 and A11 profiles. The base message events are loaded and constrained to develop the PSDIA11_V1.0 and PSDIA08_V1.0 profiles. These profiles are then leveraged to create the remaining profiles sharing the same message event using the IGAMT cloning capability.

Profiling at the value set, segment, field, and data type (component) levels is followed, and it can be achieved in any order, thus taking advantage of the IGAMT capability that allows for the creation of reusable building blocks. The value set library can be created using the IGAMT built-in mechanism for loading value sets from HL7 tables and the CDC PHINVADS (Public Health Information Network Vocabulary Access and Distribution System) value sets [7].

VRDR data type flavors are built from the HL7 v2.6 data type library. They are defined using the constraints specified in the IG, such as length, usage, and value set binding and any constrains in the form of conformance statements. Occasionally, depending on how the data type is going to be used, the IG defines more than one data type specialization for a base data type. In the case of the "HD (Hierarchic Designator Assigning Authority)" data type, an additional flavor called HD_AA is defined. This flavor is used when an OID (Object Identifier) is assigned to designate an assigning authority.

The VRDR segment flavor are created from the HL7 v2.6 segment library using the length, data type, cardinality, usage, value set binding, and conformance statement constraints defined in the IG. In some instances, it is necessary to create additional segment flavors to indicate constraint deltas among the profiles. For example, the constraints in the PV1 segment are applied to every profile, therefore only one PV1 flavor is created. In the case of the PID segment, the constraints are different in each use case; therefore, a segment flavor is created for each use case, and that flavor can be reused in the respective profiles.

As shown, a key feature in IGAMT is the capability to create precise object definitions and to use (and reuse) the objects as building blocks to create higher level objects, such as segment and conformance profiles.

4 TCAMT

TCAMT is a tool used to create HL7 v2.x test plans that contain one or more (typically many) test cases. A test case can consist of one or more test steps. A test step can be an HL7 v2.x interaction or a manual step such as visually inspecting the contents of the system under test's (SUT's) display screen. Each test case and test step can consist of a test description, pre- and post-conditions, objectives, evaluation criteria, and additional notes and comments. Test steps for an HL7 v2.x interaction contain an HL7 v2 message (that is, specific data) that is in alignment to an XML conformance profile created from IGAMT.

TCAMT allows domain experts to create test cases that target certain scenarios and capabilities. Using these test cases provides context, which expands the scope of testing beyond just the constraints in the conformance profile. Without context, a validation tool cannot test a message exhaustively to all requirements specified in the implementation guide. For example, elements with "required, but may be empty (RE)" usage or elements with "conditional usage (C)" cannot be assessed without targeted tests. A message that is validated against the requirements of a conformance profile without any provided context is called "context-free testing". A message that is validated against the requirements of a conformance profile and with a provided context is called "context-based testing" [3]. The test cases provide context, and TCAMT is a tool that allows users to create the test cases.

A key feature in TCAMT is its use of the conformance profiles created in IGAMT as a foundation. The message definition and requirements are available to the TCAMT user based on information that was entered into IGAMT. Then, the TCAMT user provides the data associated with each message element of interest. TCAMT also allows the user to enter additional assertion indicators based on what they want to test. For example, for an element with a usage of "RE", the user might provide data that are expected to be entered into the sending system for the element, and the user also might select an assertion indicator. There are several assertion indicators that could be selected, for example, "presence". In this case, if the user provides test data and the indicator of "presence", an additional assertion (constraint) is generated by TCAMT and is provided to the validation. For elements with "RE" usage, the element must be supported by the SUT, but in a given message instance the element may not be populated. For this construct, the tester wants to ensure that the implementation has, in fact, included support for the element.

In a context-free environment, the absence of data in a message is not a conformance violation for elements with "RE" usage. However, in the example test case described above, data were provided, and an assertion for the presence of the data was selected. Now, when a message created for this test case is validated, the additional assertion triggers the check for the presence of data for this element. This method is one way to determine support for the element.

Via TCAMT, the user can create an unlimited number of test cases and test a broad spectrum of requirements. Other assertion indicators can be used to test for specific content or for the non-presence of an element. Additionally, test data can be provided to trigger conditional elements. In other instances, support for certain observations may need to be ascertained. In such cases, test data for specific observations (e.g., cause of death, date/time pronounced dead, etc.) are provided, requiring the message instance to contain an OBX segment for that observation. TCAMT provides the mechanisms to conveniently and consistently create test cases. Output from TCAMT provides the additional constraints that are interpreted by the validation engine.

5 VRDR Test Plan

The VRDR test plan consists of a set of scenarios and test cases that emulate real-world events and workflow. The scenarios are designed to target specific requirements that are not easily testable in a context-free environment. The goal of the VRDR test plan is to provide a set of test cases that collectively tests the spectrum of requirements defined in the VRDR implementation guide. Therefore, for each interaction in support of a use case, test scenarios and test cases are needed.

Figure 4 shows an excerpt of the test plan. A scenario for the Provider Supplied Death Information (PSDI) is indicated that contains three test cases and associated test

steps. Test steps have a 1-to-1 relationship to an HL7 v2 message (interaction), and each message is bound to the requirements in its corresponding conformance profile.

Fig. 4. VRDR Test Plan Excerpt

For each test case, a real-world story is given along with specific test data that coincide with the test story. The test data provide a known data set that can be used to create additional assertions (beyond those provided in the conformance profile). This approach is the principle behind context-based testing. Each test step interaction contains a message that is associated with its corresponding conformance profile. TCAMT provides a productivity mechanism to create the test messages using the underlying structure provided in the conformance profile. Once the test message (and therefore test data) has been created, additional assertions can be specified that align with the testing goals.

Table 1 shows two important examples of how this process works. TCAMT facilitates specific assertions by allowing the test plan designer to assign assessment indicators to the data elements. The combination of the provided test data and assessment indicator generates the assertions. The PDA-2.9 example shows a case where the Death Location Description element is constrained with "RE" usage, which indicates that the element must be supported but data may not always be available. To test support for this element, the test case provides test data ("Mercy Hospital") and the assessment indicator is set to "Presence". These settings will generate an assertion that makes the presence of the PDA-2.9 element required. The OBX-3.1 example shows a case where an observation (an OBX segment) is expected in which the observation is "Cause of Death". Here, the value "69453-9" is provided, which is a LOINC (Logical Observation Identifiers Names and Codes) code that indicates a "Cause of Death". By explicitly requiring the "Cause of Death" observation be included in the message, the testing is ensuring that the SUT can support this observation. In this example, only one element in this segment is shown, but typical testing scenarios will have a coordinated set of assertions for the set of elements in the OBX segment. For example, OBX-3.3 would assert that the content of this element is "LN" to confirm that the code "69453-9" is in fact drawn from the LOINC code system.

Creating test cases that target specific capabilities, such as "sending the Cause of Death observation" is an important aspect of testing and a key incentive for conducting context-based testing. Without this level of specificity, assessment of systems is limited. Only a few examples have been provided here to give the reader a sense of the sorts of items that can be tested. However, this approach expands the test space significantly. Other aspects that can be tested include cardinality, length, value set constraints, conditional elements, specific content, workflow, and functional requirements.

Element	Name	Usage	Test Data	Assessment Indicator	Rationale	Conformity Assessment
PDA-2.9	Death Location Description	RE	Mercy Hospital	Presence	Assess that the SUT can support this element.	PDA-2.9 SHALL be present.
OBX-3.1	Observation Identifier	RE	69453-9	Value-Test Case Fixed	Assess support for providing the "Cause of Death" observation	An OBX segment SHALL be present in which OBX-3.1 SHALL be "69453-9".

Table 1. Context-based Assertion Examples

6 Infrastructure and Framework

NIST has built an HL7 v2.x testing infrastructure and framework to aid in the process of creating conformance testing tools. The testing infrastructure provides a set of services utilized by the test tool framework to build specific instances of tools. A test tool can be specific for a particular domain, or it can be general-purpose. The general-purpose tool is a NIST-hosted web application where a user can upload conformance profiles and test plans to create a test tool. The conformance test tool essentially is generated as a by-product "for free" once the validation artifacts have been created. This liberates the domain experts from the tool building process. Alternatively, the framework can be leveraged, customized, and installed locally. Using the framework, developers can choose to create domain specific or general-purpose web application conformance test tools, access the validation via web services, or incorporate validation via JAR (Java Archive) files or source code. Regardless of the use, the NIST platform can significantly improve the quality of implementation guides, assist in the creation and maintenance of test plans, expedite the stand-up of a validation tool, and, overall, reduce the cost and time of the entire process.

7 VRDR Test Tool

A VRDR conformance testing tool is built using the testing infrastructure and framework, the IGAMT-produced conformance profiles, and the TCAMT-produced test plan. The test tool is a web-based application (see [8] to access) that supports both context-free and context-based validation. In addition to performing message validation, the tool provides a browse-able view of the requirements for each conformance profile. In the context-based mode, the test story, test data, and an example message are provided for each test step.

In the context-free mode, the user simply selects the conformance profile to validate against and imports the message. The validation is performed automatically and a report is given. In the context-based mode, the user selects the test step and imports the message to validate. The test tool sets the validation to the conformance profile linked to the test step, performs the validation, and provides a report. In both modes, a tree structure of the message is shown on the left panel of the validation screen and can be used to inspect the content of individual data elements.

8 Summary

We presented an end-to-end methodology and platform for developing standards, writing test plans, and creating testing tools in the HL7 v2 technology space. The platform includes three key foundational components: (1) a tool to create implementation guides and conformance profiles; (2) a tool to create test plans, test cases, and associated test data; and (3) a testing infrastructure and test framework to build testing tools. We demonstrated the approach by creating a test tool for the HL7 v2.6 Vital Records Death Reporting use case. Requirements were captured in IGAMT and exported as conformance profiles. TCAMT was used to create a set of test cases based on the conformance profiles. A conformance test tool was created by combining the validation artifacts with the testing infrastructure and framework.

9 References

[1] Health Level 7 (HL7) Standard Version 2.6, ANSI/HL7, October 2007, http://www.hl7.org.

[2] *Principles for Profiling Healthcare Data Communication Standards*. R. Snelick, F. Oemig. 2013 Software Engineering Research and Practice (SERP13), WORLDCOMP'13 July 22-25, 2013, Las Vegas, NV.

[3] *Healthcare Interoperability Standards Compliance Handbook*. F. Oemig, R. Snelick. Springer International Publishing Switzerland, ISBN 978-3-319-44837-4, December 2016.

[4] *HL7 Version 2.6 Implementation Guide: Vital Records Death Reporting, Release 1*. Draft Standard for Trail Use. August 2016. http://www.hl7.org.

[5] *NIST Resources and Tools in Support of HL7 v2 Standards*. http://hl7v2tools.nist.gov/

[6] *CDC National Vital Statistics System: http://www.cdc.gov/nchs/nvss/evital_standards_intiatives.htm*

[7] CDC Public Health Information Network Vocabulary Access and Distribution System (PHIN VADS); https://phinvads.cdc.gov/.

[8] *NIST Vital Records Death Reporting (VRDR) Conformance Testing Tool*; http://hl7v2-vr-r2-testing.nist.gov