Dynamically Generating Conformance Tests for Messaging Systems

Robert Snelick
Len Gebase
Sydney Henrard
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg MD, USA

Abstract *- The advent of XML technologies for data exchange negotiations in B2B applications is proliferating a new class of messaging standards that are only fully designed at the implementation level. These specifications offer unmatched flexibility. However, this presents a challenge for ensuring correctness, as each implementation is potentially unique--conformance tests are needed. We propose a generalized methodology and a tool that produces self-adapting test messages. The messages are dynamically created after design-time and factor in unique characteristics of an implementation. This contrasts to traditional methodologies, where tests are developed irrespective of any specific implementation.*

We demonstrate the utility of our methodology by implementing a tool, called Message Maker, which dynamically produces conformance test messages. Our target test domain is the HL7 version 2 healthcare messaging standard. Message Maker is a graphical-based tool that generates test messages based on any given HL7 XML message template, referred to as a message profile. The resulting messages can be used to test HL7 applications to ensure that they adhere to the message profile specification. Employing a comprehensive testing program at the onset of an implementation leads to more reliable systems, and ultimately, reduced costs.

Keywords: Automated Methods; Conformance Testing; Healthcare Information Systems; Messaging Systems; Novel Software Tool.

1 Introduction

Today's industries, including healthcare, manufacturing, and e-commerce, rely heavily on the seamless exchange of information to drive their business. These enterprises define standards for the exchange, manipulation, and integration of industry-based information. The underlying infrastructure is based on standards and realized through XML (extensible markup language) technologies [7]. This offers highly sophisticated and dynamic systems. A growing number of such standards provide a framework for negotiation that allows many optional features. To achieve interoperability, implementations must constrain this set of optional features in a consistent manner. This provides the means to create well-defined special purpose specifications. However, this paradigm shift makes testing a difficult proposition since each specification would need its own set of tests. Software is notorious for not being adequately tested, for being rushed to market full of flaws. A primary reason for inadequate testing is the time and expense of developing comprehensive test suites. Correct implementation of software is critical--conformance tests are needed to promote interoperability among implementations. Old methods of systematically hand crafting tests over the course of several years must give way to newer methods of generating self-adapting tests. We have developed a methodology and a tool that produces self-adapting tests that are dynamically created and factor in unique characteristics for defined subsets of a given specification. The technique relies on the message profile represented in XML; it provides the layout of the message and governs the makeup of each element in the message. A processing engine follows a predetermined set of rules to combine the message profile, data, and test options settings dynamically. This creates a set of test messages. The test messages are unique and designed to assess different aspects of an implementation. The methodology has been applied to the Health Level 7 version 2 (hereafter HL7) messaging standard [1,2] and is realized in Message Maker—a tool for dynamically creating test messages [5].

In the section that follows we give an overview of the relevant concepts of the HL7 messaging standard and describe the inherent problems the standard has with regard to conformance. We present the concept of message profiles as a remedy and illustrate how it can be used to build test messages. Next we present a general methodology for dynamically developing test messages. A detailed description of Message Maker follows. Finally, we offer some discussion on conformance testing and development of an adequate set of test messages.

2 Case Study: HL7

A major challenge for the healthcare industry is achieving interoperability among information systems. HL7 is a widely-used standard for moving clinical and administrative information between healthcare applications. Systems that support the HL7 standard allow clinical data to be exchanged with other HL7 systems. This 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.

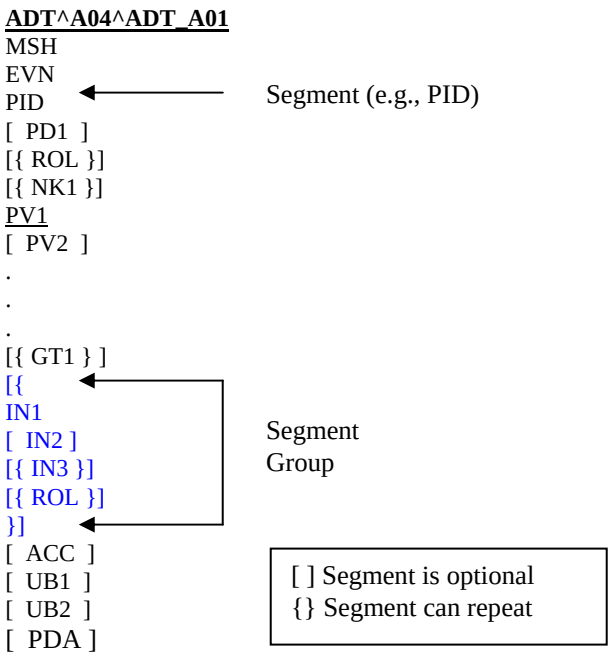

Figure 1. Abstract Message Definition

An HL7 *message* is an atomic unit of data transferred between systems [2]. Typical HL7 messages include admitting a patient or requesting a lab order for a blood test. HL7 describes an *abstract message definition* (Figure 1) 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 (Figure 2) is hierarchical in nature and consists of building blocks generically called *elements*. These elements are *segments*, *segment groups*, *fields*, *components*, and *sub-components*. Each element has associated attributes that 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 structure.

2.1 HL7 not the Solution Anticipated

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 for which there was no single, consistent interpretation. As a consequence, systems were difficult to implement and debug; a further result was undue cost. Implementers have described HL7 interfacing as total chaos. Fortunately, HL7 recognized this as a limiting factor for effectively managing healthcare communication. To help alleviate shortcomings, the concept of *message profiles* was introduced. 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.

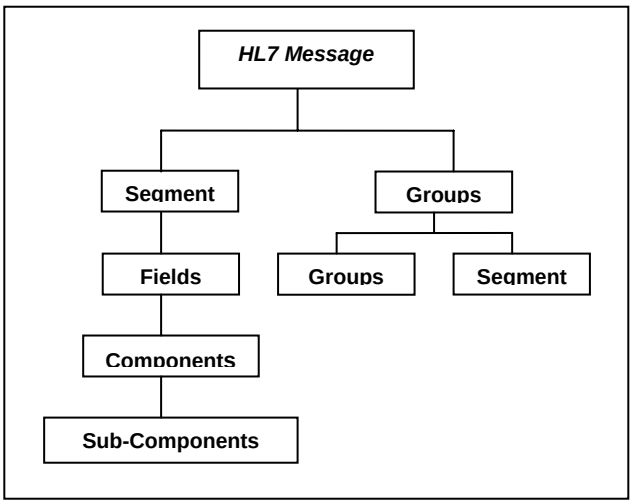

Figure 2. HL7 Message Framework.

Tools, such as the Messaging Workbench [4], have been developed to help in the construction of message profiles. Message profiles in one form are XML documents constrained by an XML schema. The schema, defined by HL7, describes explicitly the message layout and constraint attributes and their values allowed in the message profile. Profile builder tools enforce the rules of message profiles and have utilities that export profiles in XML. The message profile provides the template that enables the construction of HL7 messages. Figure 3 shows a snippet of a message profile represented in XML. Each element in the message profile is listed along with its associated attributes. Details on the message profile structure and constraints follow. The constraint attributes are important since they provide the opportunity for varying the test message instances.

Message Structure: The overriding rules for constructing a message are described by the message framework (Figure 2) [2]. In addition, for each message event, for example

"Admitting a Patient (ADT A04)", a specific abstract message definition is given (See Figure 1) that further defines the message. Messages that are created of a certain type must follow the template given in the abstract message definition and the rules given by the message framework. We refer to this as the *message structure*; it defines explicitly the elements and the order the elements must appear in a message instance. For example, in Figure 3, the "PID" segment contains the field "Set ID – PID", and so on.

Usage: Usage refers to the circumstances in which an element appears in a message [3]. Some elements must always be present, others may never be present, and others may only be present in certain circumstances. The set of usage rules that is considered are Required (R), Required but may be empty (RE), and Not Supported (X). 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.

```
…
<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="250" ItemNo="00123">
      <Reference>3.4.2.20</Reference>
      <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>
…
```

Figure 3. Snippet from a Message Profile.

Cardinality: Cardinality refers to the minimum and maximum number of repetitions an element may have [3]. The implication of cardinality has the same connotations at the segment, segment group and field levels. Cardinality does not apply to components and sub-components. Examples of an element cardinality include [0..1]; the element is optional, but can only have one occurrence and [1..*]; the element is required and may repeat an unlimited number of times.

Value Sets: A table of allowable values can be defined and associated with a certain element. For example, see the "*Issuing State, province, country*" component in the profile snippet (Figure 3); this element must be populated with a data value that is defined in table 0333. HL7 has mechanisms to define such tables; some tables are predefined by HL7 while others can be defined locally.

Length: The length attribute defines the maximum allowable length a value can have for a particular element.

Data Type: 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 [2]. 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.

2.2 Message Profiles Provide a Path to Conformance Testing

Message profiles provide a path to conformance and interoperability testing. Prior to the introduction of message profiles, interface specifications were not well defined. It was not possible to predict the number of message instances that could be derived from an interface. It was also difficult to ascertain what constituted a valid and invalid message instance. Message profiles solve this problem by providing the clarity that is essential to conduct testing.

2.3 Test Message Types

The constraints defined in the message profile provide the parameters that can be varied to construct test message instances. For each of these constraints, we describe some test type examples that can be constructed. The examples given below are not comprehensive, but provide a flavor of the test messages that can be generated. Test messages can be valid or invalid. Message variation with regards to *Usage* and *Cardinality* is relative to a *base message* in which the minimum values for these constraints are set.

Message Structure: The structure of a message can be manipulated to create test messages. For example, an extra segment can be inserted randomly into the message structure definition. A message generated using this message structure will be erroneous with regard to the message profile.

Usage: Various message instances can be generated given the usage constraint. For example, a test message can be

constructed such that an element with an R (required) usage is not populated with data. The implanted error results in an invalid message instance.

Length: Test messages can be constructed such that the data values will have lengths that exceed the limit defined in the message profile for an element.

Data Type: Test messages for elements with a primitive data type can be constructed such that invalid data values are populated for that element with regard to the data type. For complex data types, additional components can be inserted to render the element invalid with respect to the structure of the data type.

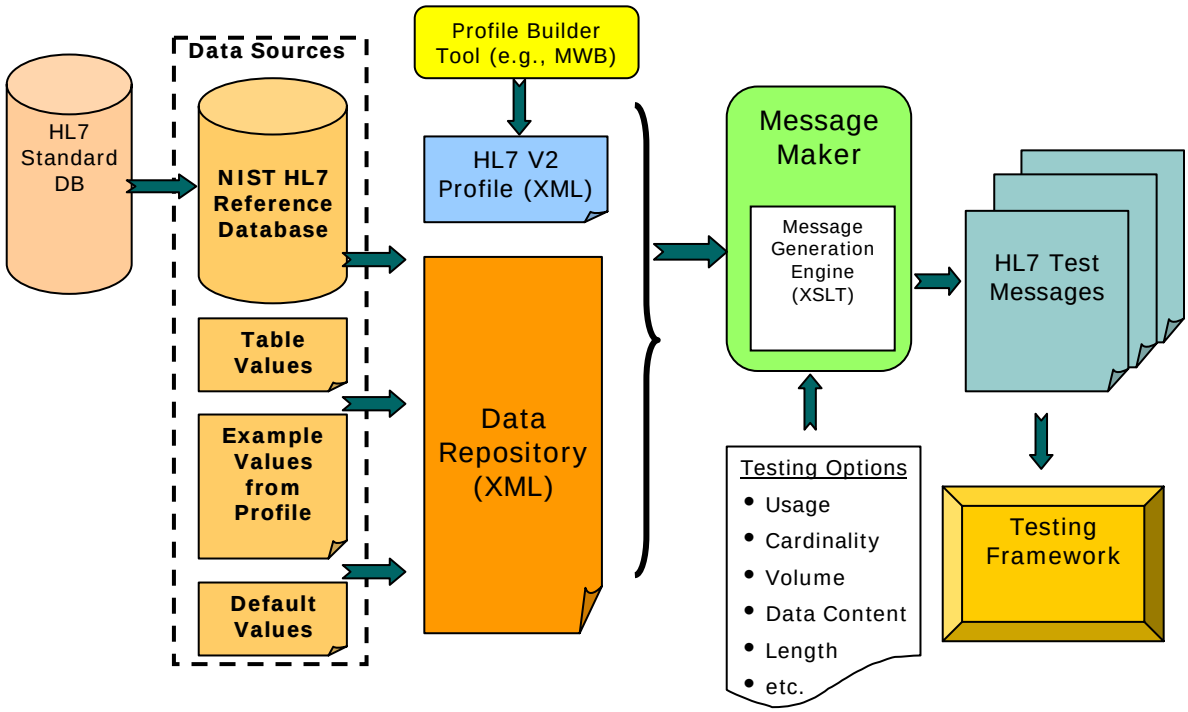

Figure 4. Message Maker Design.

Cardinality: Depending on the cardinality constraints imposed on a particular element a number of test messages can be constructed to test valid and invalid instances. For example, in the profile snippet in Figure 3, the "*Driver's License Number – Patient*" field supports exactly one instance of this element (indicated by the cardinality of 1..1). Test messages can be constructed for invalid instances by creating messages where the element will fall outside of the valid cardinality range; the two messages instances are:

- The Driver's License Number – Patient element is not present
- The Driver's License Number – Patient element appears twice

Value Sets: The test message that can be constructed for an element linked to a value set is one in which the element will be populated with a data value that is not contained in the value set.

At any given HL7 installation, many interfaces (specified by message profiles) will be defined and need to be tested. As described, numerous test messages can be constructed for each element in a given message profile. Hand-crafting an adequate set of test messages becomes a daunting task; in many cases is cost prohibitive. We next describe our automated and self-adapting approach that we have developed to address this challenge.

3 Message Maker Design

Figure 4 illustrates the conceptual design process for constructing test messages. The message profile provides a guide for which messages instances can be generated. The Message Generation Engine (MGE) reads the XML profile and will generate a message instance based on the *map* provided by the message profile and the instructions given by the test options settings. The test options can specify that

random or specific test messages are to be generated. For random requests, the MGE will select the test message type and the location of the test within the test message. For specific requests, the MGE can be instructed to create a test message of a specific type and at a specific location.

Message Maker dynamically constructs message instances while parsing a message profile. Data values for the primitive elements (i.e., fields, components, and sub-components) defined in the profile are obtained from a number of data sources. These include a database of HL7 primitive data items, HL7 tables, local tables, example values from the profile, and default values. These sources make up the data repository which is drawn upon by the MGE during the construction of a message. Utilities provided by Message Maker allow data items in the repository to be added, deleted, or modified. Each data item has associated attributes that can be used by the MGE in its processing. For example, data items can be declared as *configuration data items* which indicates that only one value exists and it must be used in every message that is created. This is an important feature since some information in the message header segment is site-specific.

Message Maker can create messages that can 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. Data content variation is achieved by randomly selecting items from the HL7 items database.

For each element the MGE reads the name, constraints, and test options settings and then performs a predetermined set of actions. For example, if we look at the "*Driver's License Number*" component, the profile *instructs* the MGE to populate (it has usage R, for required) this element with a *Driver's License Number* value randomly selected from the data repository. The MGE will ensure that the value is of ST (String) data type and does not exceed 100 characters. This example assumes that *validity indicator* in the test options setting for this element is set to "valid".

```
<Component Name="Driver's License Number" Usage="R"
Datatype="ST" Length="100">
```

If the test options setting had indicated that a usage error was requested for this element, then the MGE would not populate this element with a data value. In this case the rule for an R usage error is to *change* the usage to Not Supported (X). Similarly, test type rules are defined for each of the constraints given in the message profile specification.

An important feature in this design is that for each message instance that is generated, associated metadata is given that describes the purpose of the test message and the element location in the message that is of interest. This information is very useful during the testing phase.

The core engine of Message Maker generates messages in XML. These messages can be subsequently transformed into the native HL7 ER7 format. In addition Message Maker can display messages in a convenient tree structure for browsing and editing. See the Message Maker User's Guide for a complete description on all the features and how to use the tool [6].

4 Conformance Testing

We have described a methodology and tool for automatically constructing test messages for any given message profile. The test messages provide an integral part of a conformance testing system. The test messages can be used to construct test cases and ultimately test suites. A test case is a sequence of operations needed to perform a single conformance test. These operations may include initial setup, sending the message to the implementation under test, evaluation of the response, etc. A set of test cases defines a test suite that can be used to assess the overall conformance of an implementation. Future work will include building tools to automatically create the test cases and the development of a test framework to execute the conformance tests.

For a realistic message profile the number of derivable message instances that can be generated is extremely large—it is not practical to create them all. An issue for conformance testing is what constitutes a reasonable and adequate set of test messages to sufficiently assess an implementation. We are currently working on automated methods to extract a subset of test messages that strike a balance between reasonable coverage and the number of message instances that are generated.

5 Summary

Today's standards are often large and complex, and they have gained widespread industry support through universal and all-inclusive designs with many optional features. However, this approach often results in standards that are difficult to sufficiently constrain to provide a single and consistent interpretation. The overall effect inhibits plug-and-play installations, which both industry and user desire. Systems become hard to implement and debug, resulting in undue costs and narrowed utility. The market is smaller than one might expect with full interoperability. One solution has been to adopt profiles that specify a proper subset of the standard that specifically states the optional constructs and processing rules. Profiles are the key to standardization at the implementation level and the promotion of plug and play (interoperable) systems. To ensure interoperability among systems, installations must be implemented correctly—conformance testing is

essential. However, since profile creation is essentially unbounded, developing conformance tests is problematic. Automated tools are necessary.

The HL7 messaging standard typifies issues described above. 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. Message profiles provide the mechanisms that allow for implementations to be tested for conformance. However, it doesn't eliminate the complexity or the enormous possible interpretations of the standard. Each site-specific implementation must be tested. Automatic and dynamic testing tools are essential. We have developed a methodology to generate conformance tests and have demonstrated its utility through the implementation of Message Maker. In practice Message Maker has been used to create test messages for real world message profiles. The tool significantly reduces the time and effort in conducting conformance tests that lowers costs and improves the reliability of healthcare systems.

6 References

[1] Health Level 7 (HL7); http://www.hl7.org

[2] HL7 Standard Version 2.4, ANSI/HL7 V2.4-2000. October 6, 2000.

[3] HL7 Standard Version 2.5, ANSI/HL7 V2.5-2003. June 26, 2003. Chapter 2 Section 12 "Conformance Using Message Profiles", pg 43-62.

[4] Message Workbench (MWB). Developed by P. Rontey at the U.S. Veterans Administration; http://www.hl7.org. Conformance Special Interest Group.

[5] Message Maker; Developed by the National Institute of Standards and Technology (NIST). http://www.nist.gov/messagemaker.

[6] Message Maker User's Guide. Version 1.4 January 2006, R. Snelick, L. Gebase, S. Henrard. http://www.itl.nist.gov/div897/ctg/messagemaker/docs/UsersGuide1.4.pdf.

[7] Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004. http://www.w3.org/TR/2004/REC-xml-20040204.