Message-Independent Test Harness for Content-Level Test of B2B Standards: An Application Information Mapping Test Case Study

Junho Shin¹, Jaewook Kim¹,², Nenad Ivezic¹
¹Manufacturing Engineering Laboratory, National Institute of Standards and Technology
²Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
junho.shin@nist.gov, jaewook@nist.gov, nivezic@nist.gov

Abstract
NIST has been developing a B2B interoperability testbed to provide tools that facilitate interoperability testing of B2B applications. One of its capabilities focuses on content-level testing in terms of B2B message interpretation. Currently, test harnesses are available only for messages in XML format. Consequently, content testing of messages in other formats such as EDI and ASN.1 cannot be performed. This paper proposes a message-independent test harness based on the NIST Message Metamodel – a neutral form of messages – and a formal model of test guidelines. The paper illustrates our approach on a set of EDI logistics messages commonly used in long distance supply chains.

Keywords: B2B Interoperability, Content-level test, B2B message-representation standards, NIST Message Metamodel

1. Introduction
One way of ensuring business-to-business (B2B) interoperability is to test conformance of the participating applications with B2B interface and messaging standards. The NIST B2B interoperability testbed has been developed for such tests [1, 2]. In particular, a number of content-level tests, which focus on B2B message interpretation, have been proposed. The Application Information Mapping Test (AIMT) [3] is one of those tests. It is designed to ensure that any message element is interpreted according to its intended meaning in an Application Under Test (AUT).

The AIMT was designed to test only XML-based messages that conformed to the OAGIS BOD (Business Object Document) messages [4] created using the W3C XML Schema language [5]. It could not test other commonly used message-representation standards such as EDI [6], ASN.1 [7]. This paper presents a novel, advanced message-independent test harness for AIMT based on the NIST Message Metamodel [8] and the AIMT Guideline Metamod. The Message Metamodel is a concise, syntax-neutral representation form of B2B messages and B2B message specifications. Splitting the testbed into message-independent parts and message-dependent parts by using the Message Metamodel and AIMT Guideline Metamodel improves the reusability of testbed modules.

The paper is organized as follows. In chapter 2, we first give an overview of AIMT and the concepts of the Message Metamodel. In chapter 3, we describe the message-independent AIMT harness that is based on the Message Metamodel and on the AIMT Guideline Metamodel. In chapter 4, we detail the proposed AIMT Guideline Metamodel, as a generic definition independent to various message-representation standards. In chapter 5, we address implementation results of the message-independent AIMT harness. Finally, we provide concluding remarks in Chapter 6.

2. Related Works

2.1. Application Information Mapping Test

The goal of AIMT is to assure that B2B message elements are interpreted correctly by the AUT – correctly means according to the meaning specified by B2B participants. AIMT provides the proof of correct interpretation by verifying the mapping between the message element and its corresponding concept in the AUT, defined as local data element.

Two types of tests have been proposed: input and output. The input test analyzes the mapping between the elements of an input message and their corresponding local data elements (See Figure 1(b)). Conversely, the output test analyzes the mapping between the elements of
an output message and their corresponding local data elements (See Figure 1(c)).

Verifying such mappings between message elements and local data elements directly is difficult. So, we developed an alternative method that checks the equality between input message instance and output message instance. The assumption is that an inequality between input and output message is rooted only in an invalid mapping. The process involves a human who maps the message content from its local representation to the standard message-representation for the input test, and vice-versa for the output test. Finally, the analyzer creates the test report by comparing the message instance initially created with the message instance delivered back from the AUT.

2.2. NIST Message Metamodel

The NIST Message Metamodel is a neutral form for representing message syntax. The Message Metamodel is only briefly summarized below while details are given in [8].

The Message Metamodel has three parts: the schema part, the configuration part, and the message part. The schema part captures the naming, structure and value concepts that are present in the corresponding message schema language, such as the XML Schema. The configuration part allows trading partners or communities to define usage rules for, and other modifications to, the standard schemas. The message part captures the elements present in each message instance and elements’ association to its definition, which is captured in the schema model. The details of the schema part and the message part will be described further.

The schema part shown in Figure 2 is used to capture message definition concepts, or schema concepts. The root concept is a Schema. Schemas define ContentModels and Components. A Component represents definition of message elements (ElementModel) and definition of attributes (AttributeModel). The Component has associated contentModel, which is either (1) a StructuredContent that is a definition of a complex and structured message-component type, or (2) a SimpleContent that is a definition of simple message-component types.

The message part shown in Figure 3 is used to capture the runtime message concepts. These concepts include the structures and data elements that appear in the actual exchange messages. An Element is (1) a StructuredElement containing a sequence of Items, most of which are other Elements or a (2) a SimpleElement containing the message information unit – a SimpleValue. The Element may contain Attributes, which also capture message information
units. Every Element and Attribute corresponds to its model (ElementModel and AttributeModel, respectively) as captured by the schema part. In some cases, the Element can be specified by some ContentModel that is used only in a particular instance (e.g. xs:any XML Schema type).

3. A Message-independent Test Harness

Content-level tests focus on the validation of the contents of B2B application message exchanges. The Message Metamodel enables the content-level test to deal with the messages regardless of their syntax. In addition, test harness based on the Message Metamodel can be reused even if that syntax changes.

Figure 4 shows that test harness. Instead of treating a message instance of specific syntax directly, the test message instance generator and the message analyzer only deal with the neutral-form (N-form) message instance specified by the Message Metamodel. Thus, the only effort necessary for supporting any target message-representation syntax is to prepare message-specific translators into and out of the neutral form.

4. A formal Guideline Metamodel

Using the N-form message instance also enhances the reusability of the existing test case management module – a test case parser. For example, the verification conditions embedded in the test case use XPath in the original AIMT harness, which was designed to test XML BODs, could not be used to test EDI messages. However, N-form message instance embedded in the test case specification can now be used for both.
formal AIMT guideline - by defining its metamodel - will provide a reuse capability similar to the one described in the preceding section.

Figure 5. AIMT Guideline Metamodel

Figure 5 illustrates the key concepts of this metamodel. The AIMTGuideline principally consists of the unit of guidelines, defined as UnitGuideline. The UnitGuideline is specified for every message element that corresponds to the ElementModel or the AttributeModel of Message Metamodel. Consequently, it has a 1:1 relationship with AbstractContextPath, a reference path to the message element. The UnitGuideline contains two types of information: TestDataMetadata and TestRule. The TestDataMetadata captures the usages of message elements from which test data that satisfy test requirements can be generated. The TestRule that consists of Condition and Assertion concepts captures the business rules to check the validity of the message instances. Any message instance either present in a running AUT or generated from the test data metadata can be an artifact to be validated against the rules.

The schema model of TestDataMetadata is shown in more detail in Figure 6. The rationale for developing the detailed schema model is to reuse the test data generation method based on the test data metadata definition for other interoperability testbeds. The TestDataMetadata concept is applied to MICContextPath, a reference to the element of a particular message instance. To this context, the MICContextPath has the notation where the first path element in its path indicates the message instance to which the specified usages apply. For example, a MICContextPath, ‘/SyncShipmentSchedule[1]/*/*/ShipToParty/Location’ indicates the ‘Location’ element in the first generated ‘SyncShipmentSchedule’ BOD message instance.

The TestDataMetadata is of two kinds: ValueMetadata and OccurrenceMetadata. The ValueMetadata serves as a value generator that creates an element value, as specified for the element in a particular test instance. By definition, the ValueMetadata can be composed of multiple metadata functions for generating values and operators between them. Normative expression in the model consists of one initial MetadataFunction and of zero or one FunctionExpression that has an operator and one associated MetadataFunction. The associated MetadataFunction can have its

<table>
<thead>
<tr>
<th>Table 1. Metadata functions for test data metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value (String <em>/ String)</em></strong></td>
</tr>
<tr>
<td><strong>IntegerByLength (int)</strong></td>
</tr>
<tr>
<td><strong>IntegerByRange (int, int)</strong></td>
</tr>
<tr>
<td><strong>UserSpecified()</strong></td>
</tr>
<tr>
<td><strong>ValueByPath (MICContextPath)</strong></td>
</tr>
<tr>
<td><strong>Datetime()</strong></td>
</tr>
<tr>
<td><strong>DatetimeBefore (dateTime)</strong></td>
</tr>
<tr>
<td><strong>DatetimeAfter (dateTime)</strong></td>
</tr>
</tbody>
</table>
FunctionExpression in a recursive manner. The MetadataFunction contains ValueExpressions as parameters of the function, which can be either a normal value (Value in the model) such as an integer value or the MetadataFunction itself.

As shown in Table 1, nine types of metadata functions have been designed to specify what kind of values can be generated. For example, an instance of ValueMetadata concept with two metadata functions, ‘Value (Fragile. Handle with care. Care level) + IntegerByRange (1, 10)’, is supposed to generate a value with a string ‘Fragile. Handle with care. Care level’ and any integer value between ‘1’ and ‘10’.

The OccurrenceMetadata specifies the number of occurrences of the message element in a particular test message instance. The model notes that the associated message element whose path in the test message instance is within MIContextPath (i.e., is a self or a descendant of a node in the node-set) should occur as many times as specified in the numOfOccur property.

5. Implementation of the message-independent AIMT harness

We have implemented the message-independent AIMT harness and AIMT Guideline Metamodel to deal with EDIFACT messages, which are used to obtain logistics visibility during ocean freight shipping [11]. Several message-independent modules - test case parser, test MI generator, and analyzer that can cope with new AIMT Guideline Metamodel and Message Metamodel definitions - have been implemented to configure the message-independent AIMT harness. The translator modules that are necessary to support the translation of message schemas and message instances to and from Message Metamodel form are available as software libraries. Plugging these software modules into the existing testing infrastructure was enabled by defining three common interface classes: (1) a class for reading a message schema, (2) a class for converting the N-form message instance to a EDIFACT message instance, and (3) a class for converting a EDIFACT message instance to the message instance model.

Using a Java-based modeling tool [12], we were able to automatically generate class definitions and their basic encapsulated methods definitions from the AIMT Guideline Metamodel (Note that the generation of Message Metamodel classes have been done in a same manner, which is, however, not the scope of this paper). Instead of developing or refining all the components of the testbed, the test engineer only needed to incorporate the EDIFACT syntax-specific parser into the AIMT.

The paper proposes the message-independent AIMT harness that utilizes the NIST Message Metamodel and AIMT Guideline Metamodel. This approach provides test harness independence of a particular message-representation standard and consequently enhances its reusability. The new configuration to minimize the dependency on particular message-representation standards will allow the AIMT to respond to emerging test requirements of B2B environments more rapidly and flexibly.

7. Acknowledgement

Any mention or use of commercial or open source software products within this paper is for information and demonstration only; this does not imply recommendation or endorsement by NIST, nor does it imply these products are necessarily the best available for the purpose.

8. References