Content-Level Conformance Testing: An Information Mapping Case Study

Boonserm Kulvatunyou, Nenad Ivezic, and Albert T. Jones

National Institute of Standards and Technology, 100 Bureau Drive,
Gaithersburg, MD 20899, USA
{serm, nivezic, jonesa}@nist.gov

Abstract. Content-level conformance testing is a key to achieving interoperable data exchange among applications deployed across collaborating, yet independent enterprises. In this paper, we identify four types of content-level conformance tests to support interoperable data exchange: document-verification tests, information-mapping tests, transaction-behavior tests, and scenario-based tests. We describe in substantial detail our experience with information-mapping tests within an industrial B2B integration effort. We review different approaches to information-mapping conformance verification including logical consistency checking, human-computer interaction, and event-based checking. We adopt the human-computer interaction approach and describe a test-case generation methodology. The methodology details modeling, test requirements specification, abstract test-case definition, and, ultimately, executable test-case generation. Lastly, we provide experimental results of applying our methodology in the context of an automotive industry development of data exchange standard for interoperable inventory visibility applications.

1 Introduction

The research study described in this paper is a result of a large business-to-business (B2B) integration initiative called Inventory Visibility and Interoperability (IV&I). The initiative is a collaboration that includes the Automotive Industry Action Group (AIAG) [1], its member companies, and the Manufacturing B2B Interoperability Testbed at the National Institute of Standards and Technology (NIST) [2]. A key objective of this initiative is to enable different tools supporting vendor-managed inventory (VMI or Inventory Visibility (IV)) to interoperate using an internet-based B2B integration infrastructure. In this paper, we explain the role of content-level conformance testing in achieving this objective, detail the testing methodology, and provide results from applying the methodology to one type of content-level conformance test: the information-mapping tests. In Section 2, we present the interoperability problem addressed in the IV&I project. Section 3 summarizes the content-level conformance tests needed to support interoperability. Section 4 provides an overview of the information-mapping test, alternatives to the information-mapping conformance verification, and the rationale for our selected approach. Section 5 describes the test-case generation methodology for the information-mapping test.

F. Khendek and R. Dssouli (Eds.): TestCom 2005, LNCS 3502, pp. 349 – 364, 2005.

Section 6 illustrates experimental results from interaction with participating IV tool vendors. Finally, Section 7 concludes the paper by summarizing key research results.

2 Inventory Visibility and Interoperability Problem

Currently, the automotive customer companies typically require their suppliers to monitor customer inventory and replenish parts using IV tools that use proprietary formats to exchange data. Consequently, each supplier needs multiple IV tools to communicate with multiple customers. Fig. 1 shows the current status of the IV tool usage that involves costly data exchange using proprietary formats.

Fig. 2 shows the target usage scenario where IV tools interoperate in a federated architecture. In this case, each supplier needs only one tool since the IV tools exchange inventory data using a standard message called SyncQuantityOnHand (SQOH). SQOH is a Business Object Document (BOD) based on a standards specification from the Open Application Group Integration Specification (OAGIS) [3]. These OAGIS specifications use the eXtensible Markup Language (XML) format [4].

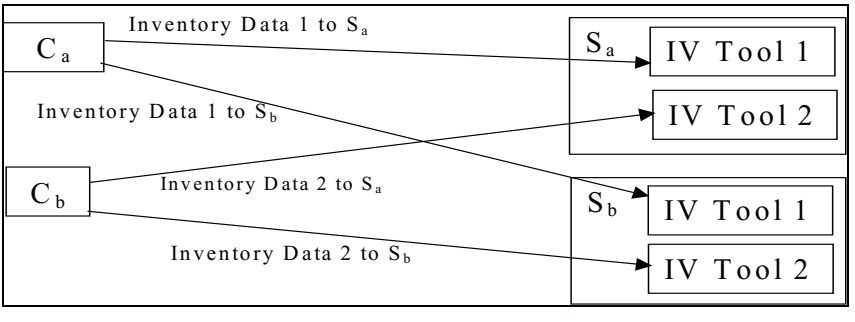

Fig. 1. In the current status of the IV tools usage, customer C_a sends inventory data in a proprietary format (Data 1) to IV Tool 1, while C_b uses another format (Data 2) with IV Tool 2. Both C_a and C_b trade with suppliers S_a and S_b that, consequently, need to employ both IV tools

Before interoperability can be achieved among IV tools, it is important to assure that each IV tool consistently uses the data in the SQOH BOD specification. This is the purpose for developing the content-level conformance testing methods. It should be noted that these conformance tests are not necessary in the Fig. 1 scenario, since the IV tools only operate using their own proprietary representations.

3 Types of Content-Level Conformance Testing

Content-level conformance testing is a function testing (also called functional testing) [17] focusing on the application-level interoperability in the B2B stack [14]. We classify the conformance tests at the content level into one of the following four types that are equally important and complementary: (1) document-verification tests, (2) information-mapping test, (3) transaction-behavior test, and (4) scenario-based test.

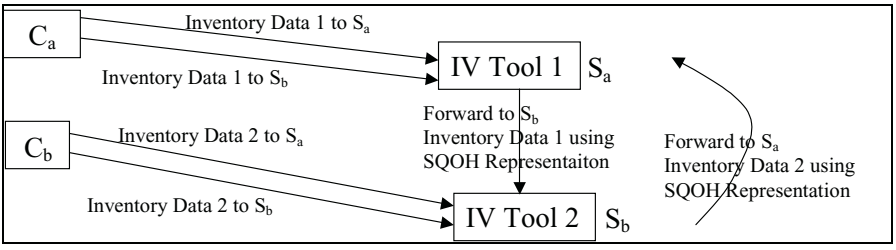

Fig. 2. The figure illustrates the target scenario for the IV tools usage. Customer C_a may still send inventory data in its proprietary format (Data 1) to IV Tool 1, while C_b may still use another proprietary format (Data 2) with IV Tool 2. However, both IV Tool 1 and IV Tool 2 are capable of exchanging the data using the SQOH BOD standard message. Therefore, suppliers S_a and S_b. may use only a single IV tool

3.1 Document-Verification Test

The document-verification test spells out structural, syntactic, and semantic rules that must hold for a message instance containing data exchanged between applications. A document-verification test includes a generation and a consumption test components. In the generation case, the tests verify whether the application under test can generate minimally valid message instances. In the consumption case, the tests verify whether the application under test appropriately consumes minimally valid message instances. In both cases, valid means structurally, syntactically, and semantically correct. This test is a pre-requisite for the three remaining tests.

3.2 Information-Mapping Test

The information-mapping test validates that the intended usage of exchanged data within the application under test in fact conforms to the agreed upon shared semantics, which is declared in the relevant content standard specification. For example, the SQOH specification includes data definitions for Customer Id, Inventory Id, and Storage Location Id. It is important that the applications exchanging the inventory data 'interpret' the data the same way. This means that they must support the use of the data in a manner consistent with the content standard specification. On one hand, an application may interpret 'Customer' as a manufacturing plant, 'Inventory' as the quantity on hand at a specific building that has associated delivery docks within a manufacturer's plant, and 'Storage Location' as the quantity on hand at specific bins or an area within the site. Another application, however, may interpret the 'Customer' as an OEM who has multiple manufacturing plants, 'Inventory' as quantity on hand at a specific plant (which has one or more buildings), and 'Storage Location' as a building within a plant. Such inconsistent interpretation of the business data by two applications could cause the execution of inappropriate business actions.

3.3 Transaction-Behavior Test

The transaction-behavior test focuses on transactional response of the application under test. In the transaction-behavior test, responses based on transactional success,

partial success, and failure conditions are verified. These conditions are referred to as business rules, which must be agreed upon as part of the content standard specification. For example, there is a requirement for a protocol that specifies whether passing inventory information about an item, which is unknown to one IV system, should be treated as a success or a failure by another IV system. In Fig. 2, if IV Tool 1 considers the above circumstance a failure condition, while IV Tool 2 accepts partial successes, Tool 1 might end up resending duplicate data. In addition to testing the behavior based upon those business rules, we must test for the proper response, and the proper set of follow-up behaviors to other typical error conditions such as boundary conditions of the field's domain [16, 18]. Transactional behavior at the content level may potentially interact and be confused with the transport protocol level behavior. The boundary between the two levels needs to be well established before interoperability can be achieved.

3.4 Scenario-Based Test

While the previous types of tests typically involve only one business document at a time (not counting the transactional response document), the scenario-based test typically involves multiple business requests and responses. The scenario-based test seeks to verify the business logic of an application that must be common across data exchange steps and participating partners. It focuses on testing a high-level control flow of business actions and consequences. Transaction flow analysis [16] and cause-and-effect graphing [13] on an agreed upon business process could help generate test cases. For example, in some instantiations of the IV Min/Max business scenario, a business response to a shipment notice is expected within a certain time period when the inventory data shows that the available quantity is below a specified minimum level. In others, a receipt notice and purchase order transactions that are handled by multiple, segregated applications are required. As these examples show, scenario-based testing can be very involved, difficult to perform, and hard to verify [5].

4 Information-Mapping Test

Information-mapping tests have been developed for other data exchange standards such as the ISO 10303, informally known as, Standard for Exchange of Product Data (STEP) [6]. Building on the STEP testing methodology, we propose the addition of two steps to verify the information-mapping conformance: the Input Test and the Output Test [7]. The Input Test verifies that the application can read and correctly interpret the standard representation. The Output Test verifies that the application can translate correctly from its internal representation to the standard representation.

4.1 Overview of the Test Procedure

Fig. 3 illustrates the general procedure for the two testing steps. The outputs from both the Output Test and Input Test go through a verification process to determine conformance by checking syntactic and semantic integrity against the inputs.

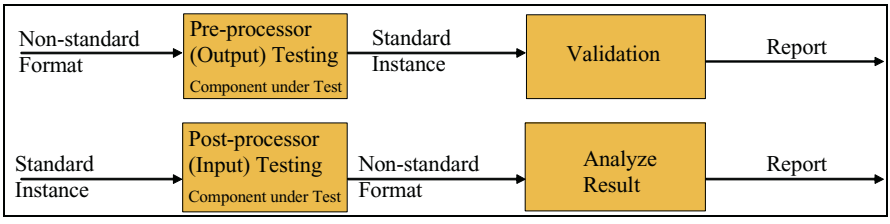

Fig. 3. General procedure for information-mapping test consists of the Output Test and Input Test. The Output Test requires the test data in a proprietary, non-standard format. The output from the Output Test is translated to the target standard representation, the standard instance. The input to the Input Test is the test data in the target standard representation, the standard instance, and the translated output is a non-standard format, a proprietary representation

4.2 Information-Mapping Test Challenges

Challenges associated with the information-mapping test include effective approaches for (1) creating the non-standard format as an input to the Output Test, (2) analyzing the non-standard format output from the Input Test, and (3) validating the standard instance from the Output Test.

Applications under test typically can read the target standard representation and at least one proprietary representation. Since the capability to read different formats varies among components under test, a straightforward way to handle this issue would be to create and maintain test data in multiple representations. However, this is a costly commitment. Similarly, analyzing different non-standard outputs from the Input Test becomes an unmanageable task. The level of difficulty in validating the output from the component under test is related directly to the target standard representation: the more flexible the standard is, the more complicated the validation. On the other hand, the more formally expressed the standard semantics are, the easier the validation.

To deal with the first and second challenges, we investigated a 'reflexive' testing approach (shown in Fig. 4) for both the Input and Output Tests. This approach would resolve issues stemming from proprietary representations because we would have to deal with the target standard representation only. However, it turns out that this approach only verifies the integrity of data passing between the input and output interfaces but not the data 'interpretation' within the application. The approach is easily compromised as it leaves the critical mapping mismatch undetectable (see Fig. 5).

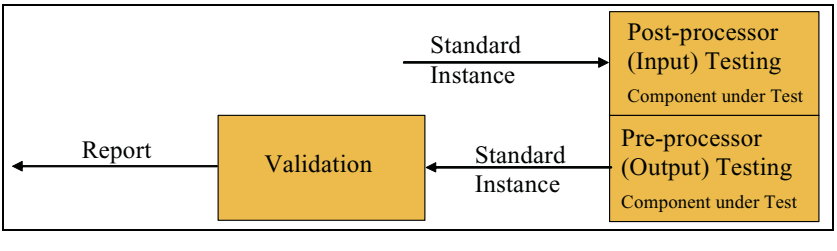

Fig. 4. An initial 'reflexive' testing approach was insufficient to detect mapping mismatch

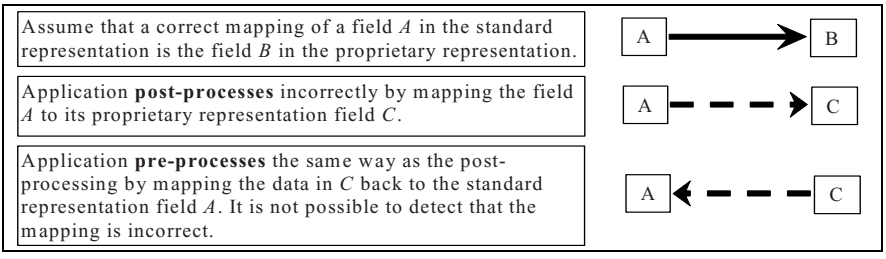

Fig. 5. The reflexive testing approach cannot detect incorrect mapping from field *A* to field *C* (i.e., coincidentally correct mapping error)

Our refined approach followed the argument that if either the Input Test or Output Test is verified to conform to the specification, then the other can rely on the reflexive testing approach. Fig.6 illustrates the proof for this argument when the Input Test is first verified to conform. It can be shown that the same conclusion holds when the Output Test is performed first. If this approach holds, then the challenge #1 is eliminated: we need to create test data only in the target standard representation. This is a major improvement from the STEP testing approach where test data are engineering graphics on the pieces of paper. In the Output Test, an engineer would draw the graphic using the tool under test and then push out the graphic data in the target standard representation.

In the next three subsections, we describe the three potential approaches to address the second and third challenges: to verify conformance of the Input and Output Tests.

4.2.1 Logical Consistency Approach

The logical consistency approach based on formal ontologies may be used to verify the results of both Output and Input Tests. However, an ontology of the standard terminology is needed for the Output Test and another ontology for each application vendor's proprietary terminology is needed for the Input Test. In practice, however, such ontologies are rarely available. Logical consistency alone may only confirm validity of a document but not correctness of the mapping. Additionally, verifying the correctness a document requires necessary and sufficient conditions for targeted terms. However, sufficient conditions often cannot be expressed nor validated from the data perspective alone but may need to be expressed in terms of business events.

Consider the term *ReceivedDate,* a data field in the SQOH BOD. Customers typically update this field whenever they receive a shipment of ordered goods from a supplier. Logical relationships between the *ReceivedDate* and other fields in the SQOH BOD may be established such that *ReceivedDate* must be before the BOD's current date. A better definition of the field would relate this field to be on or after the *ShippedDate* in the latest shipment BOD from the supplier. This requires information from another transaction. For some test cases, this information may be available; in others it may not. The two axioms about the *ReceivedDate* still represent only necessary conditions. A sufficient condition may be that the *ReceivedDate* correspond to the Date and Time at which the item is recorded into the inventory. If the item has to be inspected before it is considered received, then this sufficient condition involves

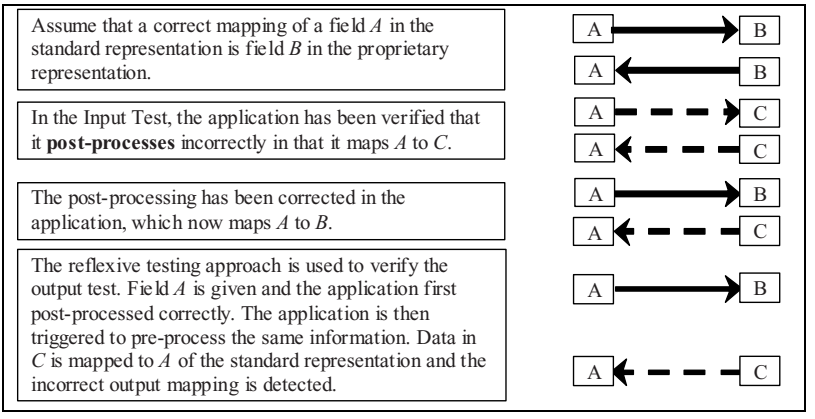

Fig. 6. In the refined testing approach, we rely on showing that Input Test is verified before using the reflexive testing approach

the execution of an event and requires knowledge about the inspection time. If the event is not broadcast and recorded somewhere, then there is no reference data to validate the condition. On the other hand, a research is being conduct in our team to combine the logical consistency approach with the model-based instance equivalence measurement as a test verification method [19]. The research have demonstrated that the logical consistency coupled with the instance equivalence measure performs well when sufficient conditions can be bound or assumed and have suggested a context where the assumption may hold.

4.2.2 Human-Computer Interaction Approach

In this approach, the user or the application developer manually encodes the data from its proprietary representation into the target standard representation for the Input Test. Therefore, the output from the Input Test will already be in the target standard representation. In such cases, the conformance verification can rely on one representation that is the target standard representation for both the Input and Output tests. Here we circumvent the second challenge. Fig. 7 illustrates this approach associated with the Input Test, while the Output Test is based on the refined reflexive testing approach as illustrated earlier in Fig. 6.

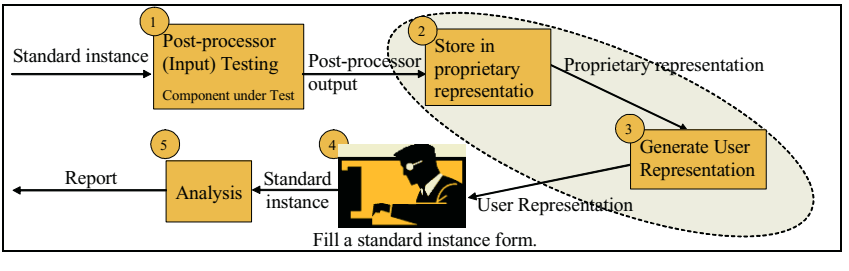

Fig. 7. Human-computer interaction-based conformance verification approach

The procedure for the Input Test would be as follows. (1) The application post-processes a test instance given in the target standard representation. (2) The application stores the data in its proprietary representation. (3) Another procedure in the application then renders the data for user consumption. (4) The user fills a new standard instance form. (5) The test verifies the syntax and semantic integrity against the original standard instance given in step 1. It should be noted that procedures in step 2 and 3 are treated as a black box, which means that we assume that the two steps, particularly the rendering of data onto the screen, are done correctly. In effect, we are testing the mapping implementation in step 1 and conceptual mapping in step 4. If a mapping mismatch is found in step 5, the vendor has to determine if the problem lies in the post-processor (step 1) or the conceptual mapping (step 4).

We note that this approach is not foolproof from the standpoint of coincidental correctness. Some incorrect mappings may still get through undetected if the vendor behavior in step 4 coincides with the symmetric mapping as described in Fig. 5. However, the chances of coincidental correctness decrease as there are two more mappings in between, from the proprietary representation to the UI representation and from the UI representation to the output. The test will have higher fidelity if a user who has no knowledge of the pre- and post-processing interfaces conducts the test.

4.2.3 Event-Based Approach

The event-based approach relies on one or more sequences of messages to trigger some events in the component under test whenever data changes. If the component under test triggers the events as expected, then it may conform to the specification. The event-based approach has a severe limitation in that there are few event-triggering fields in a typical exchange messages such as BODs. The test is harder to generate and automate the execution because multiple messages may be needed before the event is triggered and some events may be actual physical events.

5 Information-Mapping Test Case Generation Method

In this section, we describe the procedure to generate test cases for the information-mapping conformance test. As described in [8], the heart of this procedure is the business-case. The business case definition is described in Section 5.1. Then, a sample mapping table is described in Section 5.2. Finally, the detail test case generation is described in Section 5.3.

5.1 Business Case Definition

Business Case Definitions specify requisite constraints among the message elements and attributes in terms of usage occurrence and tool-support indicators.

The ‘usage occurrence’ for a BOD indicates the minimum and maximum allowable occurrences for each element/attribute in the context of particular data exchange (e.g., the IV&I project). These occurrence constraints are different from those expressed in the BOD schema because they reflect additional requirements. The occurrences of

each element/attribute are specified conditionally on their parent elements. For example, within a SQOH document schema, the *ItemStatus* (parent) element may have a usage occurrence of 0, while the *ItemStatus/Code* (child) element may have a usage occurrence of 1. The meaning is that the *ItemStatus/Code* element must occur if the *ItemStatus* element occurs; otherwise, the *ItemStatus/Code* element must not occur. The following notation applies:

- **0** means an optional element/attribute that may occur 0 or 1 time.
- **C** means a conditional optional element/attribute may occur 0 or 1 time, based on conditions involving elements/attributes beyond the occurrences of their ancestors.
- **1** means a required element/attribute that must occur once and only once.
- **0+** means an optional element/attribute that may occur zero or more times.
- **C+** is similar to **C** where an element/attribute may occur multiple times.
- **1+** means a required element/attribute that must occur at least one time.

The 'tool support' indicates optionality of elements/attributes from a functional-requirements perspective and drives the definition of the business cases for testing purposes. If the field's usage occurrence is required (1 or 1+), that field always requires tool support (S). If the field's usage occurrence is optional or conditionally optional (0, 0+, C, C+), the tool support indicates whether the tool must be able to process the field, if it occurs in a message. The following notation applies:

- **S** means mandatory tool support for the field, i.e., the tool must be able to store, process, and/or interpret the field.
- **NS** means optional tool support for the field, i.e., the sending tool may not expect the receiving tool to interpret, process, and/or store the field.

The S and NS tool support indicators are also interpreted conditionally on the parent of the element/attribute in the same way as the usage occurrence. All the fields with mandatory tool support constitute one or more Base Business Cases dependent upon the optional and conditional usage occurrences.

5.2 Mapping Tables

Mapping Tables specify mappings between each XML-based message element/attribute and an intended vendor tool interface. Table 1 shows a mapping table example with usage occurrence and tool support specifications. Each row of the 'Element' column is an XPATH language representation of the document structure [9]. The row with the bold type font represents an aggregate (complex type) element, which has children elements/attributes. The 'Vendor Support' column shows a vendor support of each document schema element/attribute. The difference between the Tool Support and the Vendor Support suggests an additional implementation requirement for the vendor to satisfy the user's functional requirements. For example, the vendor support of the *From* and *To* components of the *EffectivePeriod* but not of the *Duration* component is a potential problem since the 'Tool Support' column indicates all three elements must be supported by the tool.

Table 1. An example mapping table with usage occurrence and tool support definitions

Element	Description	Usage Occurrence	Tool Support	Vendor Support
Item/CustomerItemId	Customer part number	1	S	Yes
Item/CustomerItemId/Id	Customer part number	1	S	Yes
Item/CustomerItemId/Revision	Part revision number	0	S	Yes
Item/EffectivePeriod	The period part will be in production	C	S	Yes
Item/EffectivePeriod/From	Start date of part production	C	S	Yes
Item/EffectivePeriod/To	Planned end date of production	C	S	Yes
Item/EffectivePeriod/Duration	Planned duration of production	C	S	No
Item/EndEffectiveQuantity	Planned part cumulative quantity	C	NS	No
Item/AvailableQuantity	Quantity available for production	1	S	Yes
Item/MinimumQuantity	The minimum inventory the customer wishes to have on-hand.	1	S	Yes
Item/MaximumQuantity	The maximum inventory the customer wishes to have on-hand.	1	S	Yes

5.3 Test Cases Generation Procedure

As mentioned previously, mandatory tool support specification defines one or more Business Cases with different combinations of optional and conditional elements/attributes. The specification of business cases defines testing requirements for the IV&I conformance tests.

Prior to test requirements generation, we must specify possible IV&I profiles (i.e., valid combinations of Tool Support and Conditional fields and type of data will be used such as language, standard identification code, and standard code lists). The IV&I profiles determine which individual business case makes sense to support from the business requirements standpoint. Once the profiles are determined, test requirements are created to indicate data elements/attributes that must appear in test cases.

Table 2 includes some examples of business cases and associated test requirements (TR). The numbers in the test requirement columns are 'Occurrence in Test'. The possible values are 1, 1+, or 0, which indicate whether the field will be instantiated in the test data once and only once, once or more, or not at all. Business case 1 represents a baseline functional requirement as indicated in the Tool Support and the Usage Occurrence columns. In the example, the base case has the first 3 and the last 3 elements' occurrences in test equal to 1, because they all have the Usage Occurrence equal 1 and the Tool Support equals S with an exception of the *Revision* field. The Revision field can have the Occurrence in Test equal 1 in the base case, because there is no condition on its occurrence. This helps reduce the number of tests.

The *EffectivePeriod* and its child elements as well as the *EndEffectiveQuantity* have additional logic associated to deal with the plan production period or quantity; hence, they constitute the second business case. Two test requirements are necessary for the business case, because the conditions in the Usage Occurrence column indicate

Table 2. Example business cases and test requirements

Element	Usage Occurrence	Tool Support	Bus. Case 1 (Base case)	Bus. Case 2	
			TR1-1	TR2-1	TR2-2
Item/CustomerItemId	1	S	1	1	1
Item/CustomerItemId/Id	1	S	1	1	1
Item/CustomerItemId/Revision	0	S	1	0	0
Item/EffectivePeriod	C	S	0	1	1
Item/EffectivePeriod/From	C	S	0	1	1
Item/EffectivePeriod/To	C	S	0	1	0
Item/EffectivePeriod/Duration	C	S	0	0	1
Item/EndEffectiveQuantity	C	NS	0	0	0
Item/AvailableQuantity	1	S	1	1	1
Item/MinimumQuantity	1	S	1	1	1
Item/MaximumQuantity	1	S	1	1	1

that the *To* and the *Duration* elements cannot be used at the same time. We note that the mutually exclusive condition between the *EffectivePeriod* and the *EndEffectiveQuantity* fields could constitute the third test requirement in the second business case. However, the *EndEffectiveQuantity* is excluded because the user indicates that the tool does not need to support the field.

In summary, the business case concept is a logical grouping of information elements to make the tests more manageable and understandable. In Table 2, for example, the TR 2-1 could be combined with the TR 1-1 for the information-mapping test because there is no conditional conflict. This could result in a smaller number of tests.

These test requirements (together with IV&I profiles) guide test data selection, which matches sample application data with test requirements to form test data. Then, the test data are assembled in the form of abstract (i.e., independent of a specific format) test cases that match test requirements. The semantic validation rules ensure valid abstract test cases.

Before generating the executable test cases, conformance level statements are created to aggregate abstract test cases that match some conformance testing strategy. Such a strategy identifies possible aggregation of IV&I profiles and the corresponding business cases.

6 Experimental Results

Using the approach described above, we have developed test cases and executed them against two IV applications. Initially, the vendors perform the document-verification testing which is a self-test using a Reflector Tool [20]. Fig. 8 summarizes the testing approach used for the mapping test. We validated the generated BOD instances (1) using an XML parser against the schema using XML Spy 2004 [10], (2) with additional structural and semantic rules encoded in Schematron [11] using the XT 20020426a XSLT transformation engine [12], and (3) with a Schematron diff tool

using the same XT implementation. The Schematron diff tool has been developed in this project to assist the conformance verification process. The tool takes test data, such as a BOD instance, as input and generates Schematron rules that compare the BOD output from the Input or Output Test with the test data. Due to its limited capability, the tool cannot completely automate the conformance verification. For example, the current tool would raise a flag if the test data were specified in a different order from the ones in the BOD output from the test.

The rest of this section summarizes the experiment and highlights some results from the test with the IV applications using the test cases from the base business case of the SQOH BOD partially illustrated in Table 2.

6.1 Results from the Input Test

At the initiation of the test, we identified a number of mapping mismatches among the fields *Sender*, *Receiver*, *CustomerPartyId*, *SupplierPartyId*, *Inventory/SiteId*, and *StorageLocation/Id*. We discovered these mismatches right away because they were used for authentication and authorization. The BOD development experts define the *Sender* as the OEM, the *CustomerPartyId* as the OEM plants, the *Inventory/SiteId* as a pointer to an inventory facility inside the customer plant, and the *StorageLocation/Id* as a location within an inventory facility. This means that an OEM can update "on hand data" at the level of plant, building within a plant, and location within a building. On the other hand, the IV tool under test interpreted the *Sender* to be the same as the *CustomerPartyId*, which points to the OEM (the *Sender* serves only as routing information), *Inventory/SiteId* as pointing to the OEM's plant, while *StorageLocation/Id* is an identifier for arbitrary locations within the plant. These mismatches were later resolved with the team of business process experts to match interpretations suggested by the IV tool vendors. In addition, the XML parser validation and the Schematron rules validation indicated that a required field, *Inventory/LastModification DateTime*, was missing.

The Schematron diff also raised flags, which indicated either a mapping mismatch or a representation mismatch in a number of fields. Table 3 lists these fields and provides a list of input and output values for the tool under test. Table 4 lists the concerns raised in each case and their resolutions, if there were any.

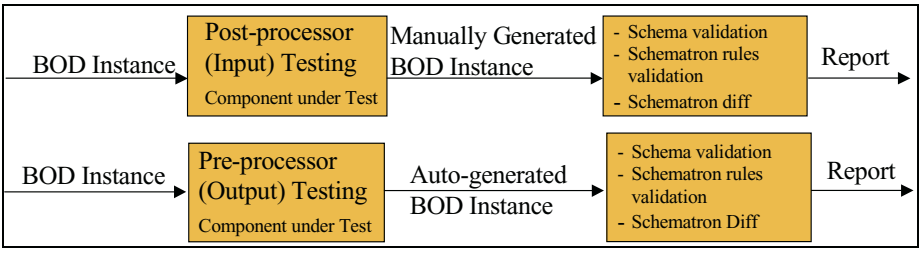

Fig. 8. Summary of the approach for BOD mapping test

Table 3. A list of fields in the SQOH with potential mapping problems as indicated by the information-mapping test

Field	Description	Test value	Return value
CreationDateTime	BOD timestamp	An arbitrary date time - 2003-11-10T11:30:47-08:00	Current date time - 2004-01-28T14:36:02-08:00
BODId	BOD unique Id in one year	An arbitrary string - 200311101130QOH442	An arbitrary string - 637a732d6c7415ee671:fa5e9fe859:-8000
LastShipment Info/ShipToParty	Location code of item's last shipment destination	DUNS number of customer plant – 832022258	DUNS number of supplier plant – 732022158
LastShipment Info/ShipFromParty	Location code of item's last shipment origin	DUNS number of supplier plant – 732022158	DUNS number of customer (OEM) – 132022257
LastShipment Info/ReceivedDateTime	Date and Time of last shipment received by Customer.	An arbitrary date time before the current date - 2004-03-02T09:30:00-05:00	The date portion of the test value without time information - 2004-03-02T00:00:00-08:00
Inventory/ LastModification-DateTime	The last time the Inventory was changed via a (Shipment) BOD coming into the sending system or an event happened within the sending system (e.g., part consumption).	An arbitrary date time before the current date - 2004-02-28T12:00:00-05:00	Current date time - 2004-04-28T15:46:11-08:00.
Uom (unit of measure)	Quantity unit of measure	Each	An abbreviated form – ea

6.2 Results from the Output Test

In the Output Test, similar flags were raised with *CreationDateTime*, *BODId*, *ReceivedDateTime*, and *LastModificationDateTime*. The additional observations led to potential problems: (1) only one line item was returned when two were submitted; (2) the *LastShipmentInfo/ShipFromParty* and *LastShipmentInfo/ShipToParty* were missing; and (3) the field *Inventory/InTransitQuantity* contained value zero although it was not specified in the input. In the first case, we discovered that the test application did not allow inventory information (for a given item and inventory site) to be associated with more than one supplier. In the second case, it seems that a similar problem occurred in the Input Test, which could explain the observed mapping problem: If a field *A* was mapped (incorrectly) to *B* and vice versa in the first place and if this mapping were fixed (after running the input test) so that now *A* correctly maps to *C*, then the reverse output test may not have had a value in the field B to generate any output, as observed. This demonstrates the improvement of the refined reflexive testing approach as described earlier in Fig. 6.

In the third case, the *InTransitQuantity* means the inventory quantity being transported to, but not yet received by, the customer at the time of issuing the SQOH message. The difference between the data not being specified and the data using a default, not-agreed-upon value can result in a different interpretation. When the data is not specified, it means that the field is not used between the customer and the supplier. However, the supplier may use that particular field with another customer. Showing or generating a field with a default value (e.g., zero) when it is not actually in use

could result in an incorrect decision made by the supplier (e.g., supplier repeating the part shipment assuming that it has not shipped the part yet). To avoid these problems, we recommended to the tool vendors that a null field should not be generated or displayed to the user. The tool vendors have agreed that this is an issue which requires attention.

Table 4. Comments and resolution to fields with potential mapping problems

Field	Comment and resolution
CreationDateTime	The integration scenario involves federation of business data exchanged among tools used by customers and suppliers. It might be necessary for the traceability purpose that the BOD CreationDateTime remains the same from customer to suppliers. However, the tool under test generates a new timestamp for every new BOD. A group of IV business process experts indicated that this is not an issue because the scenario involves continuous updates and traceability is not needed.
BODId	The BODId holds similar potential issue and resolution to the CreationDateTime.
LastShipment Info/ShipToParty	The mismatches of these two fields appear to result from incorrect mapping. The use of customer (OEM) identifier instead of the plant identifier is an incorrect mapping. Consequently, the engineer discovered that the incorrect placements of the customer identifier into the ShipFromParty field and of the supplier identifier into the ShipToParty field are mapping errors.
LastShipment Info/ShipFromParty	
LastShipment Info/ReceivedDateTime	The tool stores and/or retrieves only date portion of the input. This is discovered to be the tool implementation problem.
Inventory/ LastModificationDateTime	At the first pass, the tool did not generate this field. In the second pass, the tool interprets and generates this field as the current date time. Both passes indicate that the information mapping is incorrect.
Uom (unit of measurement)	This error indicates the representation mismatch in the unit of measure. Typically, this field should be based on a standard. However, the business experts have indicated that in this scenario, the IV tool should generate the Uom with the same representation as it receives from the customer.

7 Conclusion

Four types of content-level conformance tests have been identified and described. All of them can affect interoperability positively at the application level. Of the four, we discussed in detail various approaches to information-mapping conformance only. Although the logical consistency approach has attractive capabilities, it could be expensive and it has implementation limitations. The human-computer interaction approach is less expensive, but it does not guarantee absolute conformance. It helps reduce the test data generation effort to only include the target standard representation. Currently, we are studying how an ontological approach could address the deficiency in the Schematron diff tool when measuring the equivalence between the test input and the corresponding test output.

We also described information-mapping test-case generation in detail. The current approach relies on filling out a business case and test requirements spreadsheet manually. In the future, portions of this process will be automated as certain assumptions for XML schema design are enforced. In the present approach, we introduced the conditionally optional concept as distinct from the purely optional field. The result is

a reduction in the number of tests as the optional fields need not be permutated. In addition, if the conditions are formally expressed, they can be used to automatically enumerate the test requirements. It should be noted that the proposed test cases and testing technique are based only on positive cases. We envision that the data validation testing techniques [15] might be useful for information-mapping test using negative test cases. The approach would rely on the implication that if the application correctly identifies an error, then it is likely that it has correctly mapped/interpreted the fields. However, there are subtle issues that require further studies and experimentation. An apparent issue is that the applicability of the test may be application specific. That is, some error conditions are not discovered by the application but by a middleware component such as a generic schema-based data parser. Another issue is that the content standards are created for flexible usage, with only a small number of usage conditions specified. In addition, these usage conditions could be application specific. In such situations, the only useful conditions could be the common business rules used in the transaction-behavior test and the scenario-based test.

Finally, we discussed experimental results of a mapping-conformance test using the human-computer interaction approach to verify the conformance of IV tool implementations with IV&I SQOH BOD specification. We witnessed a strong need for repeated cycles of testing whenever the vendors updated their tools in response to new conformance requirements or bug fixes. The feedbacks received from the IV tool vendors indicate significant benefits from the conformance testing runs, which identified a number of problems and inconsistencies. Currently, we are planning to run experiments to analyze benefits of performing the content-level conformance tests as a prerequisite for interoperability testing and system deployment.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

References

1. Automotive Industry Action Group Web Site, accessed December 2004. Available online via <http://www.aiag.org>
2. The Manufacturing Business-to-Business Interoperability Testbed Web Site, accessed December 2004. Available online via <http://www.mel.nist.gov/msid/b2btestbed/>
3. The Open Application Group: Open Application Group Integration Specification version 8.0 (2002). Available online via <http://www.openapplications.org/downloads>
4. World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Third Edition) W3C Recommendation (February 2004).
5. Morris, K.C., Flater, D. (September 1999): Standards-based Software Testing in a Net-Centric World. Proceedings of Ninth International Workshop on Software Technology and Engineering Practice, Computer Society, Pittsburgh, PA 115-122

6. Kemmerer, S.J. (July 1999). STEP the Grand Experience, National Institute of Standards and Technology Special Publication 939.
7. Morris, K.C., Mitchell, M.J., Barnard, A. (May 1993): Validating STEP Application Models at the National PDES Testbed.
8. Ivezic, N., Kulvatunyou, B.S., Jones, A.T., Frechette, S., Cho, H., and Jeong, B. (October 2004): An Interoperability Testing Study: Automotive Inventory Visibility and Interoperability. Fourteenth E-Challenge Conference, Vienna, Austria, 551-558.
9. World Wide Web Consortium: XML PATH Language Version 1.0 (November 1999). Available online via <http://www.w3.org/TR/xpath>
10. Altova GmbH: XML Spy 2004 Professional Edition
11. Jelliffe, R.: The Schematron Assertion Language 1.5. Academia Sinica Computing Center (2000). Available online via
12. Lindsey, B.: XT version 20020426a, Extensible Stylesheet Transformation Implementation in Java (2002). Available online via <http://www.blnz.com/xt/index.html>
13. Elmendorf, W.R. (1973) : Cause-Effect Graph in Functonal Testing, TR-00.2487. IBM Systems Development Division, Poughkeepsie, NY.
14. Kulvatunyou, B.S., Ivezic, N., Martin, M.J., Jones, A.T (october 2003) : A Business-to-Business Interoperability Testbed: An Overview. The 5th International Conference on ELECTRONIC COMMERCE (ICEC), Pittsburgh, PA.
15. Beizer, B. (1983): Software Testing Techniques. Van Nostrand Reinhold electrical/computer science and engineering series, NY.
16. Beizer, B. (1990): Software Testing Techniques, 2nd Ed. Van Nostrand Reinhold, NY.
17. Beizer, B. (1995): Black-Box TestingJohm Wiley & Sons, NY.
18. Myers, G.J. (1979): The Art of Software Testing. Wiley Series in Business Data Processing.
19. Anicic, N., Ivezic, N., and Jones, A (Feburary 2005) : An Architecture for Semantic Enterprise Application Integration Standards. First International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland.
20. Accordare Web Site, accessed April 2004. Available at <http://www.accordare.com>