Emerging patterns in the use of XML for information modeling in vertical industries

S. Hinkelman
D. Buddenbaum
L.-J. Zhang

Extensible Markup Language (XML) has emerged as the predominant non-binary information format. The impact of XML has been most strongly felt in information exchange environments and information modeling. This paper focuses on a set of innovative patterns that has emerged in the use of XML for information modeling and business content design in the health-care, travel, insurance, and other industries. We provide historical perspectives on this development and characterize XML's current state in relation to Web Services.

INTRODUCTION

The use of XML (Extensible Markup Language) for information modeling within vertical industries has taken many diverse forms. Some, but not all, of these forms have been influenced by the emerging service-oriented architecture (SOA) XML infrastructures. Despite the diversity of approaches taken by industry-level consortiums working with XML, there is a great deal of commonality, as exemplified by four basic patterns for XML business content design which have recently emerged within vertical industry consortiums. These patterns are (1) Business Content Envelope, (2) Web-Services-Based Infrastructure, (3) Wrapped Content, and (4) Top-Down Modeling. This set of patterns, though limited, provides a framework that can aid Web Services adoption efforts by industry standards organizations.

In this paper, we begin with a review of the history of the development of a selection of XML standards. Next, we focus on the emergence of the aforementioned industry-level patterns in XML business content design and describe these patterns in detail. We then describe the associated effects and implication of mappings (i.e., "bindings") of these patterns to a Web Services infrastructure.

DEVELOPMENT OF XML STANDARDS

XML was originally designed for large-scale electronic publishing applications but has grown to handle the exchange of information in a variety of contexts. The fundamental standards activity for XML was conducted by the World Wide Web Consortium[1] (W3C**), beginning with the Extensible Markup Language[2] (XML) Version 1.0 W3C

Recommendation in February 1998. W3C continues to coordinate XML standards activity with working groups in such areas as XML Query, XML Schema, XML Core, and XML Processing Model.

Industry consortiums developing standards for XML information exchange have emerged over the last several years due to the explosive growth of XML and its ability to define structured vocabularies. Some of these organizations and their efforts are described in this paper. A number of these organizations have explicitly defined themselves as using today's commercial Web Services infrastructure for information exchange.

The Open Application Group, Incorporated[3] (OAGi) manages the Open Application Group Integration Specifications (OAGIS**) standard, which defines a business information envelope along with a set of business-information content types. The Association for Cooperative Operations Research and Development[4] (ACORD), the leader in global insurance standards, has developed messaging standards for Life and Annuity (L&A), Property and Casualty (P&C), and reinsurance products for the insurance industry. Like OAGi, it is a member-driven organization whose efforts began before Web Services were defined and widely accepted, and its mission includes staying current with infrastructure technologies such as Web Services. Remaining infrastructure-neutral like OAGi, ACORD has developed core specifications and complementary Framework Implementation Guides. As stated in the *ACORD Messaging Service XML Specification and SOAP Implementation Guide*, "The design of the ACORD core payload standards will not prohibit or intentionally favor use of any framework standards that are specified by cross-industry bodies. On the reverse, ACORD Framework Implementation Guides can be viewed as cross-industry standard profiles to support the insurance business processes in the most adequate way."[5] Like OAGi's approach, the ACORD approach is not to design Web services directly as a dependency, but to map onto Web Services capabilities. The OpenTravel Alliance** (OTA) serves a similar function for the travel industry. The standardization activities of OAGi, ACORD, and OTA are described in detail in this paper, along with those of some other companies.

As is the case in many fields, the nature of the organizations managing the development of standards has a great impact on the type and qualities of standards that are produced. In the following subsections, we list some of the organizational characteristics relevant in this context for XML standards.

Legacy-based vs "green field" organizations

Some organizations have a long history of standards work in information exchange, possibly going back to the days of Electronic Data Interchange (EDI). Their activity reflects this history, and tremendous effort tends to be spent on managing transitions between technical implementations, even to the degree of ensuring some level of backward and forward compatibility. These efforts often result in a suboptimal implementation. Other organizations lead "green field" standards efforts (i.e., those without a history), enjoying much more freedom to adopt current techniques, based on the "best of breed" thinking at the time that the development is taking place. For example, a "session" construct may be a necessary element for supporting reliable or sequenced messaging. A green field approach would typically rely on a recent horizontal (i.e., cross-industry) standard such as Web Services Reliable Messaging[6] (WS-RM), to provide this support; a more seasoned standard would probably use a legacy mechanism modeled within its architecture. In the latter case, for reasons of continuity (because approaches may be short-lived), ease of adoption, and consistency with production applications, the legacy approach may be perpetuated at the expense of, or in addition to, the application of a recently emerged standard, such as WS-RM, that provides the same function.

Comprehensive vs streamlined organizations

Some organizations mandate the development of solutions across the entire "eco-system" of implementations. This results in a more complete view of the scope of the problem, with generally wider acceptance, but with an increased burden of consensus building and use of existing production standards. This also results in a potentially more conservative approach to cross-industry standards adoption. Other efforts are managed by streamlined consortiums of like-minded organizations looking to accelerate development and adoption of a particular standard within a smaller scope of use. In this case, the effort is associated with solving a key aspect of a problem which the consortium has deep knowledge of and wishes to see addressed.

Information exchange vs service optimization focus

The organization's charter puts limits on the scope of the issues that the organization can and will address. Some standards organizations are developed to solve information sharing problems. In this case, the modeled content can be more document-oriented in an attempt to synchronize the data held by the various members of the organization, such as a standard for sharing customer data. Other standards organizations are concerned with service optimization. In this case, the modeled content is designed to efficiently provide a service, such as payments over a banking network. Still other standards organizations are more concerned about process optimization and take a more service-oriented approach aimed at bringing a process through various states to its conclusion, for example, from inception through completion of a purchase order.

Structure vs process focus

The maturity of the standards produced by an industry standards organization can be based on the degree to which it standardizes not only structure and syntax but also processes. Standardized processes provide a standardized context for the use of message formats derived from the structure and syntax of the business information model. This provides impetus for the standardized design of functions such as security, reliability, and routing. Specifying information interaction patterns among relevant technologies increases the value of the business content specification by facilitating adoption. Industry standards organizations wish to provide bindings to relevant technologies for this reason. Dominant business players tend to simplify interaction requirements, limiting them to the set of patterns and technologies that meet their business needs; industries with a more heterogeneous set of partners usually have a more complex set of required interaction patterns.

EMERGING BUSINESS CONTENT DESIGN PATTERNS

Clearly, there is no "one size fits all" approach for best practice implementations adopting Web Services in all of the vertical-industry standards organizations. Instead, there are many diverse approaches. While these divergences are significant, a basic set of patterns has emerged in recent years. This set of patterns is useful in understanding and aiding Web Services adoption efforts by industry standards organizations. The patterns simplify understanding the impact of each approach and help identify requirements that can increase the rate and sophistication of Web Services adoption.

In the following subsections, we explore these basic patterns for business content design: (1) Business Content Envelope, (2) Web Services-Based Infrastructure, (3) Wrapped Content, and (4) Top-Down Modeling. We discuss the differences between these patterns and describe some implementations that use them.

Business Content Envelope pattern

In order to endure and have a high level of insulation from change in underlying infrastructure technology, an industry-level standard is required to manage (at least) the basic specification of interaction and process indicators in a way that is consistent with, and has some level of integration with, the business content itself. A comprehensive pattern integrating business information with interaction and process indicators is referred to as a "Business Content Envelope" pattern. (An XML envelope is an XML document type that is defined to be a holder for other arbitrary XML data.) While not new, this pattern has proven to be successful and is currently used within various organizations. This pattern is based on the fundamental principles of abstraction of infrastructure and minimizing dependencies on any given information-exchange infrastructure.

The OAGIS standard uses this pattern and abides by some foundational infrastructure-neutral principles. OAGIS defines a Business Object Document (BOD), a comprehensive Business Content Envelope, along with business content. In the OAGIS envelope architecture the business content, referred to as a *noun*, is associated with multiple actions, which are referred to as *verbs*. Extensibility is also supported as a core part of the architecture. the OAGIS Business Content Envelope is by principle and by design independent of any specific lower-level protocol or transport since its inception. ACORD has also developed a Business Content Envelope.

In the following subsections, we describe in detail the approaches used by ACORD and OAGi in developing their Business Content Envelope patterns.

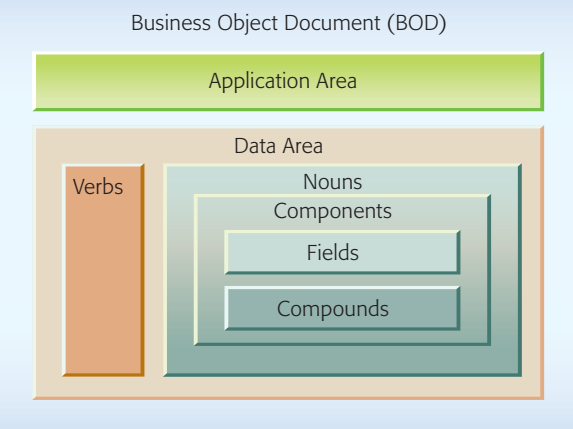

Figure 1
OAGIS BOD architecture

The OAGIS envelope

Figure 1 shows the overall architecture of the OAGIS envelope.[7] OAGIS schemas specify a naming convention for BODs, consisting of verbs and nouns such as `ProcessPurchaseOrder`. The outer layers of the BOD envelope identify the intentionality (i.e., what the message is intended to do, a verb), business content (a noun), version identifier of the document, release number of OAGIS, and a test/production flag, for example:

```
<ProcessPurchaseOrder...versionID="..."
  releaseID="..."
  systemEnvironmentCode="Production">.
```

An Application Area contains application-specific information common to all BODs, such as:

```
<ApplicationArea>
   <Sender>
     <LogicalID> ... </LogicalID>
     <ComponentID> ... </ComponentID>
     <TaskID> ... </TaskID>
     <ReferenceID> ... </ReferenceID>
     <ConfirmationCode> ...
     </ConfirmationCode>
     <AuthorizationID> ... </AuthorizationID>
   </Sender>
   <CreationDateTime> ... </CreationDateTime>
   <Signature ... />
   <BODID ... />
   <UserArea ... />
</ApplicationArea>
```

A Data Area carries the business content, which is the information that is specific to each BOD envelope, as shown in *Figure 2*. OAGIS further defines the business content in a hierarchy of elements, called components, fields, and so forth.

The Acord envelope

ACORD designed a messaging service, the Acord Messaging Service Version 1.2, which wraps each of the insurance industry XML standards (P&C, L&A, and reinsurance) based on a set of horizontal standards as shown in *Figure 3*. The purpose of Acord Messaging Service Version 1.2 is to support the transport, routing, content, and security requirements of the L&A, P&C, and reinsurance standard specifications. ACORD's stated position is to create similar implementation guides as technologies evolve. The approach taken for Web Services enablement is to create an Acord Messaging Service Version 1.2 that provides an envelope which supports specific requirements from multiple standard specifications in a technology-neutral way, including message management (requiring a unique identifier, specific message type, status, and message signature for nonrepudiation), routing (requiring sender/receiver, time stamp, and intended application), packaging, and security.

ACORD is currently developing the next version of the Acord Messaging Service Version 1.2 specification, which will be called the ACORD Web Services Profile. The profile will still be based on the Business Content Envelope pattern, but will differ from today's specification by the design of the SOAP and WSDL bindings. The goal is to align more closely with SOA principles and enable business services to be exposed by WSDL. This will be enabled by profiling the latest development of Web Service Standards (SOAP 1.2, WSDL 2.0, WS-addressing and WS-Reliable Messaging) and promoting harmonized service wrapper components within the business payloads in each of the three industry standards.

As an example, Figure 3 contains excerpts from the Acord Messaging Service Version 1.2 Business Content Envelope pattern of an ACORD message within a Simple Object Access Protocol (SOAP) body.[8]

Web Services-Based Infrastructure pattern

This pattern includes those industry-level patterns whose infrastructure is based exclusively on Web Services. Although consortiums following the Web Services-Based Infrastructure pattern use Web

```
<DataArea>
  <Process acknowledgeCode="Always">
    <ActionCriteria>
      <ActionExpression expressionLanguage="token" actionCode="Add">token</ActionExpression>
      <ChangeStatus>
       ...
      <EffectiveDateTime>...</EffectiveDateTime>
      <ReasonCode>...</ReasonCode>
       ...
        <StateChange>
           <FromStateCode>...</FromStateCode>
           <ToStateCode>...</ToStateCode>
           <ChangeDateTime>...</ChangeDateTime>
           <UserArea/>
        </StateChange>
      </ChangeStatus>
    </ActionCriteria>
  </Process>
  <PurchaseOrder>...</PurchaseOrder>
</DataArea>
```

Figure 2
Example of a data area

Services exclusively for their infrastructure, the business content design developed by these consortiums is typically developed without concern for the low-level infrastructure details of Web Services.

Unlike the Business Content Envelope pattern, this pattern defines "usage-context-free" instance documents (i.e., those containing no verbs, interaction indicators, process indicators, etc.), which can be referenced under several general operations that are defined in the infrastructure. This is in sharp contrast to a Business Content Envelope pattern, which contains much more than the business content. This content design pattern requires conforming instance documents to have a single specific root element.

The architecture of MedBiquitous.org,[9] a distinguished medical professional organization, was defined by reference to the Web Services-Based Infrastructure pattern. MedBiquitous.org explicitly and solely relies on the Web Services infrastructure to provide all interaction and process specifications for exchanging professional business content. A natural and complementary content design authoring pattern that fits well with this pattern, known in industry by some as the "venetian blind schema" pattern, is shown in the following example:

```
<xs:schema...
  <xs:complexType name="InternalElementType">
    <xs:sequence>
      <xs:element name="A"/>
    </xs:sequence>
   </xs:complexType>
   <xs:element name="RootElement"
                  type="RootElementType"/>
  <xs:complexType name="RootElementType">
    <xs:sequence>
      <xs:element name="InternalElement"
                   type="InternalElementType"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Binding conventions typically use a wrapped model for business content in which the business content schemas are wrapped with another schema that defines services operations. Within MedBiquitous .org, a wrapped Doc/Lit message style (i.e., one in which information is exchanged in a "raw" form, without encoding or other alteration) is used within the infrastructure layers specific to Web Services. The business content payloads are wrapped with request/response operation wrapper elements, which are defined in a separate schema. No first-class business content (i.e., actual business information, as opposed to associated information or metadata) is defined as part of these wrappers.[10] This marks the initial separation point between business content operations and those that are required for interactions by means of the Web

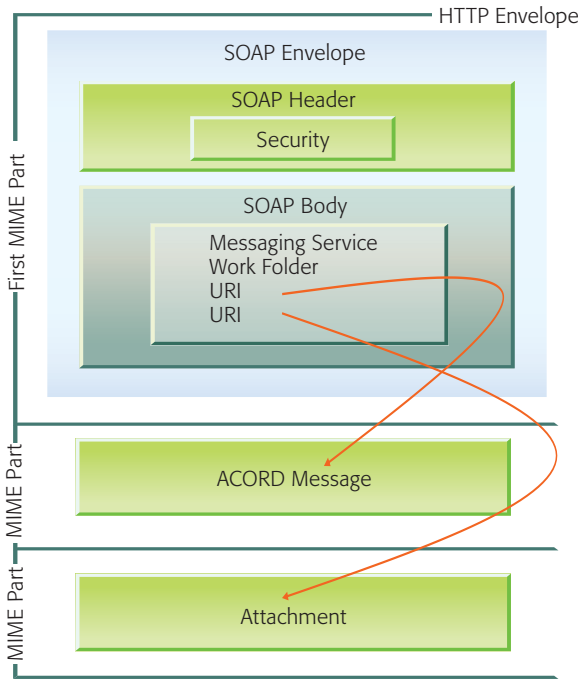

Figure 3
ACORD Messaging Service Version 1.2 architecture

Services infrastructure. The wrapper schemas, one per Web service, consist of the request and response elements for all operations defined within the service.

WSDL files import this schema. WSDL (e.g., the `<types>` structure) is *not* used directly for business content, in order to keep all business type information independent of the Web Services interface specification documents. This separation helps facilitate business payload development by shielding the domain experts from the details of the underlying Web Services technology. In the case of MedBiquitous.org, the actual payload schemas are indirectly imported into WSDL files through the wrapper schemas within the WSDL `<types>` structure, as in the following example:

```
<types>
   <xs:schema...
      (include declaration of the namespace wrapper)
      <xs:include schemaLocation=
         (location of the wrapper schema) />
   </xs:schema>
</types>
```

Business content is not authored within the WSDL files.

The imported wrappers then are used to define the request and response message parts. Naming conventions using WSDL operation names determine the message names, as shown in the following example:

```
<message name="Operation1InputMessage">
   <part name="Wrapper" element="operation1"/>
</message>
<message name="Operation1OutputMessage">
   <part name="Wrapper" element="operation1Response"/>
</message>
```

This design uses an approach in which generic interface operations are used across the entire multiusage payload. This shields the interface from change over time, but necessitates constraint checking within the business logic.

In the Web Service-Based Infrastructure pattern, the intentionality of the message is completely determined by the choice of WSDL operation names, unlike the Business Content Envelope pattern, where intentionality indicators, such as verbs or actions, typically exist as design elements integrated with the business content architecture. This is not to say that the business content design is completely defined without any idea or context of how the information is going to be used, but a Web Services-Based Infrastructure pattern formally makes the intentionality of the messages visible only at the Web Services infrastructure layer.

A complementary aspect to the usage-context-free business information modeling of this pattern lies in how it specifies infrastructure operation granularity. In order to avoid fragile Web service interfaces, general operations on service interfaces are used rather than highly specific operations. An example of this would be the use of the well-known Create/Read/Update/Delete (CRUD) operations rather than operations highly specific to the business content such as `CreateCardiologist`. Another complementary aspect is payload or content design authoring specifying a highly type-based document with a single root element, as discussed previously. The root element can be referenced by several general operations, as seen here:

```
<operation name="Create">
   <input message="Operation1InputMessage"/>
   <output message="Operation1OutputMessage"/>
```

```
  <fault...
</operation>
```

Like almost all industry groups addressing Web Services, interoperability is a key concern[11] for MedBiquitous.org. The WS-I basic profile provides a foundation for interoperability guidelines, but part of this profile is a clarification of Web Services specifications targeted for Web Services infrastructure developers and is not directly relevant to vertical-industry standards organizations. In MedBiquitous.org, areas such as common faults for all services operations are defined to supplement the basic profile in the context of the Web Services infrastructure and increase interoperability.

A ramification of general purpose operations typical of the Web Services-Based Infrastructure pattern as described here is that Web Services faults that are specific to the business content are typically not defined—hence, the need for general error definitions, such as those defined in MedBiquitous.org. One such error is a 'business rule' fault to accommodate an error associated with the business content, which is specified for every Web Services operation, and can be specified with the syntax `<fault name="BusinessRuleFault" message=../>`.

From an overall organizational and architectural view, the Web Services-Based Infrastructure pattern is arguably more efficient than the Business Content Envelope pattern, positioning an organization to take full advantage of the present and future functions of the Web Services infrastructure. Relying solely on the Web Services infrastructure eliminates concerns about infrastructure abstractions in higher-level designs. However, defining the infrastructure in a concrete way specific to a given infrastructure technology like Web Services means that when an alternative infrastructure emerges, significant wholesale replacement must be defined at the lower levels as there are no integrated interaction abstractions and process indications at the upper layers.

Wrapped Content pattern

The Wrapped Content pattern, similar to the comprehensive Business Content Envelope pattern, allows an organization to remain independent of a given message exchange infrastructure; it also requires some level of definition for interaction and process indication. The Wrapped Content pattern is limited and is based on wrapping primary business content with a single style of interaction such as request/response. There is no overall design of sections to contain information for specific purposes. In contrast, a Business Content Envelope pattern is robust and takes on many abstractions including processing actions (verbs), acknowledgements and confirmations, and an overall compartmentalized extensible stucture for containment of different information used for different purposes.

Some industry organizations, in addition to defining reusable business content, define a common set of reusable process indicators that are integrated with interaction exchange structures used as wrappers for the industry-specific business content. These process indicators are similar to the metadata defined in full Business Content Envelope patterns. In its most basic form, the business information is contained within an exchange structure such as a message request and a message response. ACORD, within its overall design and methodology, defines request/response interaction exchange structures.

The OpenTravel Alliance[12] (OTA), a well-respected pioneering organization for the travel industry, also uses this design approach. OTA has standardized a set of common attributes and indicators that may appear on the request/response interaction-exchange-structure root element for all OTA message payloads.[13]

OpenTravel Alliance and the Wrapped Content pattern

OTA predates current distributed infrastructures such as Web Services infrastructures. To illustrate the use of process indicators for this pattern, we use examples from OTA's "common types" schema and define partial example instances. The travel industry specifies many industry-specific reusable types within this schema, such as `LoyalLevel` and `HotelReference`, by using process indicators. OTA provides several indicators within its specifications to accommodate usages that are current in this industry.

An indicator may be specified when a requesting host indicates that the receiving host should include an 'echo token' of the same value in the response. Process indicators may be used to indicate the processing model (test or production) of the

```
<OTA_AirAvailRQ
     xmlns=http://www.opentravel.org/OTA/2003/05
     xmlns:xsi=
          "http://www.w3.org/2001/XMLSchema-instance"
     EchoToken="12345"
     TimeStamp="2003-07-17T09:30:47-05:00"
     Target="Production"
     Version="2.001"
     SequenceNmbr="1"
     PrimaryLangID="en-us"
     MaxResponses="10"
     DirectFlightsOnly="false"
     TransactionStatusCode="Continuation"
     TransactionIdentifier="224">
       <POS>
         <Source AgentSine="BSIA1234PM"
          PseudoCityCode="2U8"
          ISOCountry="US"
          ISOCurrency="USD">
          <RequestorID URL=
          "http://www.//provider1.org" Type="5" ID="123"/>
         </Source>
       </POS>
       <OriginDestinationInformation>
         <DepartureDateTime>2003-08-13</DepartureDateTime>
         <OriginLocation LocationCode="LHR"/>
         <DestinationLocation LocationCode="LAX"/>
       </OriginDestinationInformation>
       <TravelPreferences...
</OTA_AirAvailRQ>
```

Figure 4
Example of OTA Air Availability request instance

receiving node. The version of the message may also be indicated.

A unique identifier may indicate that all messages sent in a set of request and response messages are part of a single ongoing transaction, and a message sequence number can be used to identify the number of the transaction as assigned by the sending system. This numbering allows an application to process messages in a certain order or to request a resynchronization of messages in the event that a system has been offline and needs to retrieve messages that were missed.

A transaction status code may be defined to indicate where a specific message lies within a sequence of messages. The code may take the values `Start`, `End`, `Rollback`, `InSeries`, and `Continuation`. A process indicator may be defined to indicate the desired version of the payload response message. This requirement for specifying one of several non-error responses may provide a challenge when mappings to Web Services interfaces and operations are defined.

OTA's 2005A Air Availability schemas provide a comprehensive view of business content wrapped within an interaction exchange structure along with associated process indicators. These schemas specify the availability of flights between a pair of cities on a specific date for a specific type and number of passengers, as in ***Figure 4***. Many of the previous indicators are exemplified here within this Wrapped Content message.

OTA has not published formal binding specifications for a Web Services infrastructure. At present, only information for the June 2005 ebXML Message Service specification[14] (ebMS) is documented. Definition of the mapping to alternative infrastructure technologies is currently under development. However, OTA's robust interaction Wrapped Content pattern along with its process indicators would provide a comprehensive source for successful mapping to the evolving Web Services infrastructure with its increasing transactional and state-full capabilities, considering OTA's already defined transaction status codes and identifiers.

ACORD and Wrapped Content

ACORD XML for L&A products is based on the ACORD Life Data Model and provides a robust, industry-tested XML vocabulary. ACORD P&C is similar in many respects. An ACORD L&A XML document is built around a request/response model. The processing mode can be either synchronous or asynchronous (the response may include a "notify" statement as a trigger to the receiver that additional interaction is possible). The processing model is based upon a pair of messages (request and response) and uses the framework shown next:

```
<xsd:complexType name="TXLife_Type">
  <xschoice>
    <xsd:sequence>
      <xsd:element ref="UserAuthRequest"
                              minOccurs="0" />
      <xsd:element ref="TXLifeRequest"
                              minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:sequence>
      <xsd:element ref="UserAuthResponse"
                             minOccurs="0" />
      <xsd:element ref="TXLifeResponse"
                              minOccurs="0"
                   maxOccurs="unbounded" />
      <xsd:element ref="TXLifeNotify"
                              minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:choice>
  <xsd:attribute name="Version" type="xsd:string" />
</xsd:complexType>
```

Requests can be submitted together within a single file or envelope as a single stream, and in this case will receive a single response stream although transaction order is not maintained. Not all responses can be generated in real time; if they cannot, a notification response message is used to alert client applications to the processing of an outstanding transaction. Notification is continuously sent with other transaction response messages by a server to a client application while the response remains outstanding.

Some examples of ACORD XML for L&A processing indicators include requests for a response correlation identifier (used to determine the matching response), requests for a response indicator (used to indicate if the requestor wants notification of outstanding delayed responses), and requests for a transmission mode indicator (to indicate how a request should be treated in the context of other requests, such as a request that is an update or a replacement of another request).

Other processing indicators define the transaction to be processed or allow a requestor to dictate the nature of the results returned by the service provider, such as the data scope to return (Object, Object and related Objects, etc.), the start record for a search, or the maximum number of records to return. Still other indicators define how to specify the relationships between the message and multiple possible attachments.

Top-Down Modeling pattern

Formal modeling patterns, in contrast with ad hoc development, are used by some organizations for developing standard messages. Such formal Top-Down Modeling patterns are emerging within vertical-industry organizations. In these patterns, message specifications are developed with increased rigor, based on some form of information model. The technology, methodologies, and tools so developed may be accompanied by training classes to ensure consistency. In such environments, a large portion of the effort is spent on defining requirements, use cases and roles, and the information model. A Unified Modeling Language** (UML**) profile is often used.

This pattern provides the opportunity for standardization of elements that may be more difficult to standardize in less formal environments. A rigorous methodology can facilitate the specification of the usage of the information, and this has ramifications on the inclusion of the information elements, their cardinality, and even their semantics. As a natural consequence of this rigor, library and registry considerations arise that play a key role in the assembly of information and the definition of usage contexts.

Health Level 7 and the Top-Down Modeling pattern

One of several organizations employing such a Top-Down Modeling pattern is the advanced Health Level 7[15] (HL7) organization, a health-care information standard in which, increasingly, XML is viewed primarily as an encoding technology rather than a source information model. This pattern holds much promise for increasing the precision of standards required to promote interoperability

between businesses, reducing ambiguity and leading to reduced complexity and cost.

Version 3.0 of the HL7 standard represents an evolution in several ways. While HL7 messages have moved to XML encoding as an "implementable technology specification" (ITS), this version of HL7 introduces the strategic Reference Information Model (RIM), the Message Development Framework methodology, and the accompanying UML profile containing hierarchical message definitions (HMDs).

These features, along with HL7's Model Interchange Format (MIF), have resulted in the generation of the implementation layer (such as XML encoding) through tooling. HL7's MIF is a set of related schemas that define the set of primary artifacts that may be developed or exchanged as part of this standard. These artifacts provide a common exchange format for use between tools and repositories. Although this methodology is central to HL7, this standard continues to evolve, and its components are not fully integrated, requiring further testing.

The HL7 Clinical Document Architecture (CDA) is a document markup standard (currently in production) that specifies the structure and semantics of clinical documents for the purpose of information exchange. The CDA document source is currently encoded in XML with "derived meaning" from the HL7 RIM. It is intended that if and when alternate implementations are feasible, future technology encodings will not be limited to XML.

Information modeling through message structure definition

Organizations like HL7 employing a Top-Down Modeling pattern (using some form of information domain model) typically specify classes of information required and the properties of those classes, including attributes, relationships, and so forth, by use of a UML profile. HL7 Version 3.0 uses data type specifications, vocabulary specifications, and a RIM to derive technology-level message specifications.

The intention of HL7's MIF is to specify an XML Schema representation consistent with its UML profile, comprised of health-system-wide information structures from which ITSes are derived to define lower-level implementation technology. HL7 defines a suite of implementation tools supporting its Top-Down Modeling pattern, which includes repository interface tools, terminology table tools, modeling and message tools, schema transformation tools, and so forth.

Schema generation uses serialized MIF models. Though the MIF is still evolving, it has already had a significant impact on the direction and development of HL7 tools and plays a central role in schema generation.

WEB SERVICES BINDINGS

In the following three subsections, we describe the approach taken for Web Services bindings by the OAGi, ACORD, and HL7 organizations.

OAGIS Web Services binding

The OAGIS BOD architecture predates Web Services specifications and is independent of any communication mechanism. It can be used with simple transport protocols such as HyperText Transfer Protocol (HTTP) and SMTP (Simple Mail Transfer Protocol), but it also can be used with more complex transport protocols such as SOAP, ebMS, or any other EAI (enterprise application integration) system.

Mapping the OAGIS BOD architecture to Web Services technology occurs largely at the WSDL abstract layer.[16] WSDL's "binding" layer is accommodated by using naming conventions to define a SOAP binding specifying a WSDL document/literal encoding style, and WSDL's "service" structure is also accommodated by naming conventions. All OAGIS mappings to Web Services use a request/response exchange pattern and define mappings for an "asynchronous push" model and a "synchronous" model.[17]

The WSDL <types> structure imports all of the envelope schemas for given business content. Business content is not authored inside the WSDL <types> structure, as shown in the following example:

```
<types>
    <xs:schema elementFormDefault="qualified"
      targetNamespace=
      "http://www.openapplications.org/oagis/8.0/ws">
       <xs:include schemaLocation=
       ".../xsd/MessageResponse.xsd" />
       <xs:include schemaLocation=
       ".../.../OAGIS/BODs/AddPurchaseOrder.xsd"/>
```

```
        ...
    </xs:schema>
</types>
```

The name of every root envelope BOD element defines a single WSDL <message> structure with a single <part>, which references the top-level element, as in the following example:

```
<message name="AddPurchaseOrder">
   <part name="Message" element="oa:AddPurchaseOrder"/>
</message>
<message name="CancelPurchaseOrder">
   <part name="Message"
         element="oa:CancelPurchaseOrder"/>
</message>
<message name="ChangePurchaseOrder">
...
```

The naming convention for a portType dictates the use of the noun name, preceded by Request, Response, or Sync, and followed by PortType. Within the portType, for asynchronous push requests and synchronous request/responses, WSDL operations are defined and named by using envelope root-element names typical of the seller side of an interaction for the given business content, as in the following example:

```
<portType name="RequestPurchaseOrderPortType">
   <operation name="AddPurchaseOrder">
      <input message="oagws:AddPurchaseOrder"/>
   </operation>
   <operation name="CancelPurchaseOrder">
      <input message="oagws:CancelPurchaseOrder"/>
   </operation>
  ...
</portType>
```

Within the portType, WSDL operations are defined by using the envelope root element names typical of the buyer side of an interaction for the given business content, as in the following example:

```
<portType name="ResponsePurchaseOrderPortType">
   <operation name="ShowPurchaseOrder">
      <input message="oaws:ShowPurchaseOrder"/>
   </operation>
   <operation name="ListPurchaseOrder">
      <input message="oaws:ListPurchaseOrder"/>
   </operation>
   <operation name="ConfirmBOD">
  ...
</portType>
```

OAGIS includes process indicators within its BOD architecture, such as a confirmBOD indicator indicating the confirmation of BOD reception, as in the following example:

```
<portType name="SyncPurchaseOrderPortType">
...
   <operation name="CancelPurchaseOrder">
      <input message="oaws:CancelPurchaseOrder"/>
      <output message="oaws:ConfirmBOD"/>
   </operation>
...
   <operation name="GetPurchaseOrder">
      <input message="oaws:GetPurchaseOrder"/>
      <output message="oaws:ShowPurchaseOrder"/>
   </operation>
   <operation name="GetListPurchaseOrder">
      <input message="oaws:GetListPurchaseOrder"/>
      <output message="oaws:ListPurchaseOrder"/>
   </operation>
...
   <operation name="ProcessPurchaseOrder">
      <input message="oaws:ProcessPurchaseOrder"/>
      <output message="oaws:ConfirmBOD"/>
   </operation>
</portType>
```

ACORD Web Services binding

Because ACORD insurance specifications predate Web Services standards, much of their interaction design and supporting process indicators are implemented within the messaging content (see the section "Wrapped Content pattern"). As a result, the ACORD Web Services binding is a mapping exercise intended to address specific requirements, given the capabilities of available horizontal standards.

An example of this is the Acord Messaging Service Version 1.2 mapping based on SOAP 1.1 (document mode), WSDL 1.1, and the WS-I 1.0 basic profile (WSI-BP). When Acord Messaging Service was built, ACORD and its membership evaluated available Web Services capabilities and adoption in the context of insurance-industry requirements with the objective of preserving compatibility with existing insurance-industry messaging standards. As a result, Acord Messaging Service Version 1.2 approached areas of concern such as document and attachment management, routing, session handling, and reliability by designing support for them in the Acord Messaging Service wrapper and interaction design itself. The Acord Messaging Service Version 1.2 approach is to support the breadth of insurance-

```
<wsdl:types>
  <schema
  targetNamespace="http://www.ACORD.org/Standards/AcordMsgSvc/1.1.0"
  xmlns="http://www.w3.org/2001/XMLSchema">

<!--Inbox port messages-->
    <xs:element name="ListInRq" type="xs:anyType"/>
    <xs:element name="ListInRs" type="xs:anyType"/>
    <xs:element name="PostRq" type="xs:anyType"/>
    <xs:element name="PostRs" type="xs:anyType"/>
    <xs:element name="StatusInRq" type="xs:anyType"/>
    <xs:element name="StatusInRs" type="xs:anyType"/>

<!--Outbox port messages-->
    <xs:element name="ListOutRs" type="xs:anyType"/>
    <xs:element name="ListOutRq" type="xs:anyType"/>
     ...
<!--Call port messages-->
    <xs:element name="CallRq" type="xs:anyType"/>
    <xs:element name="CallRs" type="xs:anyType"/>
  </schema>
  <schema targetNamespace=
     "http://schemas.xmlsoap.org/soap/envelope">
     <xs:element name="Fault" type="xs:anyType"/>
  </schema>
</wsdl:types>
```

Figure 5
Generic WSDL: types element overloaded with ACORD message structure

industry standards by providing four generic modes of interaction: one-way business-message push and pull, business-message request/response, and request/response without a business message.

The binding to SOAP is thus designed to support the defined message exchange. The approach to mapping the architecture of the ACORD insurance-industry standard to Web Services technology is evident at the WSDL abstract layer. WSDL's binding layer is accommodated by using conventions to define a SOAP binding specifying a document/literal encoding style; WSDL's service structure is accommodated by a set of predefined functions; and the WSDL types structure is used to name a "generic" element that can be overloaded with any ACORD message structure as required, as shown in *Figure 5*.

Specific element names are provided as interaction proxies and are used as a means of attaching a message specification from one of the standards:

```
<wsdl:message name="PostRequest">
  <wsdl:part element="ac:PostInRq"
     name="PostRqPart"/>
</wsdl:message>
```

The required message interaction patterns are then generically invoked by using conventions for a set of predefined ports as required to satisfy an interaction with a partner. The convention for using the ports is defined in the Acord Messaging Service Version 1.2 specification. The WSDL for the ports supporting ACORD message exchange patterns is shown in *Figure 6*.

The Acord Messaging Service Version 1.2 approach uses a combination of specific message element content and protocol design to provide support for a range of requirements. Acord Messaging Service Version 1.2 and specific protocol instructions within the specification are intended to provide indicators for support of reliable and sequenced interactions, notification capabilities, and indication of "in process" status, as well as standard request/response processing.

The application element contains information indicating the ACORD standard defining the content (L&A, P&C, or reinsurance), and the version of the associated specification that defines the content.

The following code shows how Acord Messaging Service Version 1.2 introduced SOAP body `Sender/`

```
<wsdl:portType name="AcordMsgSvcInbox">

  <wsdl:operation name="Post">
    <wsdl:input message="ac:PostRequest" name="PostRq"/>
    <wsdl:output message="ac:PostResponse" name="PostRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="StatusIn">
    <wsdl:input message="ac:StatusInRequest"
     name="StatusInRq"/>
    <wsdl:output message="ac:StatusInResponse"
     name="StatusInRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="ListIn">
    <wsdl:input message="ac:ListInRequest" name="ListInRq"/>
    <wsdl:output message="ac:ListInResponse"
     name="ListInRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

</wsdl:portType>

<wsdl:portType name="AcordMsgSvcOutbox">

  <wsdl:operation name="Retrieve">
    <wsdl:input message="ac:RetrieveRequest"
     name="RetrieveRq"/>
    <wsdl:output message="ac:RetrieveResponse"
     name="RetrieveRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="StatusOut">
  ...
  </wsdl:operation>

</wsdl:portType>

<wsdl:portType name="AcordMsgSvcCall">

  <wsdl:operation name="Call">
    <wsdl:input message="ac:CallRequest" name="CallRq"/>
    <wsdl:output message="ac:CallResponse" name="CallRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

</wsdl:portType>
```

Figure 6
WSDL for ports supporting ACORD message exchange patterns

`Receiver/MsgId` elements to define the routing context for a message:

```
<ac:PostRq>
  <ac:Sender>
    <ac:PartyId>urn:duns:123456789
    </ac:PartyId>
    <ac:PartyRoleCd>11
    </ac:PartyRoleCd>
  </ac:Sender>
  <ac:Receiver>
    <ac:PartyId>urn:duns:912345678
    </ac:PartyId>
    <ac:PartyRoleCd>87
```

```
      </ac:PartyRoleCd>
   </ac:Receiver>
   <ac:MsgItem>
      <ac:MsgId>01ff00c1-e48e-4cc5-b26c-8fc210
      </ac:MsgId>
      <ac:MsgTypeCd>TXLifeRequest:204
      </ac:MsgTypeCd>
   </ac:MsgItem>
</ac:PostRq>
```

Content packaging is of particular concern. Insurance business interactions generally require documentation, which can be varied and copious. Acord Messaging Service Version 1.2 aggregated the requirements from the wrapped specifications by providing a `WorkFolder` construct that acts as a manifest for the set of attached messages, as seen in the next example:

```
<ac:PostRq>
   <ac:WorkFolder>
      <ac:MsgFile>
         <ac:FileId>cid:A01EFAE7-5490-43D0-DC
         </ac:FileId>
         <ac:FileFormatCd>text/xml
         </ac:FileFormatCd>
      </ac:MsgFile>
   </ac:WorkFolder>
</ac:PostRq>
```

Security functions in ACORD Web Services bindings use the standard SOAP recommendations. However, given the Acord Messaging Service Version 1.2 approach, ACORD created an additional set of security profiles that describe different levels of protection.[18] These profiles describe usage of Acord Messaging Service Version 1.2 extensions for the ACORD Referred Message Signature, the ACORD file digest, the ACORD file signature, and the ACORD file cipher.

Aspects of messaging and the infrastructure binding layer in HL7

HL7's messaging components exemplify the infrastructure binding layers within a Top-Down Modeling pattern. Just as high-level HMDs are exposed as XML, lower levels of messaging detail are similarly defined. At this level, the methodology includes storyboards and use cases, which define the application roles and the purpose of the information exchanged between health-care applications, and trigger events defining what prompts information exchange. Defining the information message content in the context of an ITS (e.g., the XML ITS), the required set of classes, attributes, and associations are used to develop an HMD, which provides a base template for a specific "message type"—essentially a unique set of constraints for an XML schema.

A message using the XML ITS is wrapped in SOAP and transported using secure HTTP over the Internet. The use of SOAP+WSDL is defined in a Web Services profile.[19] HL7 has announced the approval of Draft Standards for Trial Use (DSTUs) built around the HL7 RIM and has clarified its methodology from modeling to message definition and ultimately to an XML syntax representation.

Use of WSDL with the HL7 XML ITS for early industry prototypes revealed tooling issues due to the complexity of the XML ITS schemas. Typical Web Services tooling implementations using stubs and skeletons (i.e., software that is generated from service descriptions) result in the total encompassing of the content along with any ITS transmission wrappers. This is often not consistent with the domain-level application's programming model, which is not concerned with transmission wrapper semantics or processing of its constructs. This highlights the difficulties and ramifications of using differing approaches in industry-specific specifications of process indicators, such as acknowledgements, which are not specific to any industry and can be accommodated either by infrastructure technologies or a widespread higher-level standard. In the future, this may be done by a common cross-industry envelope architecture. Increased commonality in the area of content wrappers and envelopes across industries will provide off-the-shelf and open-tooling packages with increased efficiency for domain application development.

Because the Top Down Modeling pattern lacks a layered processing model and support for multiple constraint mechanisms, it is inevitable that customized code will be used when infrastructure bindings are implemented. The use of wrappers and envelopes at the technology layer does not cleanly match the classic usage of the Web Services infrastructure tooling implementation built around stub and skeleton code generation. This presents a challenging area for future work in both the information-modeling layer architecture and the infrastructure.

CONCLUSIONS AND DIRECTIONS

We have presented a set of emerging design patterns within vertical-industry standards organizations. Experience, heritage, organizational principles, established standards, support for deployed products, and changing technology all play a role in defining differences between these patterns and the organizations using them, and defining how the patterns map to the evolving Web Services infrastructure. Although these patterns have, in some cases, significant differences that must be accommodated and considered when used in a Web Services environment, their small number (four) is encouraging. It is most likely that the number of patterns that will emerge across industries will be of the same order, suggesting that Web Services provides a viable strategy for a commercial infrastructure for industry-level standards organizations. Continued examination of patterns associated with industry-level standards development provides a unique opportunity for the Web Services infrastructure, because the resulting pattern-related best practices hold the potential for providing a standardized means of optimizing the adoption of the Web Services infrastructure.

**Trademark, service mark, or registered trademark of Massachusetts Institute of Technology, Object Management Group, Inc., Open Applications Group, Inc., or OpenTravel Alliance, Inc. in the United States, other countries, or both.

CITED REFERENCES

1. World Wide Web Consortium, http://www.w3.org.
2. *Extensible Markup Language (XML) 1.0*, W3C Recommendation (February 1998), http://www.w3.org/TR/1998/REC-xml-19980210.
3. Open Applications Group, http://www.openapplications.org/index.htm.
4. Association for Cooperative Operations Research and Development, http://www.acord.org/home.aspx.
5. *ACORD Messaging Service XML Specification and SOAP Implementation Guide Version 1.2.0,* Association for Cooperative Operations Research and Development (April 2005).
6. R. Bilorusets, D. Box, L. F. Cabrera, D. Davis, D. Ferguson, C. Ferris, T. Freund, M. A. Hondo, J. Ibbotson, L. Jin, C. Kaler, D. Langworthy, A. Lewis, R. Limprecht, S. Lucco, D. Mullen, A. Nadalin, M. Nottingham, D. Orchard, J. Roots, S. Samdarshi, J. Shewchuk, and T. Storey, *Web Services Reliable Messaging Protocol* (February 2005), http://specs.xmlsoap.org/ws/2005/02/rm/ws-reliablemessaging.pdf.
7. *Business Object Document Architecture, OAGIS Release 6.2,* http://lists.ebxml.org/archives/ebxml-transport/200003/doc00006.doc.
8. *Simple Object Access Protocol Specifications*, World Wide Web Consortium http://www.w3.org/TR/soap12-part1/.
9. The MedBiquitous Consortium, http://www.medbiq.org/.
10. *MedBiquitous Web Services Design Guidelines, Version 1.0*, Medbiquitous Technical Steering Committee (April 2004), http://www.medbiq.org/technology/tech_architecture/webservicesguidelines.pdf.
11. *Web Services Interoperability Organization Basic Profile 1.1*, Web Services Interoperability Organization (August 2004), http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html.
12. Open Travel Alliance, http://opentravel.org/.
13. *OpenTravel Alliance Release OTA2005A Common Types Schema—OTA_CommonTypes.xsd*, OpenTravel Alliance, Inc. (2005), http://www.opentravel.org/2005A/OTA_CommonTypes.xsd.
14. *OASIS ebXML Message Service Specification Version 2.0*, Organization for the Advancement of Structured Information Standards (April 2002), http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
15. Health Level Seven, http://www.hl7.org/.
16. *Web Services Description Language 1.1*, World Wide Web Consortium (March 2001) http://www.w3.org/TR/wsdl.
17. *OAGIS Web Services Work Group*, Open Applications Group (October 28, 2003), http://www.openapplications.org/wg/WebServices.htm.
18. *Security Profiles for the ACORD Messaging Service Version 1.0.0,* Association for Cooperative Operations Research and Development (April 2005).
19. R. Ruggeri, M. de Graauw, L. F. Cabrera, M. Regio, G. Grieve, A. Julian, J. Larson, D. Pratt, and R. Spronk, *HL7 Version 3 Standard: Transport Specification—Web Services Profile, Release 2*, http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-wsprofiles.htm.

Accepted for publication December 16, 2005.
Published online May 18, 2006.

Scott R. Hinkelman
IBM Software Group, 11501 Burnet Road, Austin, Texas 78758 (srh@us.ibm.com). Mr. Hinkelman is a senior software engineer whose work is focused on industry-level standards organizations and alignment with service-oriented architectures (SOAs). He serves on IBM's Emerging Technology team, helping set strategy in engagements with industry organizations and has been working on service-oriented software for over five years. While working with the OpenTravel Alliance, Mr. Hinkelman was elected to its interoperability committee and provided the technical foundation for the initial definition of XML B2B message architecture for the travel industry. He has served as the Chief eBusiness Architect for IBM's Travel Industry Solution unit. He is currently the MedBiquitous.org Web Services architect, serving on its technical steering committee, and the leader for the quality of service area in RosettaNet's WS-I Web Services profile work. He has been instrumental in many industry-wide and international standards organizations and initiatives. An accomplished Java and XML expert, he represented IBM for the JAX-RPC 1.0 specification, which defined the client programming model for Java Web services. He has published numerous articles, chaired standards conferences, and holds patents in distributed computing. Mr. Hinkelman's interests are focused on consistency in SOA design across industry standards.

Donald Buddenbaum
IBM Software Group, 4205 South Miami Blvd, Durham, North Carolina 27703 (buddenba@us.ibm.com). Mr. Buddenbaum

is a software engineer concentrating on emerging standards and vertical-industry standards organizations. He has helped leverage IBM middleware as the basis for finanical service solutions, serving for a time as the Chief Architect for IBM's Software Group insurance solutions. Before joining IBM, he spent time designing and implementing solutions in the life insurance industry at various independent software vendors and insurance companies. His current work targets the adoption of horizontal standards within vertical-industry standards organizations, such as ACORD.

Liang-Jie Zhang
IBM Research Division, 19 Skyline Drive, Hawthorne, New York 10532 (zhanglj@us.ibm.com). Dr. Zhang is a research staff member and the chair of the Services Computing Professional Interest Community at the Watson Research Center. He has been leading service-oriented architecture (SOA) services research since 2001. He was the Chief Architect of industrial standards at IBM. He has filed more than 30 patent applications in the areas of e-business, Web Services, rich media, data management, and information appliances and has published more than 80 technical papers in journals, book chapters, and conference proceedings. Dr. Zhang chairs the IEEE Computer Society Technical Committee on Services Computing and serves as editor-in-chief of the International Journal of Web Services Research (JWSR), which has been included in the Engineering Index Compendex database since 2005. He was the general co-chair of the 2005 IEEE International Conference on Web Services (ICWS 2005) and the 2005 IEEE International Conference on Services Computing (SCC 2005). Dr. Zhang received a B.S. degree in Electrical Engineering from Xidian University in 1990, an M.S. degree in electrical engineering from Xi'an Jiaotong University in 1992, and a Ph.D. degree in computer engineering from Tsinghua University in 1996. ■