B2B Integration over the Internet with XML – RosettaNet Successes and Challenges

Suresh Damodaran
Chief Technologist, RosettaNet
(On loan from Sterling Commerce)
1851 East First Street #1050, Santa Ana, CA, USA-92705
+1.714.480.3820
Suresh.Damodaran@RosettaNet.org

ABSTRACT
The practical experience of RosettaNet in using Web technologies for B2B integration illustrates the transformative power of Web technologies and also highlights challenges for the future. This paper provides an overview of RosettaNet technical standards and discusses the lessons learned from the standardization efforts, in particular, what works and what doesn't. This paper also describes the effort to increase automation of B2B software integration, and thereby to reduce cost.

Categories and Subject Descriptors
I.7.2 [**Document Preparation**]: *format and notation, markup languages, languages and systems, standards.*

General Terms
Design, Standardization, Languages.

Keywords
B2B integration, XML, messaging services, business process, PIP.

1. INTRODUCTION
RosettaNet, founded in 1998, is a non-profit consortium of more than 400 of the world's leading Information Technology (IT), Electronic Components (EC), Semiconductor Manufacturing (SM), and Solution Provider (SP) companies working to create, implement, and promote open e-business process standards. RosettaNet is named after the Rosetta Stone, which is inscribed with the same message in three languages that enabled scholars to decipher Egyptian hieroglyphics. In a similar way, the mission of RosettaNet is to establish a common language and standard processes for business-to-business (B2B) transactions.

1.1 B2B Transaction Components
B2B transactions are a significant form of today's commercial activity. The businesses or parties involved in such transactions are called *trading partners*. Business transactions have two major components: private processes and public processes. Both of these components may be automated.

WWW2004, May 17-22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

1.1.1 Private Process Automation
Private process automation involves automating the trading partners' internal business processes, such as Enterprise Resource Planning (ERP) systems. Trading partners integrate software systems inside their enterprises in a myriad of ways. Because the systems and techniques for private business processes develop uniquely over time for each trading partner, RosettaNet has determined that standardizing this component of automation is not within its scope.

1.1.2 Public Process Automation
The public process is the part of the business process that is visible to both partners. Automating the public business process involves automating the sending and receiving of business documents between partners in a manner mutually agreed upon and understood. Because the trading partners share the business documents and business processes, standardizing the documents and processes facilitates a common understanding of the syntax and semantics of the business processes, which, in turn, facilitates automating the business processes efficiently. The goal of RosettaNet is to establish standards to automate public processes and thereby make B2B transactions efficient and economical.

1.2 Partner Interface Process (PIP[1])
RosettaNet developed the Partner Interface Process (PIP) specification for public business processes between trading partners (see Figure 1). The PIP standardizes *public* (business) process automation by standardizing business documents, the sequence of sending these documents, and the physical attributes of the messages that define the quality of service. RosettaNet also defines the messaging system used to send and receive these documents. The business documents are delivered to business services defined by each trading partner.

Figure 1 also shows that a PIP-based B2B transaction requires a trading agreement in place between trading partners. In practice, such an agreement is created prior to executing B2B transactions.

[1] PIP is a registered trademark of RosettaNet.

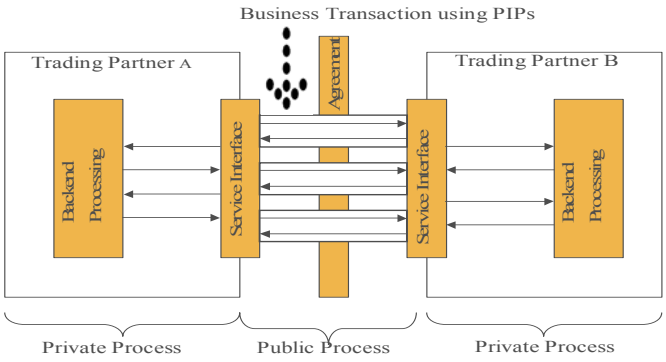

Figure 1. Public and private processes

A PIP is *validated* only after it is used in an actual implementation between trading partners. Currently, more than 50 validated PIPs are available for the business areas described in the honeycomb diagram (see Figure 2). RosettaNet divides the entire supply chain domain into clusters and segments. Each PIP is categorized according to the cluster and segment to which it belongs. For example, PIP3A4 is the fourth PIP in Segment A of Cluster 3 of the RosettaNet classification system.

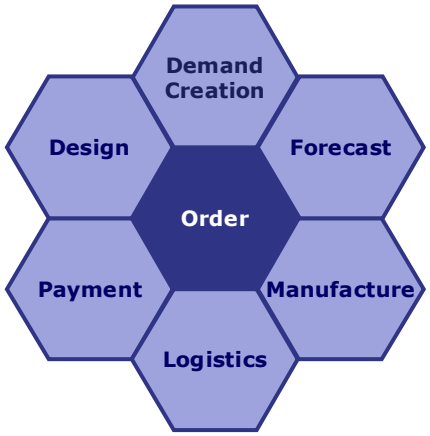

Figure 2. Business process areas

For details on PIPs in the RosettaNet clusters and segments, and to download PIPs for free, visit the RosettaNet Web site at www.rosettanet.org.

1.3 PIP Structure

A PIP for a public process defines exactly two *roles* for the trading partners that participate in the business process. For example, PIP3A4 [4], Request Purchase Order, defines the roles *buyer* and *seller*. The business process is divided into one or more business activities. In PIP3A4, Request Purchase Order, the business activities are Request Purchase Order and Confirm Purchase Order. The messages (business documents) exchanged between the roles during the business activities are called *action* messages. The Request Purchase Order business activity sends a Purchase Order Request from the buyer to the seller. The seller activates the Confirm Purchase Order business activity and sends a Purchase Order Confirmation to the buyer, who acknowledges, at the line level, if the purchase order is accepted, rejected, or pending.

The PIP specifications for a business process define the:

- Structure of the action messages
- Sequence in which the messages are sent between roles
- Quality of service attributes for the message exchanges

1.4 Purpose of Paper

The use of XML to increase automation led RosettaNet to shift from defining PIPs based on Document Type Definition (DTD) to defining PIPs based on XML Schema [8]. The RosettaNet effort to develop B2B integration standards uses XML to create specifications, and the XML-based messages are transported over the Internet.

This paper compares the interchange structure of PIPs defined using DTDs and PIPs defined using XML Schema, and explains why RosettaNet has begun defining PIPs using XML Schema. The paper concludes with an evaluation of the success of RosettaNet and the challenges RosettaNet is currently facing in its efforts to reduce the cost of automation and thereby increase the rate of adoption of B2B integration using RosettaNet standards.

2. PIP MESSAGE INTERCHANGE STRUCTURE

A PIP message is used to interchange information between trading partners. Therefore, the structure of a PIP message is called the *interchange structure*. PIPs can be categorized according to their interchange structure as either explicit or dictionary-based. The explicit and dictionary-based interchange structures were developed to meet the different needs of RosettaNet user communities.

2.1 Explicit Interchange Structure

Consider the following hypothetical PIP message fragment:

```
<PhoneNumber>10</PhoneNumber>
```

The syntax of PhoneNumber is specified in the XML Schema or DTD corresponding to the PIP message, and can be validated with the DTD or XML Schema, respectively. PIPs that use the explicit approach to define the syntax of interchange structures are referred to as *explicit* or *fixed* PIPs.

2.2 Dictionary-Based Interchange Structure

The following representation illustrates another manner of presenting the same PIP message fragment shown in section 2.1.

```
<property>
<name>PhoneNumber</name>
<value>10</value>
</property>
```

In this representation, PhoneNumber is the value of *name,* which is a property specified as a <name,value> pair. The syntax of the value of PhoneNumber can only be specified as the syntax of the *value* element. If the property element is used to define many <name,value> pairs, the syntax of the value element has the least restrictive of the syntaxes of all the values of all the possible different properties. Obviously, validating the value against such a syntax does not serve any useful purpose.

Another modification to the message fragment references a dictionary that stores the syntax and semantic constraints corresponding to the PhoneNumber attribute of property, as shown in the following sample:

```
<property DictionaryReference=RND834-1-34>
<value>10</value>
</property>
```

Note the absence of PhoneNumber in the message! Instead, the DictionaryReference attribute of property points to the definition of PhoneNumber in a dictionary. Only the *value* of PhoneNumber is sent. PIPs that reference a dictionary for property values have the dictionary-based interchange structure. Using a dictionary permits referencing different properties in different PIP messages, all of which can be validated with the same XML Schema. However, a dictionary is necessary to resolve the dictionary references during execution. Within RosettaNet, using a dictionary to create and use PIPs is called the *reflective* approach; however, in this paper, the approach is referred to as *dictionary-based.*

The fundamental difference between the explicit and dictionary-based methods of representing business documents is the way the syntax of the XML message is specified. In the explicit interchange structure, the syntax of all the elements of the interchange structure is provided in the corresponding XML Schema or DTD, whereas in the dictionary-based PIPs, the syntax of some or all the elements is defined in an accompanying dictionary.

2.3 Representing Code Lists in Explicit and Dictionary-Based PIPs

A familiar and interesting case to consider is how code lists are represented using these two approaches. Code lists define a long list of codes for an element, for example a CountryCode, with enumerated values such as US. These enumerated codes are called *entity instances.* In explicit PIPs, the entity instances of a code list are located in the schema associated with the XML message. In dictionary-based PIPs, the entity instances are in the dictionary, and the PIP message references them. When countries are added or removed, the schema of the interchange structure does not change in the dictionary-based PIP, only the dictionary changes. In contrast, when the explicit approach is used to represent interchange structure, the DTD or XML Schema of the interchange structure must be changed to reflect changes in country codes.

2.4 Advantages and Disadvantages of the Approaches

The approach used to create the explicit PIPs is also used to create most DTDs and XML Schemas in use today. Therefore, this approach is widely known. It works well when the types used in a business document do not change much during the course of its use. In contrast, the dictionary-based approach is used when the types can change over the course of the use of a PIP specification. Today, the dictionary-based approach is used within RosettaNet for exchanging technical product (for example, fixed resistor) information. The products change rapidly (for example, components of a digital camera) due to technological advances. Therefore, the types of these products are defined in the RosettaNet Technical Dictionary (RNTD), which is defined by the RosettaNet community.

However, it is worth noting that it is possible to have an equivalent explicit interchange structure corresponding to a dictionary-based interchange structure. Such an explicit interchange structure requires that the entire dictionary is available in the XML Schema of the explicit interchange structure. Making the entire dictionary part of the interchange structure is somewhat cumbersome from an implementation point of view, and any change in the dictionary content requires revising the schema of the interchange structure itself.

3. XML FOR AUTOMATION: FROM MONOLITHIC TO MODULAR PIPS

RosettaNet has recently completed an important transition from using DTD to define the structure of PIPs to using XML schema. PIPs based on DTD, called *monolithic,* are specified using a considerable amount of free-formatted text, tables, and diagrams that are not machine interpretable. The XML Schema-based PIPs are called *modular* because they are created with reusable structures. Modular PIPs are specified with considerably more machine-interpretable specifications than PIPs based on DTD. This section examines this transition and the reasons for it.

3.1 Anatomy of a DTD-Based PIP

Until 2003, the normative specification format for PIPs was DTD, which includes:

- DTDs for action messages
- *Message guidelines* in HTML that describe additional details and constraints that the DTDs for business documents cannot describe
- Business activities in the business process and sequence of exchange of the action messages in a UML activity diagram
- Quality of service attributes of the action and signal messages

Most RosettaNet solutions currently implemented use PIPs based on DTD. PIP3A4, the Request Purchase Order package[4], illustrates such a design.

3.1.1 DTD

In PIP3A4, two interchange structures, Purchase Order Request, and Purchase Order Confirmation, are sent between the buyer and seller. Below is an excerpt from the DTD of the Purchase Order Request document.

```
<!ELEMENT PurchaseOrder
(AccountDescription?,
comments?, ContractInformation*,
DocumentReference*, FinancingTerms*,
generalServicesAdministrationNumber?,
GlobalGovernmentPriorityRatingCode?,
GlobalPurchaseOrderFillPriorityCode?,
GlobalPurchaseOrderTypeCode+,
governmentContractIdentifier?, installAt?,
isDropShip, OrderShippingInformation?,
ProductLineItem+, proprietaryInformation?,
requestedEvent?, requestedShipFrom*,
SecondaryBuyer?, shipTo?,
TaxExemptStatus?, totalAmount?)>
```

3.1.2 Message Guidelines

The DTD for PIP3A4 is augmented with message guidelines illustrated in the following excerpt from the Request Purchase Order DTD. The message guidelines help map the XML message corresponding to the DTD to the document formats and structures used in the private process. The first column of Table1 lists the line number, the second specifies the cardinality, and the third

lists the expanded elements. Vertical bars in the last column indicate the hierarchy of the elements.

Table 1. Message guidelines

Line No	Card	Element
14	1	PurchaseOrder
15	0..1	|-- AccountDescription
16	1	| |-- accountName.FreeFormText
17	0..1	| |-- AccountNumber
18	0..1	| |-- billTo.PartnerDescription
19	1	| | |-- BusinessDescription
20	0..1	| | | | businessName.FreeFormText
21	0..1	| | | |-- GlobalBusinessIdentifier

The message guidelines also describe constraints on the message fields that cannot be expressed in a DTD, as shown in the following sample constraint on Payment Terms:

PaymentTerms

Constraint: If PaymentTerms is used, then at least one occurrence of netTermsDay.DayOfMonth or netTermsDays.CountableAmount is required.

In XML Schema-based PIPs, the representation of many constraints is possible in the schema itself, which obviates the need for message guidelines, as described in Section 3.4.

3.1.3 Business Activities and Sequences

The exchange sequence of action messages for a specific PIP is described in the specification guide for that PIP. Figure 3 illustrates the UML activity diagram for accomplishing Request Purchase Order business activity for PIP3A4. The PIPs follow one or more business transaction patterns, such as request-response or notification. Depending on whether one or two action messages exist in a PIP, a RosettaNet PIP can also be classified as a *one-action* PIP or a *two-action* PIP. A request-response business transaction pattern results in a two-action PIP, whereas a notification business transaction pattern results in a one-action PIP. The UML diagram is not machine-readable; therefore, modular PIPs have an XML description of these activity diagrams. See Section 3.5.

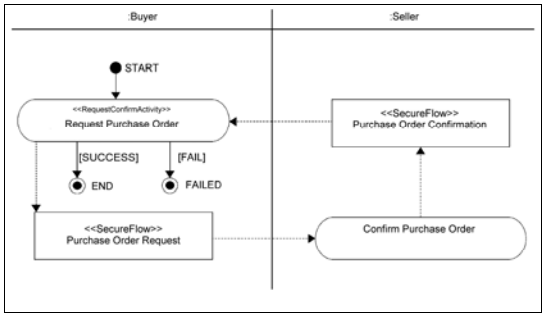

Figure 3. Business document flows

3.1.4 Message Attributes

The attributes of messages sent between the trading partners are also specified. These attributes are related to the quality of service. Table 2 lists the attributes of an action message between a buyer and a seller. Note that there are two additional messages in Table 2: two instances of Receipt Acknowledgment, a signal message corresponding to the action messages of Purchase Order Request, and Purchase Order Confirmation.

Table 2. Message attributes

	Name	Time to Acknowledge Receipt Signal	Time to Acknowledge Acceptance Signal	Time to Respond to Action	Included in Time to Perform	Is Authorization Required?	Is Non-Repudiation Required?	Is Secure Transport Required?
	Purchase Order Request Action	2 hrs	N/A	24 hrs	Y	Y	Y	Y
	Receipt Acknowledgment	N/A	N/A	N/A	Y	Y	Y	Y
	Purchase Order Confirmation Action	2 hrs	N/A	N/A	Y	Y	Y	Y
..	Receipt Acknowledgment	N/A	N/A	N/A	N	Y	Y	Y

Action messages are acknowledged by positive or negative signals. A Receipt-Acknowledgment signal acknowledges that a message has been received and is syntactically validated, whereas an exception signal indicates an error. Section 3.5 describes how the content of Table 2 is specified in XML in the modular PIP specifications.

3.2 Problems with the Monolithic Model

RosettaNet originally used a distributed approach to creating PIPs. That is, different teams defined the business requirements for each PIP, and each set of business requirements was given to engineers or contractors who often handcrafted individual PIPs without any effort to coordinate the work. This approach created many problems:

- Inconsistencies in the naming, syntax, and semantics of the elements used in the PIPs by implementers who interpret the PIP specifications increase the effort and cost of implementing the PIPs. RosettaNet now uses the RosettaNet business dictionary to ensure consistency among PIPs.
- Absence of reuse of XML types and element names across multiple PIPs. RosettaNet is addressing this problem by creating a set of commonly used types called Universal Structures. To facilitate unique identification of the types used in PIPs, namespaces are used.
- Handcrafting of DTDs and other specifications introduces human errors, and makes it difficult to create a consistent syntax for PIP specification.

3.3 Improving PIP Specifications

Although PIPs based on DTD have automated implementing and running RosettaNet solutions for B2B transactions considerably, PIP specifications must have more machine-readable content to further these improvements. As the discussion of the message guidelines document in section 3.1.2 illustrates, monolithic PIPs describe certain types of constraints in free-formatted text. Therefore, humans must interpret these constraints and convert them to software to implement the PIPs, which increases errors and cost. Reducing or eliminating human intervention is crucial to improving automation and reducing cost. To achieve this goal, RosettaNet has begun using XML schema to specify interchange structure, Schematron [7] to specify most constraints, and ebXML

BPSS specifications [2] to specify the activity diagrams and attributes of messages, as demonstrated by the *modular* PIP.

3.4 Modular PIPs

PIP specifications based on XML Schema are called modular because they are created using reusable structures. A modular PIP is specified using the following components:

- XML Schema-based PIP interchange structure with embedded machine-readable constraints
- *Message structure,* that is, a spreadsheet of the interchange structure (in the same format as the message guidelines for monolithic PIPs)
- Specifications of business activities and action message exchange sequences in XML that conforms to ebXML Business Process Specification Schema (BPSS)
- Specifications of attributes of messages in XML that conforms to ebXML Business Process Specification Schema (BPSS)

3.4.1 XML Schema for Interchange Structure

The XML for a modular PIP is created from UML models using software that requires almost no human intervention. The UML models are created by PIP engineers. The UML models are drawn according to the rules specified by RosettaNet to facilitate automated translation of the UML models to XML. The models for interchange structures are created according to the business requirements defined in a RosettaNet milestone program. Using an automated production line to create XML code that uses the XML and UML guidelines reduces the errors that would result if the XML code were created manually. The XML Schema for a PIP is also created using a set of rules specified by RosettaNet. The following excerpt from the XML Schema of PIP PIP4E1, Sales Report Notification, illustrates how XML Schema differs from the DTD representation.

```
<xs:element name="SalesReportNotification"
type="tns:SalesReportNotificationType"/>
 <xs:complexType name="ProductTransferType">
        <xs:annotation>
            <xs:appinfo>
                <urss:Definition>This object describes a POS
(point of sale) information for known/unknown end
user(s).</urss:Definition>
                <urss:Context/>

 <urss:CreationDate>08/01/2004</urss:CreationDate>
                <urss:Keyword/>

 <urss:LastUpdatedDate>08/01/2004</urss:LastUpdatedDate>
                <urss:TypeVersion>1.0</urss:TypeVersion>
            </xs:appinfo>
            <xs:documentation>
                <urss:Purpose/>
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="dsspt:ParticipatingPartner"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SalesReportLineItem"
type="tns:SalesReportLineItemType"
maxOccurs="unbounded"/>
        </xs:sequence>
            <xs:attribute                    name="schemaVersion"
type="xs::listOfVersions" fixed="1.0"/>
        </xs:complexType>
…
</xs:element>
```

Note the use of namespaces in the XML example describing constraints. The use of namespaces allows precise identification of the type of any element in a PIP message. Also, note the use of version for types, using the attribute *TypeVersion* for the complexType *ProductTransferType*. Versioning of types allows evolution of types, and also can be a useful feature when message fragments with values of a few elements are exchanged. The attribute *schemaVersion* may be used to identify the versions of the schema that the message fragments conform to.

3.4.2 XML Schema and Constraints

XML Schema is capable of representing many of the constraints pertaining to single elements. In monolithic PIPs, representing these constraints requires message guidelines. For example, the constraint

"`Only allowed value of GlobalDocumentFunctionCode is "Request"`"

is easily represented using XML Schema. However, XML Schema does not allow expressing constraints that depend on multiple elements, except in cases where hierarchical relations can be used. For example, the following constraint cannot be expressed in XML Schema:

"`There must be at least one occurrence of ProductLeadTime, unless isOnAllocation is "yes", in which case ProductLeadTime is optional.`"

This type of constraint can be divided into simpler constraints as shown below, and later specified using Schematron [7]:

`If value of` *`isOnAllocation`* `is equal to` *`No`* `then the occurrence of` *`ProductLeadTime`* `has to be` *`1`*`.`

`If value of` *`isOnAllocation`* `is equal to` *`Yes`* `then the occurrence of` *`ProductLeadTime`* `has to be` *`0..n`*`.`

Although XML Schema and Schematron can represent many types of constraints, some constraints must still be specified in free-form text because of current limitations of the RosettaNet Integration Architecture. For example, when the value or cardinality of an element depends on the existence or value of a document extraneous to the current document, this type constraint must be specified in free-form text, as shown here:

"`The isFinalForecast.AffirmationIndicator must match the value used in the forecast referenced by previousForecastNotificationIdentifier`"

RosettaNet expects that even this type of constraint will be expressed in machine-interpretable format in the future.

3.5 Business Activity and Message Attributes

The business activities, exchange sequence of action messages, and the attributes of messages are specified using ebXML Business Process Specification [2]. ebXML BPSS provides an XML Schema, and the PIP specifications for business activity and message attributes are described in XML that conforms to this

XML Schema. Below is an excerpt from the PIP4A3 specification.

```
<BusinessTransaction
    name="Notify Of Threshold Release Forecast"
    nameID="NotifyOfThresholdReleaseForecast_BT"
    isGuaranteedDeliveryRequired="true">

  <RequestingBusinessActivity
    name="Threshold Release Forecast Notification Action"
    nameID="ThresholdReleaseForecastNotificationAction"
    isAuthorizationRequired="true"
    isIntelligibleCheckRequired="true"
    isNonRepudiationReceiptRequired="true"
    isNonRepudiationRequired="true"
    timeToAcknowledgeReceipt="PT2H"
    retryCount="3">
    …
</BusinessTransaction>
```

In this excerpt, note the message attributes of RequestingBusinessActivity with name Threshold Release Forecast Notification Action. In contrast, these attributes are expressed in a table in monolithic PIPs.

4. ROSETTANET IMPLEMENTATION FRAMEWORK (RNIF)

In addition to defining the PIP specifications, RosettaNet also defines an infrastructure to transport PIP messages. This infrastructure is called RosettaNet Implementation Framework (RNIF). The RNIF specification defines the packaging, routing, and transport of all PIP action messages and signal messages. The specification of security mechanisms (signature, encryption) and reliability mechanisms at the message level is part of the implementation framework. Action messages are acknowledged by positive or negative signals: A Receipt-Acknowledgment signal acknowledges that a message has been received and is syntactically validated, whereas an exception signal indicates the opposite. Error codes identify the types of errors. Two RNIF specifications are in use today: RNIF 1.1, released in November 1999, and RNIF 2.0, an improved messaging specification released in 2002. This paper describes only RNIF 2.0.

4.1 Components of the RosettaNet Business Message

This section contains definitions of the components of the PIP message from *RosettaNet Implementation Framework: Core Specification*, *V02.00.01* [6]. The structure of a RosettaNet Business Message is illustrated in Figure 4. The components of the message that relate to transporting the message are defined.

4.1.1 Preamble Header

This header identifies the RNIF version the XML message complies with. This header is mandatory and occurs only once in a message. This header is never encrypted.

4.1.2 Delivery Header

The information contained in this header overlaps the information in the Service Header; nevertheless, this header is separate from the Service Header to allow access to this information by an intermediary. An intermediary relays PIP messages from a sender to a receiver. This header is never encrypted so that an intermediary can read this information.

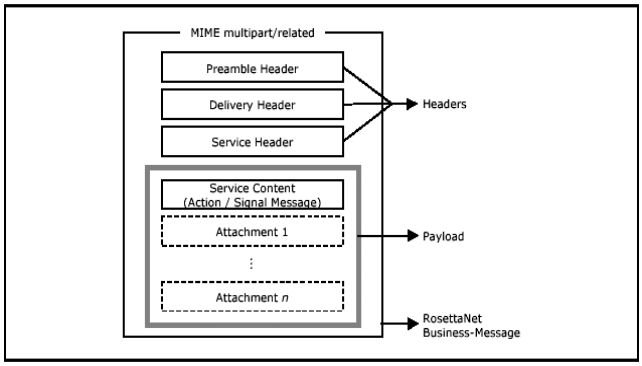

Figure 4. RosettaNet RNIF 2.0 compliant message

This header contains sending and receiving trading partner identities, tracking ID, and date/time stamp, and is created by the sender of the message. This header is mandatory in a PIP message and occurs only once. This header is never encrypted. It may also specify whether secure transport of the message is required.

4.1.3 Service Header

This header identifies the PIP, the PIP instance, the activity, and the action to which this message belongs. This header is mandatory in a PIP message and occurs only once. This header may be encrypted.

4.1.4 Payload

The payload contains the Service Content, which is the action or signal message for a specific PIP. The payload may be encrypted. Zero or more attachments may be packaged along with Service Content. Attachments are documents or files that are not part of the Service Content but need to be packaged and sent as a part of the message. The payload, the attachments, or both can be encrypted.

4.2 Creating a RosettaNet Business Message

A RosettaNet Business Message is a logical grouping of the Service Header, Payload, Delivery Header, and Preamble Header. Creating a RosettaNet Business Message has three steps (see Figure 5).

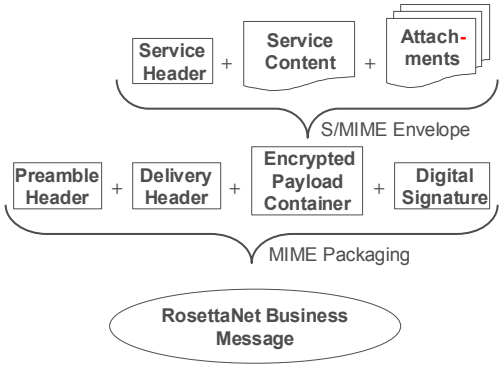

Figure 5. RNIF 2.0 business message construction

The Service Content (from a specific PIP) and any attachments with it are wrapped together with the Service Header, optionally inside an encrypted S/MIME [1] envelope. This envelope is packaged using Multipurpose Internet Mail Extensions (MIME) specifications [3] with the Preamble Header, Delivery Header,

and, optionally, Digital Signature, to create the RosettaNet Business Message. The process of unpacking a RosettaNet Business Message follows the reverse procedure of packing.

5. SUCCESSES, CHALLENGES, AND THE ROAD AHEAD

RosettaNet has achieved tremendous adoption of its standards in the high-tech manufacturing supply chain over the past three years with more than 3000[2] documented production implementations and a growth rate of approximately 500% from 2001 to 2003. The standard's broad acceptance in the marketplace is demonstrated by the growing number of RosettaNet-based transactions being conducted, estimated at several billion in annual revenues. The following key decisions that RosettaNet took early in its development have contributed to this widespread adoption:

- Focus on standardizing the public process. Get the business experts to define the business aspects of the business process, and ensure implementation of the business process by obtaining commitment from the parties involved.
- Define and promote interoperability and conformance of messaging services.
- Create standards with global use in mind.

5.1 Cost: Roadblock to Adoption

Although RosettaNet has achieved relatively widespread adoption in large companies, the cost of implementing RosettaNet solutions has slowed adoption among small and medium-sized businesses. With the exception of small and medium-sized businesses with customers that require RosettaNet to conduct business, small and medium-sized businesses are reluctant to make a significant investment if the number of transactions is low, and they cannot justify the return on investment. Several factors contribute to the cost.

5.1.1 RNIF Transport Requirement

RNIF implementations are secure, reliable, and robust. However, some of the RosettaNet trading partners may not require this level of security, reliability, and robustness.

5.1.2 Mapping Cost

To process a PIP message, it is necessary to map the PIP message structure to the back-end data structure. If PIP messages for different PIPs have inconsistent message structures with differing syntax and semantics for the same back-end data structure, the cost of mapping increases due to the human labor involved in interpreting each PIP message.

5.2 Reducing Costs

Reducing the cost of implementation is crucial to expanding adoption of RosettaNet standards. RosettaNet is working to reduce costs in several ways.

[2] Figure denotes production implementations as reported by RosettaNet Global Council Members only. Total number estimated in excess of 10,000.

5.2.1 Use of Multiple Messaging Services

RosettaNet is revisiting this requirement by investigating the feasibility of sending PIP messages over multiple messaging services, including RNIF.

5.2.2 Automate PIP Production

RosettaNet has begun to use an automated PIP production process that creates PIPs based on XML from UML models of PIPs. This process uses XML and UML guidelines to create consistent formats for the PIPs. Automation helps reduce manual errors.

5.2.3 Increase Consistency of PIP Data Structures

RosettaNet stores consistent definitions of business terms and data structures used in explicit PIPs in a dictionary. Use of the dictionary allows consistent naming, syntax, and semantics of the elements. This dictionary is called the RosettaNet Business Dictionary (RNBD). RNBD is currently implemented as a relational database. To aid in building and using consistent PIP message structures, RosettaNet is creating the RosettaNet Information Model for specific business areas so that related PIPs share the same data structures.

5.2.4 Consistent PIP Variations

Many companies require their trading partners to use variations of a specific PIP; sometimes several variations are required for different uses of the PIP. These variations have more constraints than those in the original RosettaNet PIP. RosettaNet has created the RosettaNet Appliance Enablement program to define formats and methods to standardize the methods and use of variations of PIPs. The RosettaNet Appliance Enablement program enables automated processing of trading-partner specific PIPs by defining clear syntax and semantics for the permissible changes to a PIP.

5.3 Challenges to Improving Efficiency

In the document-based B2B integration paradigm, the exchange of complete and legally valid business documents alone forms the basis of integration. However, RosettaNet partners have encountered problems implementing the pure document-based integration paradigm. RosettaNet is working on a new RosettaNet Integration Architecture to address these problems. Some of these problems are discussed in the following sections.

5.3.1 Large Messages with Redundant Content

Evidence suggests that an XML message is from 7 to 10 times larger than the equivalent EDI message. Business processes such as collaborative forecasting require large amounts of forecasting data. The messages become large because often information sent in a prior message is repeated in the current message. Since current RosettaNet Integration Architecture does not specify a technique to identify the data sent in a previous PIP message, partners often send huge amounts of the same data multiple times. Therefore, high-volume, complex, collaborative B2B integration strains the infrastructure due to the huge amounts of data exchanged between partners. Additional strain in the infrastructure occurs when the data is processed. RosettaNet has tried to address the verbosity of XML by specifying compression techniques for the messages [5]. However, compressing the message addresses only the symptom, and the fundamental problem, the inability to avoid repetitious exchange of data, remains.

5.3.2 *Large Running Process with Related Content*

A typical order-to-cash scenario starts with the purchase order sent from buyer to seller, results in a shipped and received product, and ends with money paid to the seller. The entire scenario revolves around an *order*. The order, which is created and often modified during the process, defines how the shipment is made and payment is rendered. A number of PIPs can be used for the end-to-end business process, as shown in Figure 6, and the entire process may take days or weeks.

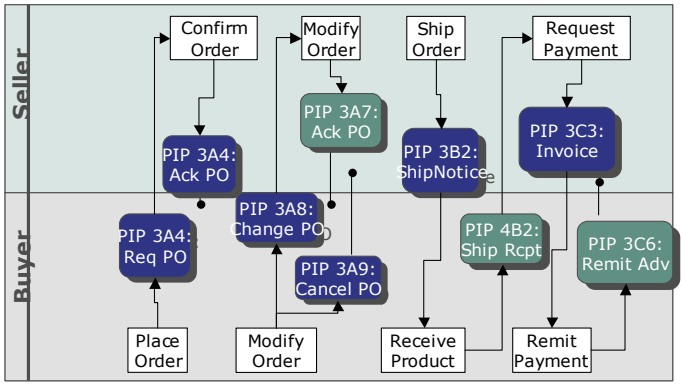

Figure 6. Order-to-cash business process

Each of the PIPs defines the interchange structures for the action messages it contains. However, these interchange structures do not share the order structure that is created with PIP3A4, modified with PIP3A7, shipped with PIP3B2, paid for with PIP3C6, and so on. Absence of the shared order structure results in both the buyer and the seller sending each other all the information related to the order on every PIP message. Although the order structure can be used to create a business context that can be shared among the various interchange structures, the absence of such a business context defined by RosettaNet results in trading partners creating their own ways of maintaining business context for long-running business transactions, such as order-to-cash. Often, these business contexts are modified by trading partner-specific business rules used in back-end applications. Defining this business context in a standardized way to facilitate standardized execution of related PIPs remains a challenge.

5.3.3 *Precise Message Error Identification*

When an error occurs in the processing of a PIP message, it would be useful for the sender to be able to identify the type and location of the error in the message. Incorporating such error identification mechanisms during the validation of the syntax and semantics of a message is one of the goals of the new RosettaNet Integration Architecture.

5.4 Other Obstacles

Implementation cost is not the only factor slowing the widespread adoption of RosettaNet PIPs. As large numbers of businesses within a specific supply chain adopt RosettaNet PIPs, it becomes necessary to provide services that support the increase. RosettaNet is proposing to address this issue through the RosettaNet Services Architecture. The implementation of this architecture should result in the ability to securely access profiles of trading partners for physical connection and enable users to access PIPs and their variations from a repository defined by RosettaNet.

6. CONCLUSION

RosettaNet has brought standardization of business processes to the XML-based business information exchange over the Internet. The original goal of this standardization has been to reduce cost while allowing disparate trading partners to conduct electronic commerce in a mutually understood way—both syntactically and semantically. RosettaNet is continuing to further the goal of reducing the cost of implementation and execution of these business processes. As discussed in this paper, making the specification of the business processes more machine interpretable results in fewer manual hours spent in reading and interpreting RosettaNet PIPs. Increased automation further reduces errors and related costs. RosettaNet is currently working on the challenges to making the execution of the business processes more efficient. The goal of making automated B2B integration affordable and accessible to large numbers of small and medium-sized businesses is being addressed by the definition of a services framework, and by standardizing even more aspects of B2B integration.

7. ACKNOWLEDGMENTS

Technology created in RosettaNet is the result of dedicated efforts by a large group of talented individuals. This paper draws upon their efforts. Two of these individuals considerably influenced my thinking: Paul Tearnen and Derek Coleman. Thanks to Karen Cox for help with the figures. Thanks to William Cook and Maggie Harris for help with this paper. Also thanks to Sterling Commerce for support during my time with RosettaNet.

8. REFERENCES

[1] Dusse, S., et al. S/MIME Version 2 Message Specification, RFC2311. http://www.ietf.org/rfc/rfc2311.txt.

[2] ebXML Business Process Specification Schema, Version 1.01, May 2001. http://www.ebxml.org/specs/ebBPSS.pdf.

[3] Freed, N., et al. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045, IETF, Network Working Group, 1996.

[4] PIP3A4: Request Purchase Order, Version 2.01, Aug. 2002. http://www.rosettanet.org.

[5] RosettaNet Implementation Framework V02.00.01: S/MIME Compression of RosettaNet Payload, Technical Advisory, July 2003.

[6] RosettaNet Implementation Framework: Core Specification, V02.00.01, March 2002, http://www.rosettanet.org.

[7] Schematron: An XML structure validation language using patterns in trees. http://www.ascc.net/xml/resource/schematron/schematron.html.

[8] XML schema part 1 and part 2: W3C recommendation, May 2001. http://www.w3.org/XML/Schema