



SECOND GENERATION LEGACY TO WEB STRATEGIES VIA XML PART 3 - DISCIPLINED XML

by
Don Estes

Table of Contents

- 3. Disciplined XML
- 3.1 No Quick Hit
- 3.2 KIS XML
- 3.3 Schema Controlled XML
- 3.4 Schema Definition Schema
- 3.5 XML Validation over the Life Cycle
- 3.6 Retrofitting XML into Legacy Applications
- 3.7 Conclusion

3. Disciplined XML

We have discussed the benefits of second-generation XML implementations with application integration strategies, particularly for business-to-business e-commerce. XML also supports globalization initiatives, by allowing localization of presentation and componentization of transaction processing modules, for both new development and legacy applications.

In particular, retrofitting XML into mainframe CICS programs allows native mode XML document creation where it makes the most sense: at the point where the data are organized for use rather than with middleware where there may be no appropriate error recovery logic in the application program for a validation failure. These upgraded CICS programs allow direct use of browsers without middleware, and re-use of the programs as transactional components.

But, there is a flip side to this fine story. Properly implemented, XML can bring substantial benefits. However, if not properly implemented, an enterprise may be better off without it.

3.1 No Quick Hit

Despite all the potential benefits, some of our clients have raised the contrary question of whether or not XML makes sense at all. This is a refreshing question, for answering it requires analysis from first principles, free of the hype currently surrounding XML. The quick answer is that in the short term, it probably doesn't make any sense.

XML offers no dramatic gains in efficiency in any one area that one can point to it and say, "see, there's the value!" Rather, the benefits of XML tend to be diffuse and cumulative rather than concentrated and specific.

We compare XML to SQL databases in several respects. When the first relational databases came into general release, similar questions were raised as to whether the benefits justified the cost. The benefits of the relational data model and of SQL data access are also diffuse and cumulative, and therefore hard to quantify while the costs are obvious. Standard data files store data just as well as relational tables, and provide better performance in many cases. SQL also provides a loosely coupled data interface into application programs in a way that is very similar to XML.

Indeed, we have identified no technical goals that require XML for accomplishment. Just as VSAM stores data just as well as DB2 or Oracle, and C++ produces applications that operate as well or better than those written in Java, so fixed record formats can do everything that can be done with XML. The point is that the cumulative gains from the many small efficiencies of a relational database over standard files proved its value after large-scale adoption, and the same will be true of XML.

What is required for a realistic, balanced appraisal is a focus on the cumulative gains over time, not the single "quick hit". If the standard is to be a quick payback, then your XML initiative, arguably, should be focused only on meeting external requirements and perhaps on new application development, with little attention paid to retrofitting XML selectively into existing applications.

On the other hand, if the business standard to be applied to XML is to be optimized value creation over time, (both through cost reductions and through enablement of new business opportunities)



90 New Montgomery Street
San Francisco
California 94105
Tel 415-543-1515
Fax 415-543-6701
www.forecross.com

then a broadened perspective emerges. In this case, we would recommend a strategy of balancing theoretical research and development with practical demonstration projects that are not throwaways, to provide the practical experience that a thoughtful manager wants without a wholesale commitment. Neither theory nor practice should be too far in front of the other, given the rapidly evolving nature of the XML standards and the need for disseminating practical knowledge of XML throughout the IT organization.

In essence, this approach combines a leap of faith that the cumulative value will appear in time to provide a positive return on investment from the efforts, with sufficient near term reality checks to allow mid-course corrections or outright abandonment if warranted. In our opinion, adopting an "XML ready" application package does not really address this issue, although it does expose members of the staff to XML technology. We recommend that small projects involving locally maintained programs be defined, perhaps involving only single programs, to explore this technology and evaluate how to ensure a positive return on investment over the medium and long term. Pragmatism and sensible cost/benefit analysis are the order of the day.

3.2 KIS XML

XML can be implemented in a way which is daunting to the most seasoned IT professionals, such that only Ph.D.'s in computer science feel completely comfortable with the results. Or, XML can be implemented in a minimalist way that captures the essential business value without overly complex constructs, and is readily accessible to all.

We are a strong proponent of the KIS rule (Keep it Simple) for XML. Clever XML, like clever logic in any program, has long term costs borne by those who must maintain it after the original author has had his or her fun exploring the neat features. Unless complex constructs can be justified in reducing the total life cycle costs of the application, we recommend against allowing them in your organization.

We teach an introductory XML course which is designed to focus on the basic constructs and what the business needs out of XML. In general, this will be satisfied by:

- ◆ A global understanding of data tagging and establishment of local standards for tags
- ◆ Basic XML document construction
- ◆ Basic schema construction and document validation

- ◆ Operational and programming strategies for ensuring integrity over time

The two advanced topics that we explore have to do with the schema definition schema and with XSLT, and these are usually reserved for senior technicians in data administration. Otherwise, we limit the scope to what a programmer needs to know for regular daily use. Again the analogy with SQL is useful: a programmer doesn't need to know all possible uses of SQL constructs, just those required for getting the job done. Similarly, programmers need roughly an equivalent level of XML knowledge in the beginning, with advanced knowledge coming over time.

How XML is implemented will determine whether or not there is a positive return on investment. If data tagging standards are not in place and enforced, if version coordination is not present and enforced, or if construct usage is not constrained to ensure that all can understand what was done in any given instance, then the gains could turn out to be minimal or even negative.

3.3 Schema Controlled XML

The early stage XML usages we have seen at major data centers reminds us of source control in the early days of the computer industry. Then, programs were written on an ad hoc basis, with no implementation standards, no source code library system, no global routines, and no COPY books. Every program had its own unique description of data elements and local implementation of common logic, leading to higher error rates and greater maintenance costs. If XML usage at your site is allowed to develop along a similar path and you expect to have more than a few trivial usages, then we recommend that any use of XML be reconsidered.

However, there is a mechanism provided for in the XML specification that allows for most of the appropriate standards to be enforced without imposition of an onerous bureaucracy: the schema. Any given instance of an XML document must be capable of being validated against the appropriate schema. The schema specification is in the process of replacing the Document Type Definition (DTD) specification that has proven inadequate for this purpose.

We recommend that the appropriate database administrator or data administration group develop XML schemas in concert with database definitions, and then publish those schema on the company intranet or on the mainframe. Where important data are stored in conventional file structures,



90 New Montgomery Street
San Francisco
California 94105
Tel 415-543-1515
Fax 415-543-6701
www.forecross.com

XML schemas can be developed for these as well. So long as both programming and operations can validate any given document at any time, and do so on a regular basis, then the minimum set of standards enforcement is in place for disciplined and productive results from XML usage. We recommend that XML be used with enforced schema control or not at all.

3.4 Schema Definition Schema

One of the major differences between a DTD and a schema is that a schema is an XML document, while the DTD used a non-XML syntax. Like any XML document, there is a global schema against which each schema must validate: the schema definition schema. (For mathematical completeness, the schema definition schema must validate against itself.)

There are two schema definition schemas in common currency at this time, one from W3C and one from Microsoft. If you write a schema to which your XML documents must conform, you may write it using either the W3C definition, the Microsoft definition, or in some new schema definition that may be published at any time. In principle, you could write your own schema definition schema, but in practice we think this is unlikely to be an appropriate strategy to follow. Some sites may choose to limit some of the advanced syntax available in either of the schema definitions in order to enforce the KIS rule, and this may be an appropriate reason to modify one or the other schema definitions for local use. However, we discourage this as well for new XML users, until experience shows what should and should not be used.

Consider the following COBOL record description:

COBOL Record Description	
01	MASTER-RECORD.
03	MASTER-REFERENCE-NO PIC 9(6).
03	MASTER-NAME PIC X(50).
03	MASTER-CURR-BALANCE PIC S9(8)V9(2).
03	MASTER-CURR-TRANS-DATE PIC 9(8).

If we had a single data record stored in a file of this description, and if we defined appropriate data tags for these items, we could encode that file into this XML document:

XML Document
<pre><?xml version = "1.0"?> <MASTER-FILE> <MASTER-RECORD> <REFERENCE-NO>000020</REFERENCE-NO></pre>

<pre><NAME>JOHN SMITH</NAME> <CURR-BALANCE>+123.45</CURR-BALANCE> <CURR-TRANS-DATE>2000-08-26</CURR- TRANS-DATE> </MASTER-RECORD> </MASTER-FILE></pre>
--

This document will validate¹ against the following schema, prepared using the W3C syntax:

W3C Syntax Schema
<pre><?xml version = "1.0" encoding="UTF-8"?> <xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchem a"> <xsd:element name="MASTER-FILE"> <xsd:complexType content="elementOnly"> <xsd:element name="MASTER-RECORD" type="MASTER-RECORDType" /> </xsd:complexType> </xsd:element> <xsd:complexType name="MASTER- RECORDType" content="elementOnly"> <xsd:element name="REFERENCE-NO" type="xsd:positiveInteger" /> <xsd:element name="NAME" type="xsd:string" /> <xsd:element name="CURR-BALANCE" type="xsd:decimal" /> <xsd:element name="CURR-TRANS-DATE" type="xsd:date" /> </xsd:complexType> </xsd:schema></pre>

This document will also validate² against the following equivalent schema, prepared using the Microsoft syntax:

Microsoft Syntax Schema
<pre><?xml version = "1.0"?> <Schema xmlns="urn:schemas-microsoft- com:xml-data" xmlns:dt="urn:schemas-microsoft- com:datatypes"> <ElementType name="REFERENCE-NO" con- tent="textOnly" dt:type="string" /> <ElementType name="NAME" content="textOnly" dt:type="string" /> <ElementType name="CURR-BALANCE" con- tent="textOnly" dt:type="float" /> <ElementType name="CURR-TRANS-DATE" content="textOnly" dt:type="date" /> <ElementType name="MASTER-RECORD" con-</pre>



90 New Montgomery Street
 San Francisco
 California 94105
 Tel 415-543-1515
 Fax 415-543-6701
 www.forecross.com

```

tent="mixed">
  <element type="REFERENCE-NO" />
  <element type="NAME" />
  <element type="CURR-BALANCE" />
  <element type="CURR-TRANS-DATE" />
</ElementType>
<ElementType name="MASTER-FILE" content="eltOnly">
  <element type="MASTER-RECORD" />
</ElementType>
</Schema>

```

The "eXtensibility" in XML provides both its greatest benefit and its greatest potential weakness. The very flexibility that gives XML so much of its power to eliminate barriers to the exchange of data also provides the rope an organization can use to hang itself. It's not hard to imagine the confusion that would result from multiple groups, one using W3C, another using Microsoft, a third using a home-grown schema definition, and a fourth avoiding all use of schemas. Therefore, there does need to be a central group which establishes global standards to which all groups must adhere, and that these standards need to be in place as early as possible, preferably before any significant use of XML. At the same time, care must be taken to avoid establishing an XML Gestapo that inhibits productive use of XML.

While a large IT organization may find it helpful to have someone on staff with an in-depth knowledge of XML, many XML experts are computer science graduates with a strong theoretical understanding of XML. These gurus can be very productive if assigned to implementation groups where they will work directly with data definitions. However, establishing an ivory tower group with several gurus together at some distance from practical data usage may not return the desired results. People who know the data in depth should be well represented in the standards setting group as a helpful balance to those with in depth theoretical knowledge.

3.5 XML Validation Over The Life Cycle

XML usage will evolve with the applications using it over their life cycle. Let's consider that usage in more detail. We will have programs that will encode and decode complete files or database tables, as a batch task. These documents will persist for relatively long periods of time. We will have programs that encode or decode documents in a real-time data exchange mode, both with applications on the same platform and on different platforms, so that the documents exist only as a byte stream in memory and only for a few milliseconds.

Data elements in programs that will be involved in XML data exchange need to be matched to the global dictionary of valid data tags. Document encoding and decoding routines have to be written and maintained. Schemas have to be developed and maintained within the scope of the relevant schema definition schema. And, provision must be made to ensure that, as any of these elements change over time, all will be in concert and, if any elements do get out of coordination, the discrepancies will be detected and corrected before any harm can occur.

The most important capability of XML with regard to disciplined use is the schema validation procedure. XML documents that persist on disk for extended periods of time can be validated against their respective schema as a batch process. However, XML documents that are transient in nature also require validation. Therefore, programs that encode and decode these documents must include either a mechanism to capture and write the transient documents to a persistent disk file for subsequent validation, or there must be a real-time validation engine that can be called as needed. This validation must be present during all testing of new program versions before entering production. In addition, some percentage of production XML documents, up to 100% if possible, should receive validation. Documents that fail validation must have error handling logic defined, particularly as fields are strongly typed and data validity tests are centralized into XML schema.

In addition, some type of versioning procedure is needed to ensure that the validation is occurring as expected. Depending on the site configuration, validation may be by a universal routine or by a routine generated with customized logic for a particular schema. The latter will be much faster, but then it can get out of synchronization with revisions to the schema. In both cases, updates to the schema definition schema will require revalidation of the validators.

XML schema are a passive validation device, since validation occurs only when requested, and it can be bypassed. Most database systems have active validation, such that discrepancies between the current version of the database and older versions of programs using the database are detected at run time, and defined error procedures are executed. We recommend that XML validation should operate like active database validation. This can be implemented by a modest extension to the schema definition schema to create a version or a time stamp attribute for each schema that is defined to match an equivalent attribute in each XML docu-



FORECROSS®

90 New Montgomery Street
San Francisco
California 94105
Tel 415-543-1515
Fax 415-543-6701
www.forecross.com

ment. Each custom validation program will also have the version or time stamp compiled into its object code, so that a mismatch will be reported as an immediate error. In this way, normal XML facilities can be used to provide the active validation that is missing from the XML specification in the name of providing the greatest possible flexibility.

3.6 Retrofitting XML Into Legacy Applications

We see as much value in the use of XML in legacy applications on mainframes as in the use of XML in new applications and in distributed computing. We discussed above how implementing XML documents as the data exchange in CICS programs allows direct interface with XML aware browsers as well as the transformation of the programs into re-usable components. In addition to that, CALL interfaces and data exchange files can be encoded into XML, and report files can be rendered in XML for viewing in a variety of presentations with appropriate XSL stylesheets. Depending on the complexity of the files, customized batch utilities can provide the encoding and decoding, or the programs creating the files can have XML encoding and decoding logic added to them.

Again, the analogy to SQL is useful. The earliest database management systems used a CALL interface to retrieve and update their data, just as the earliest versions of CICS used macro commands. However, it proved to be more convenient to extend the COBOL language with a utility program that would generate the CALL commands and arguments for SQL or for CICS applications. To encode and decode XML documents, legacy programs could have fully expressed STRING and UNSTRING logic, use a CALL interface to a global or customized routine, or generate the necessary logic as needed by a utility program.

Regardless of which technique is used to perform encoding and decoding, we need to define data tags in a global repository where XML data typing and validation rules were stored as well. This global repository needs to tie together the XML schema specifications with the syntactical requirements of the legacy language data specifications. Ideally, this would be an extension to a data management tool such as an entity-relationship maintenance utility that could also provide version control.

3.7 Conclusion

The extent to which XML is proven to be a strong contributor to the organization's cost control and ability to respond to opportunities will be determined by the degree of discipline of its implemen-

tation. In particular, we strongly discourage non-trivial use of XML without active schema validation. A complete solution to disciplined XML usage will go further, creating a single point of maintenance with a repository of extended information beyond that supported by the implementation language, and providing operational control of both program and document versions.

¹ Using Oracle's XML Schema processor for Java Version 1.0.0, released 7/28/2000 from http://technet.oracle.com/tech/xml/xdk_java.html.

² Using MSXML 3.0, released 7/31/2000, from <http://msdn.microsoft.com/xml/default.asp>.
Second Generation Legacy to Web Strategies Via XML

Author Bio:

Don Estes is Director of Product Architecture for Forecross Corporation, a legacy to web and XML implementation solutions company. He is responsible for second-generation legacy to web products and services, including automated testing of transformed programs. Send your comments to Don_Estes@Forecross.com.

Earlier versions of this paper appeared in the May, 2000 issue of the Cutter IT Journal, and the September - December, 2000 issues of eAJ Journal.

Permission is granted for reproduction and distribution of this document provided it is complete, unmodified, and retains all identification including this statement, and provided that notification of recipient is sent to the above email address. All other reproduction and distribution is expressly forbidden.



90 New Montgomery Street
San Francisco
California 94105
Tel 415-543-1515
Fax 415-543-6701
www.forecross.com