

# **XML and Distributed Business-to-Business Intelligence**

## **Fusing XML and Java-based Expert Applications**

©2001 Earl Cox



1289 North Fordham Blvd. Suite A312  
Chapel Hill, NC 27517

(919) 678-0477  
[www.scianta.com](http://www.scianta.com)

*But all the world understands my language.*  
Franz Joseph Haydn (1732-1809)  
Reply to Mozart who advised him not to visit England  
because he could not speak the language.

## The Evolution of Intelligent B2B Applications

Intelligent applications in the latter part of the last century focused on the representation of knowledge in one of two primary forms: the rule-based paradigm starting the late 1970's by Bruce Buchanan and Edward Shortliffe with their development of *Mycin* and the artificial neural-network approach first popularized by David Rumelhart and James McClelland. The rule-based approach quickly became part of many mainstream AI systems marketed by companies like Aion and Artificial Intelligence Corp. Neural networks were likewise turned into commercial ventures by businessmen like Robert Hect-Nielsen (HNC, Inc) and Casey Klimasauskas (Neuralware, Inc.). In those days, of course, much effort was expended both justifying the technology and proving that artificial intelligence systems could, indeed, solve problems ordinarily solved by human beings. Expert and neural network systems were seen as stand-alone entities – receiving data from external sources, performing their analysis, and returning their recommendations (or classifications) with some degree of certainty. That knowledge management and knowledge encapsulation were fundamental components of intelligence rich applications was not generally understood. There were exceptions, of course, visionaries like Karl Wiig, president of the Knowledge Research Institute, published, in the 1980's a three volume *magnum opus* on Knowledge Management<sup>1</sup> that articulated a knowledge representation and deployment infrastructure that is still in advance of most of today's intelligent system architectures.

Today the effectiveness and utility of intelligent systems are no long in doubt. But modern distributed business application designers are rapidly realizing that integrated business intelligence and knowledge management are not by-products of their applications nor are they a distinct phases of the post-implementation analysis – rather they are intrinsic and important components of the application itself. The rapid pace of change in the Internet world coupled with the demands of incessant competition drives the coupling of computational intelligence, semantics, and business logic. No longer are expert systems – of any kind – distinct and stand alone programs that service the corporation's legacy applications. In understanding this new relationship between application and intelligence, designers have moved toward two related technologies: business policy rules and the Extensible Markup Language (XML). Together, these two capabilities provide a powerful, robust and extensible way of building distributed applications.

### The Extensible Markup Language (XML)

In many ways, XML is the glue that binds distributed applications. The Extensible Markup Language, like the more familiar Hypertext Markup Language (HTML), is an outgrowth of the SGML (Standard Generalized Markup Language, ISO8879). But unlike HTML, XML is designed to provide not only more reliable and cross-platform document formatting capabilities, but the ability to capture the semantics and structure of complex data aggregations. Thinking of XML in terms of document formatting,

---

<sup>1</sup> Vol. 1, *Knowledge Management Foundations: Thinking about Thinking -- How People and Organizations Create, Represent, and Use Knowledge*, Vol 2. *Knowledge Management: The Central Management Focus for Intelligent-Acting Organizations*, and Vol 3. *Knowledge Management Methods: Practical Approaches to Managing Knowledge*. Refer to <http://www.krii.com/index.htm> for Knowledge Research Institute's home page.



in fact, completely obscures its true power. As an extensible system for describing data, XML differs from HTML in several important ways:

- Knowledge engineers, business analysts, and application developers can define XML properties (attribute names) that correspond to the semantic properties of the application.
- Multiple applications can access XML attributes by name rather than position or some predefined tag. Thus XML defines a common language for the exchange of information through a dictionary of published semantics (see the Document Type Definition, below).
- XML documents (data associations) are defined by a Document Type Definition (DTD) which precisely describes the structure and configuration of the data. This insures uniformity of the data and provides a well-formed description of the data for validation and data sharing.
- Data structure definitions are nested and recursive. Highly complex data organizations can be defined using the XML language. Both simple and complex documents are accessed through a single Document Object Model (DOM) that allows easy navigation through the XML structure.

XML solves many problems associated with the definition, extraction, and use of dynamic, varying, and unstructured information. XML documents correspond to a Document Definition Type (or DTD) and define the logical properties, organization, and characteristics of some entity (or set of entities.)

## An XML Knowledge Base Representation

Within an intelligent application, XML is used to define the allowable syntax and meanings – through the document type definition (DTD) -- of such components as:

- The underlying knowledge base containing variables, fuzzy sets, and rules.
- The types, organization, and contents of System messages. Messages are .are often the “glue” that holds application facilities together as a coherent whole. The XML document type definition defines the way messages are formulated, organized, and transmitted.
- The content and organization of the system audit logs. These files are generated hourly by the integrated logging facility. The contents of the log are completely defined by the XML document definition.

Many other components of an intelligent system can be (and generally are) encoded using XML, but these are the entities that usually provide the largest return on investment. Using XML in an intelligent application generally involves two XML features – the Document Type Definition and the actual XML definition. Figure 1 illustrates part of the Document Type Definition (DTD) for a typical knowledge base (this is not a complete definition but is intended to give you a feel for the nature of a DTD file.)

```
<!ELEMENT Knowledge-Base (Variables, Fuzzysets, Rules)>
<!ATTLIST Knowledge-Base
    KBName CDATA #REQUIRED
>
<!ELEMENT Variables (Variable)+>
<!ELEMENT Variable (Description?)>
<!ATTLIST Variable
    VarName CDATA #REQUIRED
    VarDataType (STRING | INTEGER | BOLLEAN | FLOAT | DOUBLE) #REQUIRED
    VarAccessType (PUBLIC | PRIVATE) "PRIVATE"
>
<!ELEMENT Rules (Rule)+>
```

```
<!ELEMENT Rule (Option?, Condition?, Statement)>
<!ATTLIST Rule
  RuleID CDATA #REQUIRED
>
<!ELEMENT Option (Priority?, Weight?)>
<!ELEMENT Priority EMPTY>
<!ATTLIST Priority
  Value (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10) "5"
>
<!ELEMENT Weight EMPTY>
<!ATTLIST Weight
  Value CDATA "1"
>
<!ELEMENT ExposedVars (#PCDATA)>
<!ELEMENT Statement (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
```

**Figure 1.** The Document Type Definition for a Knowledge Base

Once the Document Type Definition has been created, knowledge bases are created using the attribute and property tags defined in the DTD. Figure 2 shows how a knowledge base is constructed using the previous Data Type Definition (again this only an brief example).

```
<Knowledge-Base>
  <variables>
    <variable>
      <varname> Price </varname>
      <vardatatype> double </vardatatype>
      <varrange_low> 10 </varrange_low>
      <varrange_high> 140 </varrange_high>
      <fuzzyset>
        <fzysetname> High </fzysetname>
        <fzysettype> Linear_Growth </fzysettype>
        <fzysetparms> 10,140 </fzysetparms>
      </fuzzyset>
      <fuzzyset>
        <fzysetname> Low </fzysetname>
        <fzysettype> Linear_decay </fzysettype>
        <fzysetparms> 10,140 </fzysetparms>
      </fuzzyset>
    </variable>
  </variables>
  <rules>
    <Rule>
      <rulename> R01 </rulename>
      <ruletext> our price must be High; </ruletext>
    </Rule>
  </rules>
</knowledge-Base>
```

**Figure 2.** An XML Encoded Knowledge Base

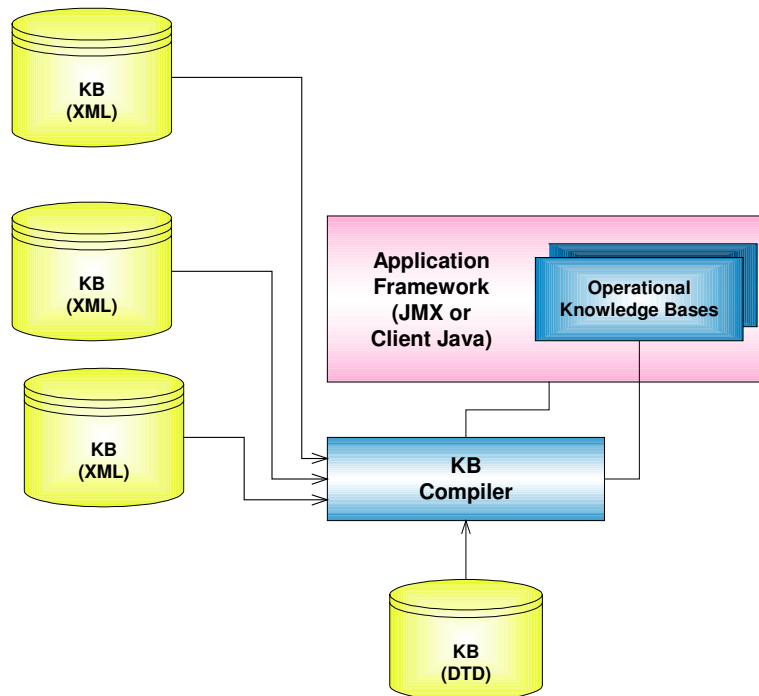
There is not room in this article to describe all the various properties of the XML or the DTD file<sup>2</sup>. However, you can see, by examining the Document Type Definition in Figure 1 that the file defines the attributes of the data, whether the attribute is required or optional (and, if optional, its default value), and the allowable or initial values for many of the attributes.

---

<sup>2</sup> For a complete description of XML see <http://www.w3.org/XML/>, this is the XML home page and provides access to the XML syntax. This page also takes you to the World Wide Web Consortium (W3C) which develops interoperable web-based technologies (and is responsible for the design and evolution of XML.)

## Connecting to an XML Knowledge Base

The knowledge base represents the core intelligence of the application. An application accesses this intelligence by compiling the knowledge base into an executable form – generally a collection of objects. This intelligence – often involving several knowledge bases – is tightly coupled to the application framework as illustrated in Figure 3.



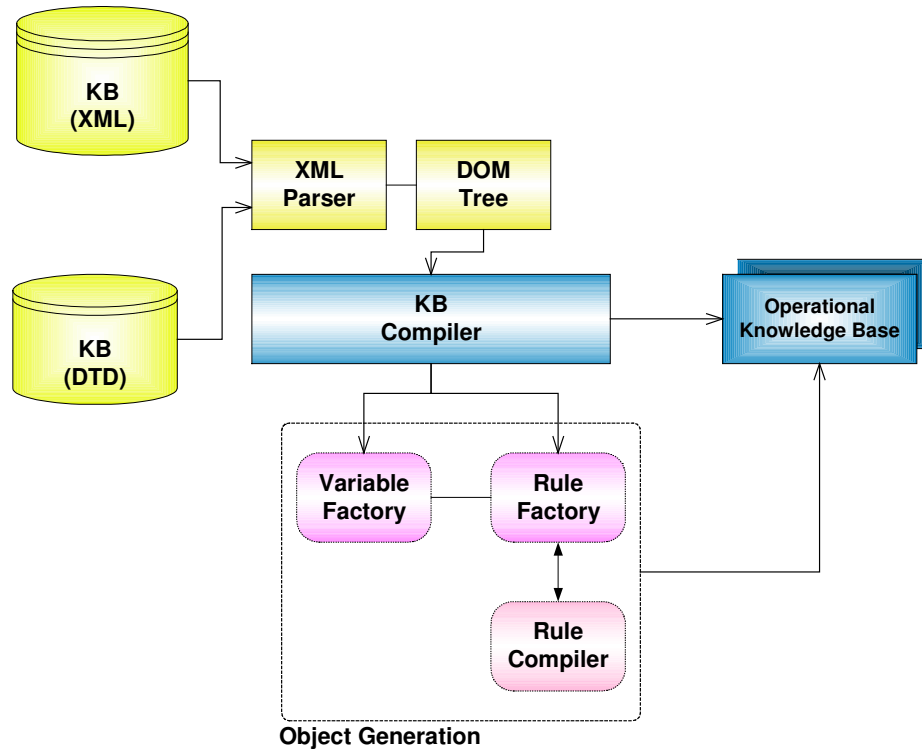
**Figure 3.** Coupling The Application to the Knowledge Base

In most cases the framework is a Java application (or servlet), but the recent introduction of Java Management Extensions (JMX<sup>3</sup>), provides a powerful and extensible framework for building intelligent system in a distributed environment. Inter-cooperating expert systems can be created at the agent level and shared among many application. Coupled with a uniform XML messaging capability, this means that many otherwise separate applications can be connected in order to incrementally build and deliver intelligent systems across the Internet or the corporate intranet.

<sup>3</sup> See <http://java.sun.com/products/JavaManagement/> the JMX homepage for a complete discussion of this cross-industry distributed management infrastructure.

## Connecting XML and a Java Application

Actually using and deploying an XML-based knowledge base is relatively easy. Figure 4 illustrates the process of generating an operational knowledge base – the Knowledge Base compiler uses an XML parser (a program that understands DTD and XML files<sup>4</sup>) to generate a syntax tree for the knowledge base and then use this tree to actually extract the various component parts from the XML file itself.



**Figure 4.** Creating an Operational Knowledge Base

How we access the contents of an XML data structure involves two related steps: parsing the XML file and navigating through the resulting output. This output is in the form of nodes arranged in a tree structure. The nodes and the tree structure comprise the Document Object Model (DOM). As the XML KB definition file is parsed (the attributes are extracted from the DOM), the elements are passed to object generator factories<sup>5</sup> to create the operational structures in the knowledge base.

<sup>4</sup> Free or shareware XML parsers are available at many sites on the Web. For some examples see <http://msdn.microsoft.com/xml/general/xmlparser.asp>, which is the Microsoft XML parser (version 3); also <http://www.jclark.com/xml/expat.html>, which is the XML parser tool kit from the Thai Open Source Software Center; and <http://www.alphaworks.ibm.com/tech/xml4j>, which is IBM's XML parser for Java.

<sup>5</sup> For a discussion of factories and other important issues in object-oriented design see Gamma E., Helm, R., Johnson R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1994). This book is absolutely required reading for all knowledge engineers as well as system architects.

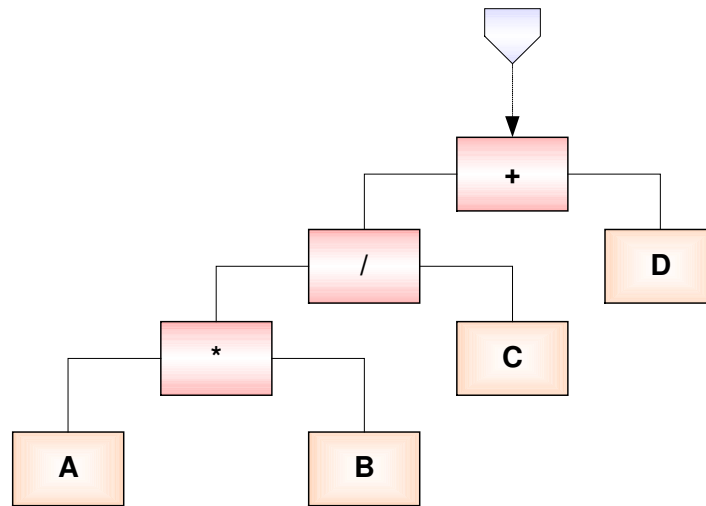
The Extensible Markup Language acts as a structured and well-formed repository of the knowledge base, but parsing the XML definition does not generate a working knowledge base. We still need a set of objects that contain the final representation of our variables, fuzzy sets, rules and other elements. In particular, the rules, stored in the KB as a text string, must be further parsed and compiled into an executable form -- such as a binary tree containing the if-then-else statements and expressions stored in an interpretable structure like early operator reverse polish (EORP). For those who are unfamiliar with reverse polish, it is a technique for transforming any expression into an unambiguous, parentheses-free representation that can be quickly evaluated. As an example, the expression,

$$( (A*B) / C ) + D$$

is represented in polish notation as,

$$AB*C/D+$$

To evaluate this expression we move right until an operator is found, The previous two terms are evaluated and the result replaces the operator and the terms. The expression then reprocessed until only the final value remains. Reverse polish expressions can be rapidly evaluated by converting them to binary trees and performing an ordered traversal through the tree. Figure 5 shows the expression as a binary tree.



**Figure 5.** A Reverse Polish Expression as a Binary Tree

A post order traversal of the tree properly evaluates this expression. The Java KB compiler traverses the DOM tree selecting each rule and compiling it into its executable form. Executing rules is then a matter to traversing the representation tree<sup>6</sup>.

This knowledge base, because it is parsed and compiled as part of the host application, provides a tight coupling between the application and the knowledge source. Since XML is the base language for our knowledge management facilities, a distributed application can easily take advantage of its transportability.

<sup>6</sup> You can find a complete data structure package in Java that includes algorithms for in-order, post-order and pre-order tree construction and traversal at <http://www.cs.fiu.edu/~weiss/dsj/code/DataStructures/>

Not only can the application load and reference an XML knowledge base, but it can access and interrogate knowledge bases as it would any XML file. This means, as an example, we can ask for all the variables in a knowledge base (using the simple XML parser discussed earlier) without actually compiling the entire knowledge base.

## The Future of XML Enabled Expert Systems

In this article we have seen how the Extensible Markup Language (XML) provides a common inter-disciplinary framework for describing, implementing and using sharable knowledge. While the integration of knowledge bases and distributed applications with XML provides one of the most visible benefits in using XML, we should not ignore the more important and subtle point – XML provides the infrastructure for creating, maintaining and extending intelligent web-based applications that need to share intelligence (or simply need to share data). Figure 6 illustrates how this architecture works.

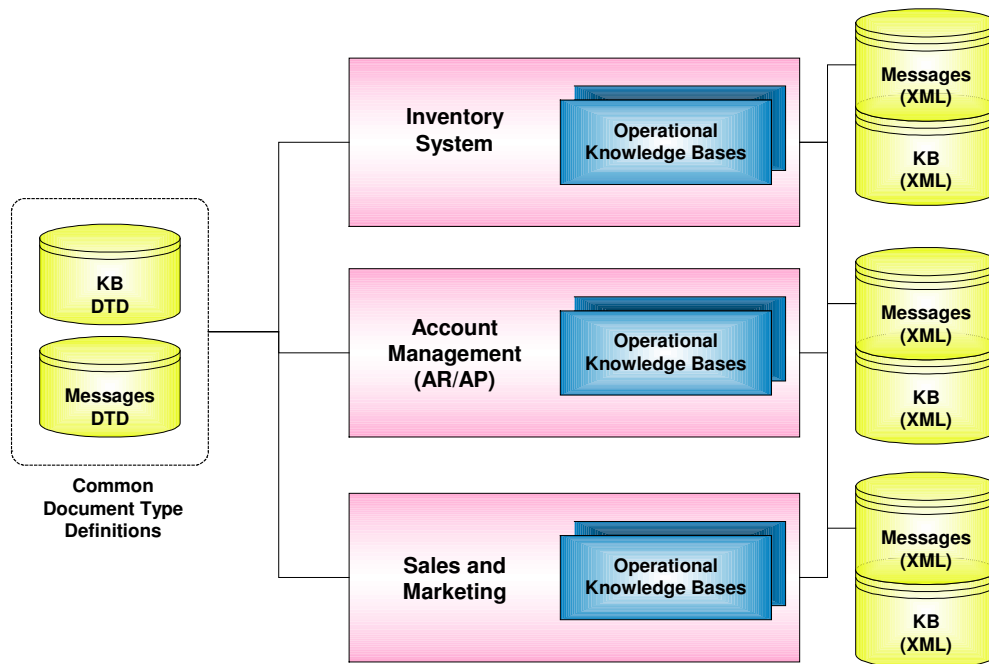


Figure 6. Multiple Applications Sharing XML objects

A central repository of Document Type Definitions for the sharable components (such as the knowledge base, messages, audit files, database query results, data dictionaries, etc) provides a flexible and extensible way of connecting applications. In the near future we will see XML playing a much larger role in the evolution of eBusiness. The description and tagging of transaction types, application and web server statistics, performance profiles (such as perfmon output), and the integration of XML into supply chain management will provide a common, uniform, and universal mechanism for sharing and exploiting data. Intelligent applications will begin to share knowledge across application boundaries as XML and Java provide a common framework for collecting knowledge and making decisions.

In the end, of course, the use of XML simplifies the encoding of data by providing a way of attaching a meaning to data elements. The Document Type Definition provides a way of telling us how these little pieces of meaning are assembled into larger structures and how these larger structures are



related to other larger structures. From naming each piece of data we derive a terrific benefit – understanding the semantics of data in its context. The idea of coupling external semantics with rule-based knowledge has long been one of the Holy Grails of knowledge engineers in building intelligent systems. XML is moving us one giant step forward.

\*\*\*\*\*

For readers interested in building XML-based applications or simply exploring the capabilities of XML, go to the following web site:

[http://www.garshol.priv.no/download/xmltools/cat\\_ix.html#SC\\_DOM](http://www.garshol.priv.no/download/xmltools/cat_ix.html#SC_DOM)

This site has tools for creating and modifying XML and DTD files, converting word processing and HTML files to XML, storing documents in an XML database, and parsing DTD and XML files. You can also find Application Program Interfaces for managing Document Object Models (DOMs). One of the best XML development environments (from the author's viewpoint) is XML Spy 3.5 (<http://www.xmlspy.com/>) a product that aspires to become a complete Integrated Development Environment for XML.

---

For more information or to schedule a presentation call (919) 678-0477 or visit [www.scianta.com](http://www.scianta.com)



©2004 Scianta Intelligence, LLC  
AR-PA-009