Headquarters
97 Libbey Industrial Parkway
Suite 300
Weymouth, MA 02189
Phone: 781-616-2100
Fax: 781-616-2121
Email: info@capv.com
www.capv.com

Europe
3rd Floor, Sceptre House
7-9 Castle Street
Luton, Bedfordshire,
United Kingdom LU1 3AJ
Phone: +44 1582 400120
Fax: +44 1582 411001
Email: euro.info@capv.com

Japan
Hiroo Office Building
1-3-18 Hiroo Shibuya-ku
Tokyo 150-0012 Japan
Phone: +81 3 5475 2663
Fax: +81 3 5475 2710
E-mail: yoshida@gsm.to
www.gsm.to

Dynamic Content Software Strategies Consulting Service

October 22, 2004

A Focus on XSLT 2.0: Understanding the Development and Business Benefits

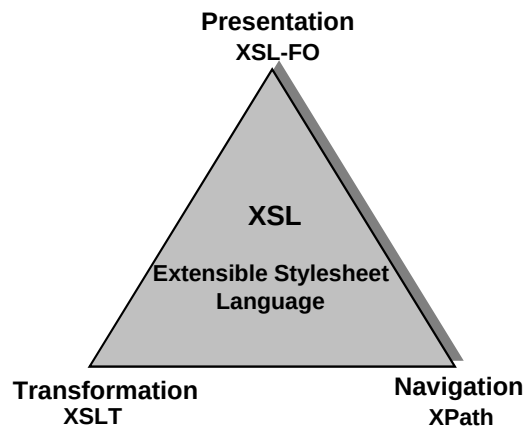
Introduction

There is little doubt that XML (Extensible Markup Language) usage is prevalent throughout a myriad of data and document-driven business processes. Everything from sales reporting and purchasing applications to content management and inventory controls systems are now often based on XML. As XML usage continues to expand, the importance of the World Wide Web Consortium's (W3C) entire family of XML standards will become more and more evident.

The evolution of XSLT (Extensible Stylesheet Language Transformations) in particular demonstrates the tangible benefits of the corporate investment in XML. Specifically, these benefits include developer productivity, data reusability and interoperability, and application versatility. As a component of XSL (Extensible Stylesheet Language), XSLT provides developers and applications with the ability to transform XML into other formats such as HTML, xHTML, and other XML vocabularies.

XSLT is a critical piece of the original premise of XML: to separate content from structure and presentation to provide data independence, flexibility, and interoperability. Together with XSL-FO (Extensible Stylesheet Language Formatting Objects) and XPath (XML Path Language), XSLT provides a powerful trio of languages that enable the transformation, navigation, and presentation of XML content.

Figure 1: The Components of XSL



This analysis discusses the power of XML transformation and navigation as it relates to benefits for developers and corporations. As such, it focuses on the most recent additions to XSLT as outlined in the W3C's XSLT 2.0 working draft (http://www.w3.org/TR/xslt20/) and the relationship of XSLT 2.0 with its sub-language, XPath 2.0 (http://www.w3.org/TR/xpath20/). As inter-dependent transformation and navigation languages, XSLT 2.0 and XPath 2.0 are the key components of functional programming for XML content.

XSLT 2.0 Fundamentals

XSLT is a tag-based scripting language that is designed specifically for XML-based transformations. An XSLT transformation describes the rules for transforming one or more instances of XML content into another format, or as described by the W3C, "the rules for transforming one or more source trees into one or more result trees." Created as an XSL stylesheet using XML syntax, the instructions define how an XSLT processor should read incoming XML content to output the desired results.

XSLT is inherently a declarative programming language. XSLT 2.0 does not change this foundation, nor does it attempt to make XSLT a general-purpose programming language. What it *does* bring to the table is the ability to use XSLT as a fully functional programming language via the application of higher-order functions such as string and number manipulation, strong data typing, and XPath 2.0 expressions. This is a pivotal evolution from its 1.0 predecessor, enabling developers to substantially reduce code complexity and streamline development while simultaneously increasing the capabilities of XML transformations.

XSLT 2.0 delivers a quantum increase in functionality through its intrinsic relationship with the W3C's XML Schema specification as well as the power of the XPath 2.0 sub-language. For example, full compatibility with XML Schema brings validation processing to XSLT 2.0 transformations, enabled through XSLT processors. Validation processes can apply to temporary and result trees during real-time processing as well as testing and debugging stages. XML Schema-awareness is an optional feature of the specification, enabling developers to take advantage of this capability when appropriate to the application.

XPath 2.0 is used by XSLT 2.0 to locate and process content within an XML document's logical hierarchy. The W3C's XSL and XQuery Working Groups jointly created XPath 2.0. It is important to understand that XQuery 1.0 is an extension of XPath 2.0, and the specifications are generated from a common source. Although XPath 2.0 is related to XSLT 2.0 and XQuery 1.0, XSLT is optimized for transforming and formatting XML content, while XQuery 1.0 is designed for extracting information from XML documents and databases.

XQuery's market adoption depends upon an investment from commercial database vendors, which is not yet complete. Oracle and IBM, for example, have announced intentions to support XQuery in their respective database products, but currently do not have production-ready implementations. Many other vendors are in the early stages of discussing XQuery as a replacement for or adjunct to established query languages, such as SQL.

XSLT 2.0 utilizes the powerful functionality of XPath 2.0 without requiring support from back-end technologies. In addition, the XSLT 2.0 Working Draft has remained fairly stable since late 2003, indicating that it has reached maturity and most likely will not change appreciably before becoming a final W3C recommendation. Hence, it can be put into practice today. The ability to locate and process content within an XML document is an essential component in designing and implementing XML-driven transformations. Together, XSLT 2.0 and XPath 2.0 are a fundamental part of the W3C vision for data independence, flexibility, and interoperability.

Understanding the Impact of XSLT 2.0

Due to the proliferation of XML data, documents, and vocabularies, the use of XSLT 1.0 is already widespread within XML-driven business processes such as electronic data exchange, transaction management, document and content management, and publishing. As developers tested the power of XSLT 1.0, however, they reported a number of shortfalls as well as capabilities that were tedious and unproductive as a result. Complaints about string handling, processing groups of related elements, and

single input and output documents were continuous in Web forums such as XSL-List, which is managed by Mulberry Technologies (http://www.mulberrytech.com/xsl/xsl-list/index.html).

Consequently, the development of XSLT 2.0 focused on known 1.0 shortcomings and on increasing the power of its XPath sub-language. As a result, the goals for XSLT 2.0 and XPath 2.0 were equivalent in a number of ways. Both languages would implement interoperability with related XML specifications such as XML Schema. Both languages would place identical emphasis on ease of use, internationalization, and backward compatibility. The 2.0 versions of both languages are expected to become W3C Recommendations by early 2005. Since they are both stable Last Call Working Drafts, however, vendors have announced 2.0-level support and developers have begun working with the specifications in earnest.

XSLT 2.0 and XPath 2.0 provide developers and corporations with a number of tangible benefits, ranging from improved productivity and quality during development to enhanced versatility upon implementation. The following subsections describe these business benefits.

Productivity and Quality

Achieving productivity in programming environments often focuses on writing fewer lines of code. When complex coding can be reduced substantially through the use of higher order functions to replace custom workarounds, the timesaving advantages can affect an entire application development lifecycle. The productivity that is realized during development also has substantial benefits for internal processes such as code re-use (i.e. designing common use cases) and new hire training (i.e. producing maintainable code that is concise and easily understood).

Achieving productivity in terms of development speed is obviously commendable, but certainly does not guarantee quality. Quality code is often characterized by its modularity, portability, maintainability, ability to support high-performance applications, and perhaps most importantly, its ability to produce few errors during testing and debugging.

XSLT 2.0 brings significant benefits for those striving toward productivity and quality. For example, XSLT 2.0 provides a therapeutic solution for those who have written complex coding workarounds for grouping (i.e. the "Muenchian method") or string and number manipulation. 2.0 extensions such as <xsl:for-each-group>, <xsl:group-by>, and <xsl:group-adjacent> work hand in hand with functions such as current-group( ) and current-grouping-key( ) that are accessible from within XPath 2.0 expressions.

XSLT 2.0's support for strong data typing is another example of its ability to increase developer productivity and code quality. Historically, the advantages and disadvantages of strong data typing have been heatedly discussed in forums dedicated to best practices in development environments. Some extol its virtues, while others are concerned with the performance impact of strictly enforcing type rules.

In the case of XSLT 2.0, strong data typing support can increase the stability of transformations focused on electronic data exchange and remove the need to develop error-checking code. By reducing the number of bugs in these transformations, strong data typing support should also increase the quality and reliability of the exchange. It is worth noting, however, that one should choose an XSLT processor with care, as the power and design of the processor drives the level of performance associated with electronic data exchange.

XSLT 2.0's productivity and quality benefits also include:

- The ability to remove and avoid proprietary vendor extensions that addressed 1.0 shortcomings.
- Backward-compatibility with XSLT 1.0.
- String and number manipulation with XPath functions such as compare( ), replace( ), string-join( ), tokenize ( ), and translate( ).
- Regular expression support, bringing familiar programming constructs to XSLT development as well as providing more powerful validity-checking opportunities.

Reusability

Reusability is often described as the Holy Grail for data and document-driven applications. In fact, the ability to implement and efficiently *use* a single-sourcing strategy is perhaps one of the best indicators of a high-performance workgroup. XSLT 2.0's support for user-defined functions is a strong example of reusability opportunities.

The ability of <xsl:function> to receive parameters, return values, and be called from XPath expressions makes this support even more powerful. The impact spans data and document-driven transformations by enabling the use of practices such as common code libraries and single-sourcing content strategies that isolate repeatable content such as company logos, header and footer information, and disclaimer text. XSLT processors that are compliant with 2.0 should alleviate a dependence on vendor-driven extensions by supporting user-driven functions that are portable across multiple products.

XSLT 2.0 support for the concept of transclusion is another example of its reusability advantages. The ability to dynamically include data from another source through functions such as doc( ) is bolstered by the 2.0 support for XML data as well as unparsed text. XSLT 2.0 also enables data extracts to be stored in variables and parameters, allowing for the reuse of the extract as needed. Efficient use of this feature can support single-source strategies for data and document applications.

XSLT 2.0's reusability benefits also include:

- Support for multiple output processing from a single XSLT transformation, enabling the production of multiple instances of XML, HTML, xHTML, or text outputs. This is a huge advantage over the one output per process capabilities of XSLT 1.0.
- Support for grouping, allowing the collection and manipulation of related data.
- Support for aggregation, allowing mathematical functions such as sum( ), avg( ), and count( ) to be applied to grouped data. The combination of grouping and aggregation support is extremely beneficial for applications such as report generation and table generation for structured documents.

Interoperability and Compatibility

As described in the section on productivity and quality, the advantages of strong data typing can be profound. This capability is directly related to XSLT 2.0's support for XML Schema, which exposes 44 data types to the XSLT 2.0 development environment. Beyond productivity, however, lie the advantages of interoperability available to XML-driven electronic data exchange projects as a direct result of XSLT 2.0's support for XML Schema. In fact, InfoTrends/CAP Ventures believes that XSLT will play a larger role in enterprise application integration strategies due to the applicability of 2.0 extensions, the processing power of XPath expressions, and XML Schema interoperability.

A prime example of increasing XML Schema usage comes from the financial services and accounting industries. XBRL (Extensible Business Reporting Language), an open standard for the electronic communication of business and financial data, is becoming the model for electronic data exchange and reporting in these industries.

XBRL has had a global impact. In fact, the European Commission mandates the use of XBRL for corporate tax reporting beginning in 2006. It has also invested 1 million EURO in efforts to promote awareness of XBRL and speed adoption (http://xbrl.edgar-online.com/x/newsletter/091504.asp). The U.S. SEC (Securities Exchange Commission) is evaluating XBRL in calendar year 2004 for voluntary supplemental reporting (http://www.sec.gov/news/press/2004-97.htm).

The use of XML Schema vocabularies like XBRL is increasing across a wide range of vertical industries. Examples include MODA-ML from the manufacturing industry, MathML from the scientific community, and ACORD XML from the insurance industry. Consequently, the interoperability benefits of XSLT 2.0 and XPath 2.0 have the potential to encourage and enable the use of vocabularies across industries as well as within them. A feasible example would be the availability of industry-specific stylesheets for rendering and transformation that developers reuse "as is" or assemble into larger processing tasks according to business process requirements.

XSLT 2.0's interoperability benefits also include:

- Support for multiple input processing, including non-XML formats such as HTML, xHTML, and delimited files.
- Improved internationalization support, including support for string sorting and comparison. The use of XPath 2.0 expressions also enables localized formatting of numbers and dates.

Versatility

As discussed previously, XSLT 2.0's introduction of user-defined functions can have a profound effect on reusability. This feature also demonstrates XSLT 2.0's versatility. Another indicator of versatility is the introduction of "temporary trees," which are already garnering a positive response from developers.

This concept replaces XSLT 1.0's use of result tree fragments, whose benefits were often contested. For example, information contained within XSLT 1.0 variables could not be fully accessed or utilized with XPath expressions in every case, severely limiting any manipulation of variable data. According to Jeni Tennison, an invited expert in the W3C XSL Working Group, the ability to create a temporary tree that is available for further processing is useful to:

- Break up a complex transformation into several steps, usually by filtering, sorting, or annotating nodes during an initial pass so that the latter transformation is easier to complete.
- Create lookup tables to translate from codes or numbers to labels or vice versa, as you would create an array or matrix in another programming language.
- Iteratively process a document until a certain constraint is true.

XSLT 2.0's versatility is also demonstrated by:

- Independent processing capabilities, removing the need for back-end database support.
- Support for conditional processing, enabled by strong data typing, regular expressions, and filtering support.
- Support for complex text and data manipulation, enabled by XPath functions such as tokenize( ) and XPath 2.0’s support for FOR loops and IF statements.

InfoTrends/CAP Ventures Perspective

InfoTrends/CAP Ventures believes that the 2.0 versions of XSLT and XPath will increase developer and corporate usage due to their unified efforts to balance the requirements of structured data and document environments.

For example, XSLT introduces support for strong data typing, string and number manipulation, and XML Schema validation. These capabilities should increase the power and reliability of data-centric business processes such as electronic data exchange and application integration. In addition, XSLT 2.0 introduces support for multiple output and input processing, grouping, and improved internationalization functionality. These capabilities should increase the versatility and interoperability of document-centric business processes such as multi-channel publishing, compound document assembly, and reporting.

Equally important is XSLT and XPath 2.0's focus on improving well-known 1.0 deficiencies, some of which were overcome through customizations in commercial XSLT processors. The effect of customization, however, was often a subservience to a specific vendor’s processor, regardless of the availability of competing products. Therefore, XSLT 1.0 developers experienced frustration in being unable to re-use transformations across multiple XSLT processors. InfoTrends/CAP Ventures believes that the 2.0 versions of XSLT and XPath may "level the playing field" by removing the incentive for

incompatible functionality differences, causing market differentiation to focus more on performance, portability, and interoperability.

MarketWatch: Altova XML Suite Version 2005

With 1.5 million users, Altova is a preeminent player in the XML development space. A long-standing member of the W3C, Altova has deep roots in serving the requirements of XML developers and has been internationally recognized for award-winning software (most notably its XSLT processor). The company's product portfolio addresses the full spectrum of developers' needs.

The products within version 2005 of the Altova XML Suite cover every aspect of XSLT 2.0 and XPath 2.0 work, including editing and debugging/testing in Altova XMLSpy 2005, visual stylesheet design in Altova StyleVision 2005, and data mapping in Altova MapForce 2005. According to Altova executives, the company is demonstrating its leadership as the first to announce a native, production-quality implementation of XSLT 2.0 and XPath 2.0 throughout its entire product line.

The company's portfolio includes four products that enable rapid developer productivity:

- **XMLSpy 2005:** XML development environment for modeling, editing, debugging, and transforming XML-related technologies such as XML Schema, XSLT, XQuery, and Web services. Generates runtime code in multiple programming languages, including Java, C#, and C++.

- **MapForce 2005:** Graphical data mapping tool that can map any combination of XML, database, flat file, and EDI to XML, databases and/or flat files for data integration projects. Automatically generates transformations and programming code in multiple output languages including XSLT 1.0/2.0, XQuery, Java, C++, and C#.

- **StyleVision 2005:** Visual design interface for creating electronic forms, database reports, and stylesheets for transforming XML and database data into multiple output formats. Simultaneously produces XSLT 1.0/2.0 and XSL:FO stylesheets, as well as output in HTML, PDF, Word/RTF, and Authentic Forms, a business format for Web-based data entry.

- **Authentic 2005:** A free content editor for business users to input data directly into XML documents and databases. Provides templates for industry standard XML vocabularies such as DocBook, News Industry Text Format (NITF), and News Markup Language (NewsML). It is well suited for use as the data entry element of advanced XML-based document frameworks and business management-oriented database.