XML Based Content Website Management

Dustin Swallow

April 10, 2003
1. Introduction

The ever expanding number of pages on the World Wide Web continues to grow at a phenomenal rate each year. This exponential rate can be attributed to growth in every continent around the globe. Consequently, businesses are continually expanding their mission objectives to provide content and services to customers which are both effective and cost efficient. In this pursuit, some companies are looking to utilize the many benefits of XML in publishing content on the Internet and other sources. XML file repositories can provide a platform independent modular system which when utilized by other publishing technologies can provide the desired benefits of XML. Centralized independent data sources can be processed and displayed in appropriate outputs in many different formats. Companies can manipulate stored data to be used in a distributed environment, within applications, or simple HTML output for the Web. Utilizing different technologies also enable companies to display the same centrally stored information in domain specific formats such as HTML, PDF, or WML.

To begin this discussion, an overview of XML will be given. The discussion will include a brief history of XML, the Pros and Cons of using XML, XML functions, XML processes, and the general features of XML which make it so appealing. The next section will discuss the justification of using XML for content generation, including some driving forces. The next section will highlight three different strategies for generating content from XML using Docbook, AxKit, and Python. Then, I will discuss in detail the implementation of a HTML website using Docbook source documents. Finally, this discussion will finish with some comparison and future implications related to these strategies and XML.
2. Overview XML

XML is an open standard electronic data storage language. For this paper, its purpose is for electronic web publishing and data exchange. Here is a simple definition of XML. (Extensible Markup Language (XML), 2003)

**eXtensible Markup Language (XML)** is defined as simple, flexible text format derived from SGML designed to meet the challenges of large-scale electronic publishing.

Consequently, XML has also played an important role in the exchange of data on the Web. The XML standard is regulated by the World Wide Web Consortium located at http://www.w3c.org/XML/. This committee defines a protocol and standard which all XML documents are required to comply. The features, and subsequent capabilities built upon this standard have contributed to the growth and world wide acceptance of XML.

In this overview, I will highlight a brief history of XML, the Pros and Cons of XML, functions of XML, XML processes, and the general features of XML.

2-1. History of XML

XML has emerged as a markup language based on an important concept, the separation of content from markup. Content defined as the material or information which you wish to present, and markup defined as how you wish to display this material. This concept was first presented in September, 1967 by William Tunnicliffe of Graphic Communications Association (GCA) in making a presentation at the Canadian Government Printing Office. This is considered the beginning of the evolution of generic coding, which later formed the basis for XML. (McKinnon, 2003) In 1973, IBM released Generalized Markup Language (GML). The purpose of GML was to create a language with a set of markup instructions, which developers could use on their separate
applications. Markup was performed on each tag name depending on its type (headings, paragraphs, styles). The actual markup was performed by other processing instructions. This formed the concept for generic encoding. The potential for independent processing and presentation soon became apparent, but for it to be truly transportable and modular, standards needed to be developed. This standard was first published and recognized in 1986 as Standard Generalized Markup Language (SGML). It is still recognized today as the overarching international standard metalanguage. It can be viewed as the Grandfather for current languages like HTML, XML, and WML. SGML looks and feels like many web language tools today. SGML separates information in the document into small pieces, which are defined as elements. Each element type can be formatted, sorted, or searched in a standardized way. SGML also introduced the concept of document types, which are formally defined using Document Type Definitions (DTDs). DTDs identify all the elements occurring in a SGML document and describe the structural relationship between these elements. DTDs are used extensively in XML to define constraints and relationships of elements within XML documents.

2-2. Pros and Cons of XML

XML offers many benefits from data communication to information storage and retrieval. XML provides an easy and cheap way to store data. The XML standard has been developed to provide an easy way to store and structure data. Thus solutions can be developed fast and cheap with little long term maintenance. One of XML’s main benefits is its ability to be customized or tailored to fit any requirement. Domain specific content can be adjusted for any industry or environment through the use of Data Type Definitions (DTDs) or XML Schemas. These tools are used to validate data stored within XML
documents with regard to specific rules stated with the DTD or schema. Document structure can then be modeled to resemble data specification. New specifications like BizTalk, are being developed to enable companies across industries to standardize content and maintain customization capabilities. (Kiely, 1999) Thus, XML can provide both a standardized format for storing information and also the option for customization through extensibility.

Another benefit of XML is portability. XML is an open standard which can be tailored to many different platforms. This factor makes a single data source portable, and capable of being utilized within many different frameworks. Applications developed in different operating systems are being developed to utilize XML documents. XML’s non-proprietary standard has enabled XML’s hierarchical structure to be available for browsers and other tools. Currently, almost every operating system includes XML parsers. This benefit is one of the most important facets with regard to using XML content to generate output in many different formats. Utilizing information stored in XML format, output is generated for websites.

XML does possess several disadvantages. A security concern which has attracted attention recently, is the possibility of unauthorized users utilizing XML’s semantic structure. XML tags are designed to describe and identify element data and attributes. Businesses can use data encryption, VPN tunneling, and public/private key encryption to protect information in transit over the Internet. The security concern is over what happens when unauthorized users gain access inside a companies network, and attains access to data stored in the XML format. The information contained within these documents can be analyzed by examining tag names and associated attributes. For instance, an unauthorized
user could search for tags that contain “<cre” which could be used to identify possible credit card numbers. Similar strategies could look for social security numbers and bank account numbers. Finding sensitive data stored in XML format is only possible because of the semantical structure associated with storing data in XML. Some companies such as Forum Systems Inc. have created encryption engines to overcome this possible security liability. Forum’s encryption engine targets information stored in XML formats as it travels from application to application and encrypts specific designated tags. Security analysts can determine which tags contain sensitive information and may choose to encrypt such data. With relation to XML content generated information, analysts will have to choose what information is sensitive and any associated costs. (Kay, 2002)

Another possible limitation is integrating with former data interchange systems. One example is businesses who have heavily invested in EDI (Electronic Data Interchange) networks. Companies who have heavily invested in such networks are less willing to invest in XML document data sources. Several adapter packages exist which recast EDI messages into XML tags and vice versa. However, some of the benefits of XML, including dynamic linking, powerful modeling, and hierarchical features, are severely limited. This limitation is only relevant with respect to companies with existing EDI networks who choose to efficiently trim costs and continue to use those networks. (Borck, 2000)

2-3. Functions of XML

What functions does XML provide? Currently XML constitutes two percent of network traffic. Estimated projections show that by 2006, 25 percent of network traffic could be passed using XML. (Greenfield, 2003) XML functions as a data
communications media. Companies are using content stored in XML format increasingly in business data communications. Vendors such as Microsoft are incorporating XML based applications and XML support tools in new products. Manufacturers such as Intel are including Simple Object Access Protocol (SOAP) stacks into all new products to utilize XML capabilities. SOAP is a network protocol to enable usage of business objects across hosts on a network. In a similar decision, Microsoft has built SOAP into its Windows products to meet the growing demand for interoperability with other network devices. Such industry trends indicate the strength and growth of XML based network traffic that businesses will initiate. With this projected increase growth of XML web traffic, it is only logical to find solutions to integrate existing content into XML documents for storage and management.

2-4. XML in Business Processes

XML is being integrated as companies adjust their business processes. Industries with automated business processes, such as health-care, banking, and manufacturing companies are starting to utilize the potential of XML. Many of these companies are choosing to use systems like Intalio Inc’s Business Progress Management System. Intalio Inc. and Fuego Inc. have created their system within a XML architecture. Their system uses XML schemas to help manage businesses processes along the supply chain management for their clients. This system supports the automation step by step through process integration. (Ferguson, 2003)

Other industries are also seeing the value of using XML to enhance business processes. American Suzuki Motor Corp has used XML to standardize its business processes with vendors. Their XML infrastructure eliminates proprietary systems and
customized applications between different vendors. (Thibodeau, 2002) A reduction in complexity and reuse of modular applications enabled Suzuki to trim overhead and cut costs. These are some of the major factors which have propelled interest in using XML to store content. Storing critical information used in exchange with vendors has become one strategic objective in supply chain management and extranets. Providing information in different formats with vendors using XML is an important area where costs can be trimmed to provide more efficient processes. Thus, utilizing XML website generated content for business extranets have the potential to improve processes and save money.

2-5. General features of XML

Now that XML has been analyzed with regard to its advantages/disadvantages, functions of XML, and usage in business processes; we will highlight some general features of XML which will be expanded upon in the different approaches to generating output from XML content. There are three main features of XML documents.

1.) Displaying information contained with tags, using different fonts and styles for different types of information. Content stored as text within XML tags can be used as simple output. A common example is to take this information and display the text on a webpage marked up appropriately with the desired font, style, and color changes. The separation of content from markup is a core component of XML. This enables designers to create new layout and designs using stylesheets, such as XSL (eXtensible Stylesheet Language) and CSS (Cascading StyleSheets). These stylesheets (the markup) are used to alter the display or rendering of the XML file (the content).

2.) Processing information contained within tags, which is used to extract and transform data based on semantic value and structure. This process utilizes the hierarchical
structure of a XML document. Information stored in a XML document is stored in
tags or nodes which are arranged in a hierarchical fashion. Every XML document
must have one root tag or node. To process information stored within tags, an
application traverses the tree structure of a XML document to find specific tags and
their associated information. The text stored within a tag is then extracted and
transformed into its appropriate form. An example could involve extracting specific
customer information and placing it into an order form which the customer used to
buy products on the web.

3.) Searching capabilities for traversing the hierarchal structure of XML documents.

Searching information stored within a XML document involves using language
techniques like XPath and XQuery. Searching capabilities coincide with the process
of extracting and transforming information within XML documents. As noted in the
previous section, searching can be used to process information, or to perform simple
queries. (Ray, 2001)

3. Website Generation Using XML Content

3-1. Justification and Driving Forces

The main justification behind using XML to generate website content is
flexibility. Storing data in XML format gives designers and developers the capability to
change this data into any format for display. The benefits of using XML content for
website generation dates back to 1996. During this time the Wall Street Journal
Interactive Edition was launched to meet the needs for news content reuse. Their mission
was to provide the richest business and financial news web site in the world. The need for
a system that could republish news content on different platforms and domains became
evident. This need for increased functionality caused them to choose XML for their storage of articles. Developers for the Wall Street Journal created a custom application to take news articles created in Microsoft Word, and transform them into XML documents with attributes specifying specific markup. To note, these XML documents had minimal markup other than font style, size, and weight. Developers would use a hierarchical system to design the interactive journal based upon edition sections, headlines, and articles. Each article was assigned a priority of its importance in specific editions and associated within particular sections. Thus, each article could be established as a separate entity. Consequently, the Interactive Journal web site designers could create custom configurations files for each platform, domain, and device specific environment. Each configuration files would reuse each article stored as a separate XML file. The use of these modular XML files provided the important flexibility which the Wall Street Journal needed to achieve its mission. (Karben, 1999)

**4. Generation Approaches**

This section will explore three strategies for use of XML based content to generate web sites. The three approaches investigated are Docbook, AxKit Toolkit, and using XML with Python.

4-1. Docbook

Docbook documents are essentially XML files which conform to the Docbook DTD (Document Type Definition). Docbook documents store content in a presentation neutral form which captures the logic and structure of your information. Docbook was designed primarily for books and papers on computer hardware and software; however, as you will see, it is not limited to this form. Docbook was created in 1991 as a joint
project of HaL Computer Systems and O’Reilly. Its popularity grew and a maintenance
group, the Davenport Group, managed the Docbook standard. In 1998, OASIS (the
Organization for the Advancement of Structured Standards) took over, and still manages
the Docbook DTD today. (Walsh, 1999)

The value of Docbook with regard to website publishing can be mainly attributed
to one individual, Norman Walsh. Walsh has served as the chair of the Docbook
Technical Committee and maintains the http://www.docbook.org/ website. Walsh is one
of the main authors of the Docbook XSL stylesheets, an open source project located at
SourceForge. This Docbook Open Repository has made a set of website stylesheets
available to developers wishing to process Docbook documents for web display. These
stylesheets provide an excellent tool for publishing content on the Web or in other
formats, without detailed knowledge of XSL. Additional control over output is available,
which requires creating customized XSL stylesheets for enhanced display and specific
markup. The purpose of these stylesheets is to process Docbook documents with an
XSLT (eXtensible Stylesheet Language Transformation) processing engine to produce
HTML output or printed output such as PDF files. (Stayton, 2003) Here is a process
diagram to illustrate.

----------------------------
Insert Figure 2 Here
----------------------------

Using Docbook documents as content has its advantages and disadvantages.
Thanks to the Docbook Open Repository and its readily available collection of
stylesheets, Docbook authors have an easy, simple, and quick environment to publish
articles and other documentation as was its original design. Designers need minimal
XML and XSL understanding to begin publishing articles right away. Tutorials and step by step instructions for setting up your authoring environment are available from many different sources. Once you become familiar with the stylesheets and can write your own customized stylesheets, you can gain a better grasp on how to design the layout of your articles, and how to mark up content on web pages. These XSL stylesheets give web page developers the option of using other external stylesheets such as CSS to render displays for Web browsers.

Some limitations do exist for Docbook and they are inherent upon its design. Docbook works great for articles and documentation, but tends to become unwieldy when using it to design websites with graphics, frames, and interactive web page activities using scripting. Thus the Docbook technology is used only for maintaining static websites. Designing a website with more interactive design requires a deeper knowledge of XSL, a steep learning curve in using Docbook elements, and requires more alteration of the existing pre-packaged XSL stylesheets. Trying to markup Docbook content and gaining a better control of web page elements would require significantly more time and energy when using this method.

4-2. AxKit

AxKit is an application and document server that uses XML processing pipelines to generate and process content and deliver it to clients in a wide variety of formats. AxKit is often termed the XML Application Server for Apache. (Slaymaker, 2002) AxKit uses a pipelining technique whereas a source document passes through a series of transformations by stylesheets. The resulting transformed document is then cached within the web server and provided for clients upon request.
Stylesheets defined in this context, simply alter the appearance, format, or structure of a document. In this setting, stylesheets are not merely limited to screen display rendering. AxKit is written within the mod_perl framework for the purpose of server side transactions occurring within the XML Application Server. AxKit is designed to run on the UNIX, LINUX, FreeBSD, or other scalable non-Windows platforms. AxKit works in parallel with the Apache Server application or any other application server running on the platform such as Cocoon, Zope, or Enhydra. (Sergeant, 2000a)

During the transformation process, AxKit takes a XML source document, and looks for XSL stylesheet processing instruction. *(Note: AxKit is not limited to only XML as source documents, but can process others such as articles, data sets, database returned results, output from scripts, etc…) AxKit then locates the corresponding stylesheet and performs the necessary transformations, and passes control on to further processing, or delivers the resulting document into the cache manager. Subsequent client web requests are handled by the cache manager. An important capability of the cache manager is capability to process client requests and deliver display formats appropriate to the client. Such display formats include HTML, XML, WML, XHTML, PDF, plain text, and etc.

AxKit provides several important advantages to using XML server side processing. AxKit supports a wide array of stylesheet processing techniques and languages. Developers have the flexibility to choose XSLT processing, XSP (eXtensible Server Pages), XSL, XPathScripts, or many other options. AxKit provides the rigidity
and robustness of using XSLT techniques, flexible solutions like XPathScripts which incorporate Perl scripts located within the stylesheet, and custom components like XSPs. XSPs provide a flexible tool for generating XML content from cookies, relational databases, and form parameters. (Sergeant, 2000b) This flexibility gives developers more control over transformation, especially in delivering dynamic content. AxKit provides seamless caching, data compression, and character set conversion. (Slaymaker, 2002) AxKit provides a centralized document management environment built upon a highly scalable platform. One of AxKit’s main advantages is its capability to manage a distributed website environment for XML generated content. Because of AxKit's pre-complied modules, and mod_perl environment, it can process client requests at approximately 80 percent of Apache Server’s speed. (Sergeant, 2000b) With consideration for the amount of time required to process XSL transformations, this is a very efficient operation. Additionally, AxKit clearly supports XML’s biggest attraction, the separation of content from presentation.

There are a few considerations when deciding to implement AxKit. The AxKit application server is an open source standard, and has easy instructions for installation and setup. However, AxKit users are required to write customized stylesheets for each source document during transformation. Centralized templates stylesheets can and should be developed to manage these changes. These stylesheets require developers to possess a higher in depth knowledge of XSL, XPath, and related XML language technologies.

4-3. XML and Python

Python is an interpreted, interactive, object-oriented programming language. Python is an open standard language which incorporates features of both scripting
languages and object-oriented languages. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. Python provides interfaces to many system calls, libraries, and windows systems. Python implementation is highly portable and can run on most platforms. (What is Python?, 2003)

Changes in XML support within Python, including XML toolkits, has made Python an ideal programming language to process XML documents. Python provides XML processing and network programming capabilities associated with object orient languages. Yet Python also provides text processing capabilities and operating systems interaction like scripting languages. (Jones, 2001) Python, classified as a scripting language provides benefits from both worlds. It is for this reason that Python is a perfect fit for using XML content to generate website output. Python provides extensive manipulating power and searching, extracting, and transforming tools required to process XML source documents. Python possesses several advantages in use as a web programming language; source code that is easy to maintain, incredibly portable code, and powerful object oriented features. (Jones, 2001)

There exist several Python tools for working with XML documents. Three packages exist with similar functionality, but specific focuses. These packages are the Python standard library, PyXML, and 4Suite by Fourthought, Inc. Each of these packages provides XML support tools through the use of SAX, DOM, and XSLT. Using Python and XML allows developers to work with the SAX interface, a simple API for XML. The SAX interface is a Java oriented interface adapted for languages such as Python to design a set of interfaces which permits Python to parse XML documents. (Fayolle, 2002) These
interfacing capabilities provide the extracting and transformation capabilities necessary to process XML documents even in a distributed environment. The one tool, which is of specific interest to XML content generated websites is the HTMLgen tool. This simple tool used in conjunction with Python files, enables developers the opportunity to process XML documents for content and provide a nice clean, simple interface to output HTML files. Specific Python files can be customized to provide the necessary processing of XML content. Display specification, and dynamic rendering of content is also possible within these Python files to provide the energetic non-static content that web surfers desire. Even the capability of writing applets for output is possible using Jython (a similar language incorporating Java & Python). (Lutz, 2001)

Python has been used with growing interest, especially among open standard projects and repositories such as Gentoo. Gentoo Linux is a versatile, fast Linux distribution geared towards power users. Website developers at Gentoo have used XML, XSLT, and Python to implement a XML content based website. Developers use Python in conjunction with XML to produce changing content and articles for their website. Python scripts are designed to detect changes made to Gentoo Linux software files. These detections are used in conjunction with log files to update Gentoo developers about changes made to the repository’s files. Here is a process flow diagram to illustrate how Python is used.

----------------------------
Insert Figure 3 Here
----------------------------
Python code utilized with the cron command searches the repositories files, and evaluates file modifications based upon dates. The Python code maintains information about changed files and produces a log of information including date, author, comments, and filename. Once the code is finished, the output is in XML format just like other source files for Gentoo. These files are then processed using XSLT and displayed on Gentoo’s website as a daily log change. Python also provides the facility to email changes to files to developers if they choose so. Daily dynamic processing using Python is used to display current updates to Gentoo’s repository to facilitate project management. (Robbins, 2001)

There are a few limitations associated with using Python. One limitation is associated with the decisive trade off of choosing Python over C++ or Java. Using Python provides additional robustness over a loss in performance. One additional consideration, when deciding to use Python involves learning a new language. The steeper learning curve associated with learning syntax of a new language may cause this strategy to progress slower than using another strategies.

5. Implementing a Docbook Content Based Website

Based upon the previous methods for XML content based website management, I choose to implement Miami University’s School of Business & Administration website using Docbook. In this section, I will discuss the process and steps I used to setup the system, writing XML source files, discussing Docbook stylesheets, XSLT processing engine, and customizing web pages with CSS and XSL. Finally I will conclude with the drawbacks and advantages for using Docbook in other websites.
To begin my project, I obtained a copy of the Docbook repository file system. This file system was a directory storage which I stored on my local computer to create files, process stylesheets, store images, and create output HTML files. The Docbook repository can be found at the Docbook project hosted at SourceForge website at http://sourceforge.net/projects/docbook/. This repository is an open standard project with the help and work from many developers. This file system structure contains XSLT processing engine, Saxon, which I used to transform my XML source documents into HTML output files. The file system also contains directories of organized XSL stylesheet files. The file system contains the standard Docbook XSL stylesheets and Norman Walsh’s Website stylesheets. As previously mentioned, Walsh’s stylesheets serve as a greater starting point for a beginner desiring to use XML content for website display. This was one of the factors leading to my decision to use Docbook. The file system also contains a simple static website which uses the XSL website stylesheets to serve as an example and guide.

The next step was to write all the HTML website pages into XML to be used as source documents. This step was accomplished by downloading the HTML pages from the SBA home page. I would examine the HTML file and write corresponding XML files to create the document structure. The SBA home page used a table structure to separate page elements and used graphics within these table data cells to markup the page with colors and links. I designed a similar XML file using the informal table element for my structural entity. The SBA XML source document contains four tables to separate the layers and to separate content with each row.
The next step was to use the pre-existing Docbook stylesheets and Walsh’s website stylesheets to markup my SBA XML source documents. These template stylesheets utilize XSL capability to extend and import multiple stylesheets for processing XML documents. The website stylesheets simply import the standard Docbook stylesheets, and can be viewed as a customization to markup specialized XML documents created for the purpose of website publishing. These stylesheets create a very plain and simple layout for all existing elements. Thus enabling the creation of customized stylesheets to markup and display your content as you desire. Customization using CSS and XSL will be explained after I discuss the XSLT processing engine, Saxon, which is used to transform XML documents using these stylesheets.

Saxon is a simple XSLT executable file. I decided to use Saxon because of its ease of use and ability to work on a Windows platform which I was using on my local computer. The Saxon processing command is executed at a command prompt, and takes parameters for output file, source file, and stylesheet used for processing. To maintain a complete website, a batch file can be implemented but due to the scale of my project, I would work on one file at a time and thus processing all the web pages within a web site every time was unnecessary. After I had created the XML source file, I would use the DOS prompt and execute Saxon with the appropriate stylesheet (CSS, XSL, or the website XSL) to output my desired HTML file.

Next, is the customization piece using CSS and XSL. After I created my Docbook source documents, I would create customized XSL stylesheets which would import the website XSL stylesheets. As noted earlier, the website stylesheets would also import the Docbook stylesheets. This extensibility feature of XSL shows the capability of writing
layers of customization code to achieve the desired markup effect. My template XSL files
would thus insert a parameter into the HTML output file for an external CSS or would
designate another customized XSL file. Each of these stylesheets can them be used to
markup the content of the HTML page in different capacities. One benefit of CSS was the
capability of Norman Walsh’s website stylesheets to ID entries in the XML document.
These entries would produce subsequent table data cell ID’s in HTML which gave me the
control to individually markup each data cell to my specifications. This was important
aspect for a HTML website Miami University’s SBA site which used tables for its
structure. The XSL customized stylesheet gave more control over output and placement
of HTML elements compared to CSS method, and thus produced a simpler and cleaner
code which would be important for maintenance. Consequently, XSL required a higher
complexity and additional time to design.

Docbook has it drawbacks and advantages in using for websites. Docbook
provides a simple, easy implementation for display static content on websites. These web
sites are used for informational purposes, or displaying articles and documentation.
Docbook could also be used to manage a portal which incorporates many links and a
structured format. Link management would utilize Docbook implementation systems and
structure. Some severe drawbacks are that Docbook is limited to static pages. Interactive
content is difficult if not impossible to implement. Websites which promote high user
interactivity would be advised to seek other alternatives.

6. Conclusion

XML is the data format for the future. The trend to separate content from
separation is a much desired characteristic for webmasters. Websites seek to develop a
consistent look and feel across all website pages using templates. XML, inherent in its nature solves this task. More and more companies are adopting XML as their data format, for data exchange, application information storage, server information retrieval, and centralized website content. The market for XML content generated websites will continue to grow.

Three strategies for XML content generated websites have been introduced in this paper; Docbook, AxKit, and Python. Each strategy possesses the important attributes necessary for generating custom output from XML source documents. Careful consideration should be given about what type of output is desired, programming environment used, and developer expertise required before choosing a XML content generation strategy. To conclude, here is a comparison of the three strategies.

Docbook is very easy to write and provides a simple architecture to create output. The availability of pre-existing stylesheets for display Docbook documents as HTML files gives Docbook the edge as far as amount of time required to create these files. Docbook has an advantage over Python and AxKit when the published content is in the form of articles and text resources. Consequently, the problem with using Docbook for website source documents lies in the fact that it wasn’t designed for this purpose. Docbook was created to be used for documentation and hardware/software articles. Docbook presents some difficulty in marking up and displaying content for attractive websites. Additionally, some implementation difficulties exist in processing multiple web pages and including CSS stylesheets. Working with XSLT engines and processing multiple web pages within a website requires more time and effort. Designing dynamic
web pages also creates complications as detailed implementation strategies are needed compared to other facilities like Python and AxKit.

AxKit clearly provides a better facility for storing XML documents and generating customized output for client needs. AxKit has been modeled after and used jointly with mature application server technologies like Apache. AxKit is very scalable and provides the flexibility and interoperatability that developers desire when choosing to use XML. AxKit provides a clear separation of content from display, which is the main focus of XML. AxKit supports numerous types of stylesheets and transformation techniques to give developers more options and control. AxKit supports more possible data output formats compared to Docbook and Python. The only limitation is the requirement of extensive knowledge of stylesheets and document storage and management in order to operate AxKit upon a UNIX compatible server. With the growing number of developers utilizing XML technologies, this should not be an issue.

Finally, Python provides the small scale capabilities that some business desire. Python has simple, efficient tools such as HTMLgen which make website publishing very easy. Python provides the object oriented language capabilities that developers desire to create dynamic and complex content without a complex system. Python is still maturing as a language and is not a commonly proficient language among programmers. For this reason, Python is limited to the support for multiple data output types it can produce. Python is mainly used to generate HTML and text output. Until Python matures, expertise and learning would be a requirement and possible downside versus Docbook and AxKit, when choosing Python power websites.
Figure 1

Source: O’Reilly: Perl.com
Figure 2

Docbook

XML Source Document \rightarrow Docbook XSL \rightarrow Website XSL \rightarrow Customized XSL \rightarrow CSS \rightarrow Optional Inclusion \rightarrow Output HTML
References


26