

# Real-world Validation with XMLProbe: a Technical White Paper

last revised October, 2006

## Introduction

This paper discusses how content validation of XML can be used for business benefit. It proposes that improving the receipt and management of XML, making these activities more cohesive, and actively enhancing the reliability of content, is the basis for an increase in data governance.

XML can make managers sad, especially when workflows, data models, and systems have become established and are not achieving the cost/benefit ratio that was expected of them. With time, the understanding of how content can be used has changed, and with new expectations have come new requirements. Managing a legacy body of content and legacy workflows in a cost-effective manner has proven an insufferable task for many. The management of content *quality* is a neglected area – even though that same content is often an organization’s most valuable asset.

## History of XML Validity

XML’s precursor, SGML, had at its basis the concept of correct (‘valid’) documents. There was no well-formed document concept before XML – everything had to validate against a DTD. Only a small number of people used SGML – with its linguistic basis and expensive tools support it occupied a niche of document preparation and publishing. The revolution came in 1997 when XML popularised structured markup and introduced the concept of well-formed documents. The barriers to entry were lowered and the party began – but immediately the need to define expectations in content was revisited and the W3C XML Schema initiative commenced [19]. Since then other validation languages have been developed, all intended to allow constraints to be thoroughly specified for XML documents.

## What is Valid, Really?

Considered more widely than in its strict technical sense<sup>1</sup>, ‘validation’ means ensuring that a common understanding has been reached.

While for many workflows a DTD or schema may exist, it is often not supported by any detailed definition of how the markup should be used, or what *additional* constraints exist that could not be encoded into the schema. In the XML world these extra constraints are sometimes referred to as ‘semantics’, and testing for them as ‘semantic validation’. The result is that while content may validate perfectly against a DTD or Schema, it may still be critically faulty as it is not ‘valid’ according to the wider business criteria surrounding it.

## Benefits of Validity

Correctness comes into its own when used at the interfaces between business functions. The interchange of information relies on an agreement between parties – if such a contract exists. This is usually expressed in a schema language and some business rules. In the web developer world there is a mantra: ‘be liberal in what you accept, and exact in what you emit’. However, with data that people are receiving from suppliers that mantra is only half true: exact specifications must exist, and be respected, for the benefit of both parties involved.

## Validation Technologies Compared

Recently there has been much activity (under the ISO standardization auspices) on defining a new set of Document Schema Definition Languages (DSDL) [6] which addresses weaknesses in the existing validation technologies that have been developed by the W3C: namely XML DTDs and XML Schema.

---

<sup>1</sup>The XML Recommendation [16] defines validity in terms of what a ‘validating processor’ must do: ‘report violations of the constraints expressed by the declarations in the DTD, and failures to fulfill the validity constraints given in this specification.’

Domain scope	Strengths	Weaknesses
<b>DTD.</b> [ISO, W3C] Structural constraints on XML documents	Intrinsic part of XML 1.x; good support in tools; schema definition is only one of the features of DTDs	Poor datatyping; not XML document syntax; no support for namespaces
<b>XML Schema.</b> [W3C] Part of the foundation of XML in the vision of the W3C (standardization)	Borrows many ideas from OOP design; high vendor support; namespace support	Considered complex; heavy emphasis on determinism; fixed datatyping system; differing interpretations
<b>RELAX NG.</b> [OASIS, ISO] Result of the merge between RELAX and TREX	Strong mathematical grounding; modular mechanism for datatype systems; namespace support; compact syntax variant	Poor vendor support; relatively low adoption rates
<b>Schematron.</b> [ISO] XPath-based language for defining context dependent rules	Conceptually simple; checks business logic as well as structure	Poor vendor support; completeness uncertain
<b>DSDL.</b> [ISO] Family of standards incorporating RELAX NG and Schematron	Document perspective; benefits of hindsight over earlier W3C work	Many parts of standards still in infancy; overall quality to be determined

Table 1: Validation technologies compared

DSDL, as it emerges, promises to address many of the validation shortcomings of the existing W3C technologies. This has already led to some debate about where technology implementers should be directing their energies [17]. Large scale technology vendors, who have generally had their fill of XML compliance requirements, are not inclined to market a new set of tools having expounded the values of XML Schema and other W3C technologies. However, the consultants and experts who are regularly confronted with customers wanting to do real things with XML technologies are finding the technical status quo inadequate.<sup>2</sup>

Of all the technologies being developed as part of DSDL one has really caught the imagination of users and technology vendors alike: Schematron [12]. This interest may lie in the orthogonal way in which validation is viewed in Schematron – it is not about the structure of the document in a narrow grammatical sense, but instead gives users the ability to express rules based on the wider business rules applying to their instances.

## Introducing XMLProbe. . .

XMLProbe is a validation engine designed to apply rules to XML instances and associated content, and report if and how those rules are violated. It anticipates and tracks emerging validation standards, and addresses weaknesses in current offerings. This paper will get readers up and running with the philosophy and practice of XMLProbe rules creation and implementation.

XMLProbe is a validation engine that efficiently interprets validation rules, applies them to content and provides predictable output. XMLProbe has its own design philosophy grounded in the concept of computational integrity, and which recognises good practice in XML validation technologies and standards, whilst simultaneously dealing with some scope and usability issues.

<sup>2</sup>It is interesting to note that some high-profile modelling work, for the forthcoming XHTML 2.0 [14] and DocBook 5 [5] specifications, is being done using DSDL Part 2 (RELAX NG) as the normative schema language; W3C Schema and DTD versions are then auto-generated from the RELAX NG source.

There are various tasks within the field of XML validation that XMLProbe addresses as part of an architectural analysis of the weaknesses of current offerings (including Schematron). These weaknesses are that:

- Constraints are not identified, so it is hard to determine what type of problems exist when they are violated
- Violations are not reported in XML, making integration with other systems difficult
- Reports should be generated detailing which constraints have been broken
- Details of where in the source instance the violations have occurred should be provided

## Addressing Questions of Governance

An indicator of the maturity of a sector is the extent to which issues of management are addressed. IT governance is emerging in other areas but not yet that of XML processing. Governance is: 'Providing the structure for determining organizations' objectives and monitoring performance to ensure that objectives are obtained' [10]. XML-based workflows too benefit from such an approach.

In general, governance is usually managed by the twin strategies of involving stakeholders and ensuring objective evidence is transparently available. Objective evidence will take the form of metrics, quality, reporting and traceability (to requirements and strategies). XML Governance ensures that the correct schemas, skills, personnel, procedures, practices, politics and feedback are in place for well-managed XML [8]. The rationale is to promote ownership: by providing transparency (visibility), so that the right people can be allotted and accept responsibility.

It is recognised that a process improvement approach to quality needs to provide organizations with the essential elements of effective processes. An example of this way of viewing an organizations reproducibility and maturity is CMMI, the Carnegie-Mellon Capability Maturity Model [3]. This view of organizational characteristics can be used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes.

For digital content, XMLProbe addresses these issues. Developed over a period of half a dozen years to tackle specific problems found in XML projects, it was recently turned into a product, with attendant licensing strategies, documentation, and deployment releases after the award of a DTI research and development grant in 2004. Since then, use of the product has grown.

A particularly interesting example of an application that XMLProbe has been put to is the 'data firewall' constructed by Cambridge University Press. CUP have managed to minimise their participation in the supply chain process by developing validation rules that are trusted enough to provide assurances that content passes a quality threshold. At this point, CUP routes documents directly from their suppliers to their onward destinations. This application has evolved over time, and has involved working with suppliers (who are given an executable version of the tool to check content locally). Under this system, everyone wins: suppliers can address issues early in the production process and the publisher gets a more timely throughput of content than would otherwise be achievable. The payoffs are thus 'meat and drink' business benefits: cheaper, faster, better.

## Selecting Nodes

XMLProbe rules are constructed around the SILCN (Selection Identification and Location of Content Nodes) language [13]. At the beginning of an XMLProbe configuration file a set of flags are used to control the operation of the processor. This controls whether such optional features as schema validation, warning suppression, namespace support, logging, encoding support, XInclude, etc are to be supported when the validator is invoked.

The ‘meat’ of the constraints, however, is constructed using a sequence of XPath [18] expressions. XPath is a very powerful language – XPath queries are written to match phenomena in the document that the rules author has decided need to be found. When XMLProbe is invoked on a document, if it finds nodes which match the pattern specified in XPath expressions, it will report them.

Below is a fragment of XML showing the use of an XPath to define a selection:

```
<silcn:set-criterion>
<silcn:id>0004</silcn:id>
<silcn:expression>
/doi_batch/body/journal/journal_metadata/abbrev_title[ . = ../full_title ]
</silcn:expression>
<probe:message>the content of elements abbrev_title and full_title is
identical, implying that the abbreviated journal title is
unknown: '<probe:eval>.</probe:eval>'
</probe:message>
<doc>A sanity check -- see p.19, 'Remarks': "full_title and abbrev_title
must both be submitted, and they can be identical. If you do not know
the abbreviated title for a specific journal, please supply the full
title in the abbrev_title element."</doc>
</silcn:set-criterion>
```

Within the `<silcn:expression>` element is the XPath expression which specifies nodes that must be reported. The complete rule is expressed by the entire content of the `<silcn:set-criterion>` element, but it is the XPath selection process that we are initially concerned with. Since the XPath expression returns a node for each matching occurrence in the instance document, rules must be written that describe situations that should *not* occur. Thus in common with XML parsers, and programming language compilers, XMLProbe maintains the idiom of ‘no news is good news,’ in that perfect input will not cause rules to trigger, and so generate no report of matched nodes.

The creation of XPath expressions to describe scenarios that are transgressions is enabled by two essential features of XPath – the use of axes to allow complete navigability within the document structure, and the ability to extend filtering and selection decisions using XPath functions and predicates. Both these capabilities are standard features of XPath 1.x. XMLProbe uses XPath’s in-built extensibility mechanisms to add a plethora of extension functions that enables a greater range of processing than ‘vanilla’ XPath implementations.

Working with XPath can be easy. Many commercial tools (such as Altova®’s XMLSpy®) have a built-in XPath evaluator which will allow developers to test XPath expressions against document instances and returns the selected content nodes for examination. This is a powerful tool that has many uses – one of which is ensuring that XPath expressions are correctly constructed.

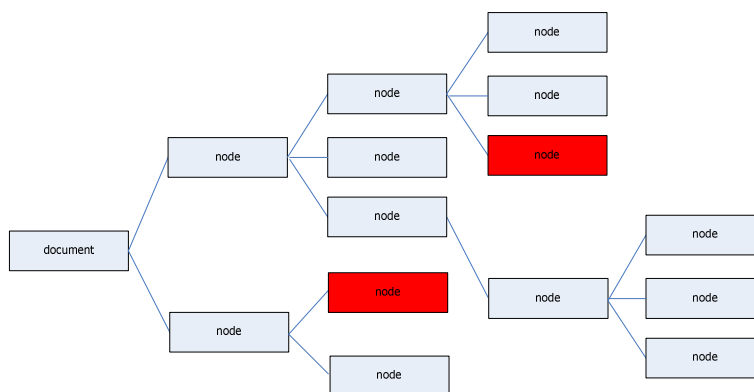


Figure 1: With XMLProbe, validation is seen as identifying and locating which of the many nodes in an XML document are faulty (here picked out in red).

## Identifying Rules

In our sample rule above, the rule carries its own identifier (the content of the `<silcn:id>` element). The use of rule identification is important to XMLProbe and is a cornerstone of its architecture. When rules are matched by transgressing nodes within documents, it is important to be able to determine what exactly has caused the match. By simply identifying the rules it is possible to support subsequent processing based on the rule that was matched. In particular, identified rules can be categorised and clustered, or have any arbitrary metadata attached to them, so that appropriate routing and action is taken under particular conditions.

When a rule is matched it does not necessarily mean that a document instance is incorrect. Reported content nodes are not necessarily wrong, but may merely indicate what might be noteworthy, requiring further machine processing or human attention. It is therefore quite possible that a document will need to be referred to different authorities depending on which rules have been transgressed, and the workflow stage the content is at.

The use of identifiers for rules allow applications to develop simple mechanisms for deciding what to do with documents based on the reports that are generated when running XMLProbe over document instances. For example, a certain rule when triggered, might be thought so serious that it triggers an action in a workflow – a seriously faulty document might be rejected by a system, say. At the other extreme, minor faults might simply be counted and reported (“there were 535 violations of rule *x* detected”) so as to avoid generating unnecessary noise in a live system. By using identifiers for rules, the action is effectively decoupled from the fault, and allows for maximum flexibility of processing.

## Authority for Developing Rules

Specifications of semantic constraints, where they exist, usually explain how the elements and other structures are to be used and what expectations exist of data adhering to the standards. However, not all specifications are either complete or well-written. Where non-computable requirements have been shelved or where the standard itself is not clear, XMLProbe can help by creating a case for developing such rules – if they can be implemented in a system, they will – once again – be worth developing.

Organisations often have their own rules which may be undocumented because they are regarded as common-knowledge, so in practice access to a domain expert is always useful when developing rules – there needs to be an authority, someone or some organization which will offer clarity on any points of ambiguity. This role is necessary not only to clarify standards but also to spot interpretation errors.

## Processing Output

In schema processing the results of a validation are binary – true or false. However, the real world is more nuanced, and validation should inform users of the *extent* to which a model is matched by the presented instance. Validation must report the extent of correctness if applications are to use that knowledge to determine appropriate behaviours in handling a document.

However, for schema processors (validating parsers) there is no standard definition of expected output of a validation. It is assumed that ‘something’ represents trouble (not-valid) and ‘nothing’ represents conformance to the model (valid). This is clearly inadequate and there is obviously a need to generate consistent and processable output if we expect ever to measure and manage content.

XML validation has traditionally been an opaque process. Since there are no standards for the output of validators, tool writers have been left to devise their own. Parsers and validators thus emit messages with arbitrary degrees of diagnostics, or (if the document is correct/valid) usually nothing at all. Non-technical users, in particular, experience frustration when confronted with the output of a failed validation, which often looks like an alien language. This problem is also true for machines. The lack of definition of what validators should produce has led to a lack of predictability about what to do with their output – the output does not compute!

The output of a validation process should be as well-defined as the inputs to the process. This is an area that has often been undervalued by the developers of software tools. But it is an area in which XMLProbe excels.

The output of an XMLProbe validation is by default an XML document. However, as XMLProbe has a built-in transformation engine it can be setup to generate other formats such as (X)HTML. Indeed there is a default HTML output which can be triggered in the configuration options by setting the relevant parameter in the configuration preamble of a rules set.

## Locating Nodes

When rule violations are found in documents it is important to be able to know where the transgression occurred. Without precise location information the validation process is really pushing the diagnostics back to the document creator (without much support). By providing the location of each transgression in the output of an XMLProbe report, the user can go straight to the problem and deal with it directly.

XMLProbe emits both an XPath expression to locate the offending node within the source document, and also line and column offsets which can be used in non-XPath aware environments when linking results of a report back into the source document, as shown below.

```
<silcn:node>
  <silcn:expression>
    /article/back/ref-list/ref[2]/nlm-citation/source
  </silcn:expression>
  <systemId>bmj_sample.xml</systemId>
  <line>271</line>
  <column>19</column>
  <text>Count for source is BMJ which is not the same as the value
    in the attribute count</text>
</silcn:node>
```

It is common for users to want to be able to walk through issues identified by the XMLProbe report. So information that allows the construction of links between the report and the instance is extremely useful.

## Documentation

The message included in the report is designed to provide human consumable evidence of why the content node has been selected. As such the message should be carefully authored in a style that can be comprehended by the intended audience. Especially useful within the report is the ability to evaluate items in the content node and include them in the output, effectively allowing the creation of ‘dynamic’ error messages. This is done by using the <probe:eval> element which specifies an XPath expression to be evaluated within the element’s text.

In providing evidence within a message, it is always useful to be able to link to a specification or document that provides support for the rule being transgressed. This is made much easier when the specification (or documentation) itself allows linking directly to the relevant section or passage of text that supports the rule. Unfortunately many pre-existing specifications were not designed to be used in this way – suggesting that organisations which control a data specification would benefit from making that data specification available for linking, ideally as (X)HTML served via a web server.

Related to the use of specifications and documentation is the issue of change management. Ideally each rule is carefully versioned and assembled into packages for use in particular contexts. In our experience change identification, notification and management is an afterthought, but something that could have detrimental impact of business responsiveness and the ability to measure and manage. As with documentation, the answer to versioning problems lies largely in the careful creation of better, more precise documentation. The irony of the documentation ‘problem’ is that writing down clearly what rules businesses are operating to has great advantages to the business itself.

## Extension Functions

With XMLProbe, extension functions can be written to check for data type constraints, for example ISBNs can be checked for syntactic and checksum correctness, or DOIs for conformance to a set pattern. Indeed, using this extension mechanism it is possible to add arbitrary checks using the capabilities of the Java programming environment.

It is also common to want to examine outside an XML instance: to inspect the existence and properties of directories, files and networked resources. Extension functions allow the presence of associated files such as images to be checked. As well as performing basic operating system queries, it is also possible to inspect properties of files (such as image headers and embedded metadata) to ensure that they are as expected.

The ability to write extension functions and the library of built-in functions are an important aspect of XMLProbe's capabilities, and effectively extend its validation scope to 'what can be computed', since the native abilities of the Java programming language are wholly available.

## Integration

XMLProbe is capable of working in a large variety of configurations. It is an engine which supports a clean architectural principle of 'XML in; XML out' and is developed in 100% Java. This flexibility is intended to support its integration into various environments and systems, from standalone desktop applications, to hosted enterprise application.

Some examples of how XMLProbe has been integrated include:

- As a hosted service within a publishing organisation on an Linux/Apache/perl/CGI platform
- As a component with a .NET enterprise application on a Wintel platform
- As part of a desktop application with a GUI and installer, for Microsoft Windows desktops
- As part of a Sun/Solaris/Tomcat enterprise application receipting content from offshore suppliers, using multiple parallel validation pipelines to leverage the hardware's multi-processor architecture.

XMLProbe has been used to trigger events based on certain rules being met and certain thresholds having been reached. The platform-agnostic nature of XMLProbe, and its ability to interface with many systems through its use of open (and *de facto*) standards, makes it a valuable component in any large-scale application of XML technologies, where a rule may be written once and executed many times.

Ultimately, integration should reward users and make working with structured content a more pleasurable experience for users and management. With appropriate validation comes increased confidence in the XML being managed. And stakeholders accept responsibility for the appropriate use of validation in a properly governed environment.

## Industry Schemas

Many industries are adopting cross-industry schemas, often developed by consortia of interested parties.<sup>3</sup> In many cases such schemas are written with the express intention of becoming *de facto* best practice for digital content representation within that industry. They may compromise on points of difference between organizations allowing for different views to be accommodated within an aspirational specification. In some cases successful schemas pervade an industry through a less consensual approach – usually those created by dominant players in a sector. In both these cases of industry/sector adoption of schemas, it is important that there is community agreement of the interpretation of the use of structures within the schemas.

---

<sup>3</sup>for example, AdsML [1] in the advertising industry, or FpML [7] in the financial products industry.

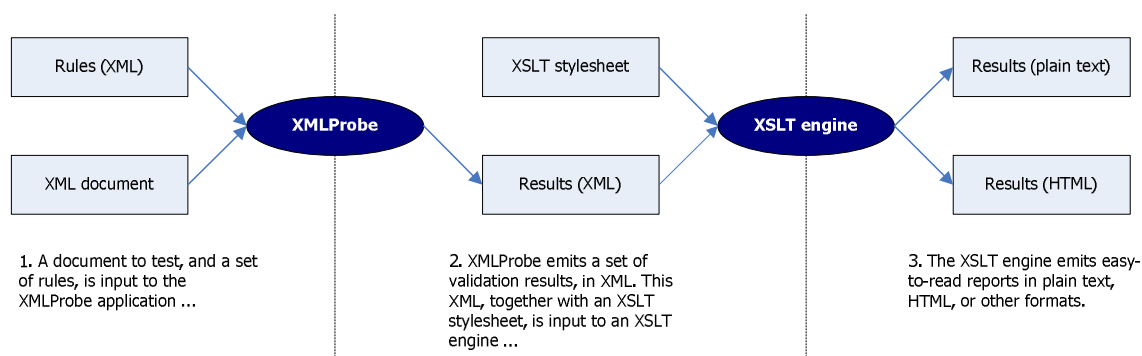


Figure 2: Typical use of XMLProbe in a pipeline

Without such agreement the opportunities for interoperability are curtailed. The use of validation allows industries to have confidence in the data used for exchange with their trading partners, or reused internally. By increasing the trust in the quality of content, XMLProbe allows organisation to conjecture new ways of re-using content, confident in knowledge that their data will perform as expected.

## Support for Industry-standard XML models

XMLProbe has been used to implement some public online services to promote the use of modern validation. The schemas that can be used for checking in this way at the moment are focused on the publishing sector

- NLM Journal Publishing DTD [9]
- CrossRef schema for linking metadata [4]
- ONIX for Books – book product supply chain metadata [11].

Through addressing particular schemas and providing publicly accessible services to validate content for rules, potential users can get a feel for the types of checks that can be done using current validation technologies.

Industry level support is often not enough for real-world applications. In practice each generator of content can have slightly different standards – which are looked upon as part of the organization’s brand values and represent the idiosyncrasies that allow knowledgeable readers to tell who created a product devoid of logos and other obvious clues. These idiosyncrasies take the form of house-styles and editorial practices and manifest themselves in things like identifier construction, use of types of English and controlled vocabularies.

To accommodate this, XMLProbe supports a hierarchical view of rule sets. The support for XInclude [15] allows rules to be imported into a standard rule-set, thereby augmenting it. By allowing the separation of different rule provenances, it is possible to cater for multiple sources of constraints in a manageable way such that constraint conflicts can co-exist between different user communities. Not everyone chooses (or has the option) to adopt industry standards, and there may be value in creating bespoke schemas which model a particular perspective. In such circumstances, organization-specific validation knowledge is essential in ensuring business integrity. The coding and management of business rules is especially important in organisations that define and manage their own schemas.

It should also be possible for individual users to create their own checks. This democratization of the validation process would allow all participants to add to the overall knowledge base, providing mechanisms for propagating insights into particular problems. This transparency of concerns addresses some general governance issues of participation and feedback in the quality of the XML. Visibility of effort also ensures that no surplus development occurs.

## Appropriate Validation

Content has different requirements made of it at different stages of its lifecycle. Traditional schemas assume a single complete document instance, in its final form. It would be better if appropriate validation occurred at specified milestones in a document's creation (for instance some metadata should be captured as the document is created and some will necessarily be applied retrospectively).

Typically in a content production process there will be stages that relate to conceptual milestones such as:

- Creation — the editorial/authoring processes which may be only partially complete at any point in time, but should converge on initial completeness
- Review — feedback to the creator/editor which will not necessarily form part of the content and may need to be separately managed
- Classification — creation of metadata may occur after the content of a document is finalised and is therefore a lifecycle-dependent issue.
- Publication — the act of publishing a document may in itself alter the content of a document., e.g. publication metadata in a journal article depends on which issue an article is published in.

Each of these stages may require different validation rules be applied. Appropriate validation should be integrated into organizations' business processes.

## Looking Forward

The aim of XMLProbe is to be the tool-of-choice for implementing managed validation rules. The main drivers of XMLProbe development are:

- Standards support — DSDL integration, Schematron language support, NVDL, DTLL, CRDL – all side effect free technologies that report on the condition of an XML document package
- Performance factors — The question of JIT rules optimization has been raised in relation to the performance of XPath queries within the XMLProbe engine. Additionally work is being undertaken to develop new, more efficient ways of processing XML using existing systems [2]. Potentially performance could be even further addressed by offering hardware based appliance solutions.
- Extensibility — enhanced off-the-shelf functionality provides a competitive advantage. It is important that extension functions are well documented so that they can be discovered, accessed and integrated easily.

In the longer term the quest is based around the question – is it enough to tell people accurately what is wrong with their content? The next stage of XMLProbe development may lead us towards an XMLCleanse, in which certain categories of error are automatically fixed by machine.

## Conclusions

Understanding the benefits of codifying business logic checks into rules, and an appropriate investment in the management of those rules is necessary for addressing the issue of XML content governance. The rules engine should be secondary to the management and specification of the rules themselves using standards-based definition capabilities.

The cost of change may have an inertial impact on the desire to adopt governance-enabling technologies – however this would be a false economy and seal the fate of XML as an ill-thought of technology.

Data needs to be correct – in the same way that computer programs are meaningless unless they can be correctly interpreted, documents should also conform to rules specifying the expectations of how they might be used. Without affirmation, the confidence of the users of content will be less than it could be. The path to correctness is the validation of business logic.

The new generation of validation technologies addresses many shortcomings in the current standards. The opportunities to improve throughput, and explicitly state business expectations of content/data should be compelling reasons to consider the application of business level validation in all environments where XML usage is more than trivial. The governance of data flow (through conceptual architectures like data firewalls) allows the measurement and improvement of content.

The intellectual property benefits of owning the rules that describe your value should not be underestimated. Proactive management prevents vendor lock-in and allows increasing maturity in the use and management of an organisation's content resources. The opportunity that exists today for organizations to radically alter the way XML is managed is phenomenal. Never before has there been such a convergence of content and capabilities in the software tools. Given the risk inherent in poor data quality – can you afford not to look seriously at the new developments? The time is right to move forward confidently and see what a difference XMLProbe can make to your perception of what working with XML means.

## About XMLProbe

XMLProbe is a ready-to-run command line tool with integrated DTD- and Schema-aware XML processing software. It comes with a library of built-in XPath 1.0 extension functions and a Java API for developing further customised XPath extension functions. XMLProbe has XInclude support so that licensees can create flexible and reusable QA suites.

XMLProbe is built using the Jaxen open source XPath implementation. It requires Java 1.4 or greater runtime. It can be accessed on the command line or using its API from a shell or servlet environment. As it reports in XML format the results of validation with XMLProbe can be used to trigger further processing options within an application. XMLProbe is available under a number of licensing schemes to suit different requirements: from standalone, to multi-site and third-party installations.

The XMLProbe team  
Griffin Brown Digital Publishing  
Orwell House, Cowley Road  
Cambridge, CB4 0PP  
United Kingdom  
Tel: +44 (0) 1223 425 730  
Email: [info@xmlprobe.com](mailto:info@xmlprobe.com)

## References

- [1] AdsML®. See <http://195.52.248.218/WebSite/adsm1.nsf/HTML/Index.html>.
- [2] Brown, Alex, *Frozen streams: an experimental time- and space-efficient implementation for in-memory representation of XML documents using Java*. See <http://www.idealliance.org/papers/extreme/Proceedings/html/2006/Brown01/EML2006Brown01.html>.
- [3] Carnegie Mellon Software Engineering Institute. *Capability Maturity Model Integration*. See <http://www.sei.cmu.edu/cmmi/>.
- [4] CrossRef XML Schema. See [http://www.crossref.org/02publishers/24upload\\_spec.html#schema](http://www.crossref.org/02publishers/24upload_spec.html#schema).
- [5] DocBook V5.x. See <http://www.docbook.org/schemas/5x>.

- [6] ISO/IEC 19757 - DSDL. Document Schema Definition Languages. See <http://dSDL.org/>.
- [7] Financial products Markup Language (FpML). See <http://www.fpml.org/>.
- [8] Jelliffe, Rick. *XML Governance*. Open Publish 2006, Sydney, Australia. See <http://www.openpublish.com.au/06papers/>.
- [9] National Center for Biotechnology Information (NCBI), Journal Publishing DTD. See <http://dtd.nlm.nih.gov/publishing/>.
- [10] *OECD Principles of Corporate Governance*. OECD Publications. 2004.
- [11] EDItEUR. ONIX for Books. See <http://www.editeur.org/>.
- [12] ISO/IEC 19757-3:2006. Information technology, Document Schema Definition Languages (DSDL), Part 3: Rule-based validation, Schematron.
- [13] SILCN 1.0. Selection, Identification and Location of Common Nodes. See <http://www.griffinbrown.co.uk/silcn.html>.
- [14] XHTML™2.0. W3C Working Draft 26 July 2006. See <http://www.w3.org/TR/xhtml2/>.
- [15] XML Inclusions (XInclude) Version 1.0. W3C Recommendation 20 December 2004. See <http://www.w3.org/TR/2004/REC-xinclude-20041220/>.
- [16] Extensible Markup Language (XML) 1.0 (Fourth Edition). See <http://www.w3.org/TR/xml/#proc-types>.
- [17] xml-dev mailing list, thread originated by Howard Bartel on 27 June 2006 with subject 'Restrictions on existence of attributes?'. See <http://lists.xml.org/archives/xml-dev/200606/msg00279.html>.
- [18] XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999. Available <http://www.w3.org/TR/xpath>.
- [19] XML Schema Part 1: Structures. W3C Recommendation 2 May 2001. See <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.