A World of Difference

Why you need an XML change control solution

www.deltaxml.com

XML Europe 2003, London

ABSTRACT

"Change Control for XML: Do It Right"

How do you manage changes to your XML data? In XML, standard tools fail to identify changes precisely, meaningfully and completely. Learn how to solve this problem intelligently, representing changes in an XML format that allows downstream processing in an XML pipeline.

This presentation explains the issues and shows how DeltaXML addresses each. More importantly, the presentation will show you how you can identify new opportunities made possible by intelligent handling of changes to your XML data.

DeltaXML provides a proven technology for identifying changes to your XML data and documents and represents those changes in XML. Whether your need is for content management, versioning, regression testing of XML software, data synchronization or B2B exchange, DeltaXML provides the most comprehensive system for change control of your XML data.

DeltaXML has redefined the benchmark for XML differencing. Its solutions have been selected by teams at Cisco, IBM, Electronic Arts, Cognos, Intelliden and Wolters Kluwer.

Robin La Fontaine, DeltaXML (Monsell EDM Ltd)

Introduction

XML, the eXtensible Markup Language, is fast becoming the standard format for data exchange. As applications and processes are increasingly designed to depend upon XML it is becoming essential to accurately identify and control changes to this XML data. And as XML becomes a mission-critical technology within the organisation, the commercial value of a comprehensive change control solution becomes evident. Such a solution must be

- Highly efficient – small footprint, high performance
- Proven in the field
- Flexible – to accommodate the diversity of XML change control requirements
- Easy to integrate into existing XML infrastructures
- Generic – to handle tomorrow's data as well as today's

It is also critically important that changes are easy to process once identified. Whether the data is to be used by humans or computer processes the change information is only valuable if it can be presented to and interpreted by the consumer. Finally, as ever, capacity is important. With the take up of XML the size of XML files is growing and it becomes important to be able to quickly find and manage individual changes within significantly large files.

This paper presents an analysis of XML change control. Specifically, (i) the need for XML change control, (ii) why XML change control is difficult, (iii) how to evaluate XML change control tools and (iv) the DeltaXML solution.

Why is XML change control important?

If your data never changes, you don't need an XML change control solution.

For the rest of us, changes to data must be identified, tracked, communicated and synchronized. Whether the data is textual website content within a content management system (CMS), manufacturing data within an XML database or financial data feeds from a third-party supplier, tracking data changes gives you fine control over your data, allowing you to focus on the updates. Typical applications range from content management and data versioning through regression testing to data synchronisation and merging.. Examples include:

Synchronizing changes made on disparate systems by automatically merging changes directly into a master XML document.
Enabling editorial processes by allowing editors to check changes made against the original data - Microsoft® Word does this for flat documents with "Track Changes", now your documents are in XML you need a generic solution for all your corporate data.
Managing data translation by identifying sections of text that need to be retranslated within updated versions of the master document.
Reducing processing by exchanging only changed data. By pre-processing data feeds, database repopulation costs can be reduced.
Reducing bandwidth by transmitting only updates. Clients can merge these updates with their existing data records to keep local copies of the master database up-to-date.
Increasing security by splitting sensitive data into separate streams which are reassembled on delivery.

With solid XML change control in place new solutions become evident, solutions that were simply not possible before. For example, pipelining of corporate data allows extraction and data mining in real time, with modified feeds merged back into the pipe. This is where XML is heading – and change control is key.

Why is XML change control difficult?

Finding the difference between two documents or files is not a new problem and many tools are available that have been servicing this need for years. However, until now documents were either unstructured - making differencing a straightforward task of applying a standard "diff" algorithm – or proprietary. The structured nature of XML mark-up means that the old "diff" solutions are now hopelessly inadequate, the rules needed to identify changes within XML files are very different from those used to identify changes in unstructured text files. Traditional tools fail when applied to tree-structured XML. Proprietary approaches, tailored to a particular document structure, discard the key advantages of XML – openness, flexibility, standardization – and can never offer a generic solution.

To understand the difficulties, we'll ask, 'What constitutes a change to an XML document?' When applied to XML traditional string-based change control will identify a huge number of differences that are not significant in XML and should properly be ignored by an XML aware comparison. For example, the following files are identical as far as XML is concerned:

Example-1 No spaces or namespaces

```
<record xmlns="http://www.myco.com/records"
id="b123">
<name>Michael Brown</name><born>1984-03-
08</born><sex>M</sex>
</record>
```

Example-2 With spaces and namespaces

```
<staff:record id="b123"
 xmlns:staff="http://www.myco.com/records">
  <staff:name>Michael Brown</staff:name>
  <staff:born>1984-03-08</staff:born>
  <staff:sex>M</staff:sex>
</staff:record>
```

The second file is different from the first in a number of ways, including the addition of white space, the order of attributes and the declaration of the XML namespace (as the default namespace in the first, using a prefix in the second). None of these changes is significant as far as information in the two XML files is concerned; the two files are "XML-identical".

Other differences in XML are significant, but not as significant as a simple string comparison would suggest. The rules for determining whether these differences constitute a valid change will vary on a case by case basis. For example, if the order is important then there is a significant difference between the two files below, but if order is unimportant the only change is the two added elements.

Example-3 Original records

```
<record id="b123">
  <name>Michael Brown</name>
  <born>1984-03-08</born>
  <sex>M</sex>
</record>
<record id="b124">
  <name>Gillian Bryan</name>
  <born>1951-03-06</born>
  <sex>F</sex>
</record>
```

Example-4 Updated records

```
<record id="b124">
  <employee-no>BR12</employee-no>
  <name>Gillian Bryan</name>
  <born>1951-03-06</born>
  <sex>F</sex>
</record>
<record id="b123">
  <employee-no>BR24</employee-no>
  <name>Michael Brown</name>
  <born>1984-03-08</born>
  <sex>M</sex>
</record>
```

As another example, when the structure of the XML has changed even slightly a text-based comparison will often report a huge change to the document. Identifying the minimal set of actual changes that have occurred is a non-trivial task involving look-ahead. When a change is found in the sequence of elements the rest of the data must be searched to identify the re-synchronization point, accounting for possible promotions and demotions within the structure. Such problems are a source of much interest in academia. You need a solution that brings academic excellence into the commercial arena.

Choosing granularity

Most XML differencing solutions available today focus on the abstract challenge of identifying the optimal change. Whilst valuable and interesting, a commercially useful XML change control tool must tackle the more difficult problem of identifying a change as close to the optimum as possible whilst also making that change information available to both human and machine consumers for ongoing processing.

Most existing tools generate an annotated version of one of the source documents with comments identifying the differences. This approach is sufficient for requirements where an operator needs to view changes in context and either approve or reject them (though it requires the delta and both the original files in order to do this) but is not useful where the delta is to form the input of a further automated process. The approach is also inefficient when that change information is transmitted, as the whole file must be sent rather than just the change

information. Sending delta files rather than retransmitting large amounts of data can significantly reduce bandwidth and storage requirements.

The granularity of changes is also important. For some applications it is not appropriate to identify changes at too low a level, such as a single field in a record or a single cell in a table: rather the system must ensure that the whole record or row is identified as modified, ensuring that a consistent set of related fields is updated at the same time. For other scenarios, such as the human review of changes to text, the correct granularity will range from the individual word or character to the paragraph. For machine interpretation of the results a minimum delta is most common. A purely academic approach to XML change will be concerned with generating the minimum change for all scenarios, regardless of whether that level of granularity is appropriate for the consumer.

Evaluating XML change control solutions

There are three essential criteria when evaluating an XML change control solution: accuracy, representation and usability.

Accuracy of the result

Does the tool accurately identify genuine changes to the XML data? It is imperative that the tool does not produce spurious results because of changes in the order of attributes, the use of whitespace or of namespace prefixes or any other difference which is not significant in XML. Within specific applications other requirements may apply – perhaps changes to a time-stamp element or attribute should be ignored – so it is important that the tool allows such configuration and still reports correct results.

Other configurable behaviour should be the handling of white space (critical for some users, irrelevant for others), and how the tool handles combinations of ordered and unordered data. To ensure 100% accuracy requires the use of keys in the data to identify sections that must be matched, so that (for example) inserting a new clause into a legal document doesn't cause misalignment of the comparison. It should be possible to add such keys to the data as needed during processing, stripping them from the output to keep business data pristine.

Representation of the result

Can the change information be used? Uses range from visual review and approval of changes, to data pipelining, to change-triggered processing. Does the solution provide the flexibility to allow the results to be used in the scenarios you are likely to encounter in your organisation?

As well as the flexibility of the representation it is important that the representation is generic and allows change processing to be automated. For example, if the representation of the difference is dependent on the name of the element being differenced it is impossible to create a generic way of handling the differences and every file you evaluate will require a unique utility to interpret the difference information. If the representation is an "edit script", perhaps based on XPath, then you need the original documents to be accessible in order to make sense of the delta. An "XPath" solution leads to almost incomprehensible deltas which impede data pipelining and drag down efficiency. By contrast, an optimal solution would keep the structure of the delta very similar to that of the original. As an added bonus, it could offer both "changes-only" and "changes-plus-original" formats, to make in-context change identification a breeze – only the delta file would be required for a comprehensive change report.

Usability of the solution

It is important that the solution is both efficient and usable. Efficiency will determine whether the solution can be applied to rapidly changing XML data or large XML data sources. To be usable the solution must be easily accessible to developers who need to integrate it into automated processes, end user tools or other applications; it must be fully standards compliant and wrap internal complexities behind a clear API. To reduce integration costs to a minimum, first-rate documentation and technical support should be a priority.

Why choose DeltaXML?

DeltaXML provides a complete and flexible solution which produces an optimal or near-optimal delta in every situation. Delta XML is efficient, allowing processing of large files at high speed. The delta is represented in XML with a structure close to the original so that it can be used for human consumption or as a trigger or input for further processing. For integration, DeltaXML has a carefully designed and well documented Java API based on the SAX/JAXP/TrAX industry standards, which enables developers to rapidly integrate DeltaXML into effective business critical solutions. Our documentation, configuration options and technical support are designed to get you up to speed as fast as possible.

The scope of the DeltaXML solution, the investment in core R&D and the industry experience that comes from production use in diverse applications place DeltaXML well ahead of contemporary offerings and provides a future-proof choice for XML change control.

DeltaXML provides…

- Accurate results in every XML change scenario – with provision for the user, or developer, to define how particular difference should be handled.
- Patent-pending difference format and highly-tuned algorithms - providing a uniquely complete solution that generates the optimum change for ongoing processing.
- Close to minimal representation of change – INRIA ran independent tests showing DeltaXML ahead of other products.
- Representation of deltas in XML – allowing consumption by humans and by automata. For example, you can generate a visual display of the delta in HTML either on its own or in the context of the original document, formatted exactly as required. Alternatively, for machine processing you can isolate the delta with a change set close to the minimum and with a notation that is accurate and allows rollback of the change at a later time.
- Representation of deltas separate to the source documents – reducing the bandwidth required to transmit the change and removing the requirement to store multiple copies of the same document
- Extensive configuration options, designed for pipeline architectures.
- Scalability : able to compare very large source files, proven at over 50Mb, soon to be expanded.
- Java API - fully standards compliant, well documented and designed for deep integration.

DeltaXML offers the only truly comprehensive solution for managing change in XML.