RevXml

a tool for the tracking of revisions

of XML/SGML-tagged documents

User Manual

16 May 2002

RevXml: a tool for the tracking of revisions of XML/SGML-tagged documents

User Manual

Content

1 Introduction

This documentation describes a program, called RevXml, for the automatic tracking of revisions between two XML or two SGML documents. It is the successor of the succesfull program Sgdiff. The program is written by Gert J. van der Steen of Palstar bv.
RevXml runs in batch and functions as “middleware”: it is assumed that the results are processed by another application.

2 Description and examples

RevXml compares two files **A** and **B** which contain documents with XML or SGML markup. **A** contains the original document, **B** the revised document. The markup of SGML documents should be non-minimized, as produced by a SGML parser. The program constructs a third file **C** which contains revision markers. The revision markers take the form of tags or processing instructions, which may be redefined by the user. The revision markers surround deviating parts of the document or precede tags which contain attributes that have been changed.

<del>…<enddel>	surrounds a section which **A** has, but not **B**
<add>...<endadd>	surrounds a section which **B** has, but not **A**
<chg>...<endchg>	surrounds a section in which **A** changed into **B**
<revattrs>X Y</revattrs>	precedes a tag for which the name or the values of attributes X and Y differ.

2.1 Revision tracking

The following example shows the comparison of the two documents **A** and **B**, resulting in **C**.
(The text is stylized with white space and record ends around the tags in order to show the correspondences between **A**, **B** and **C**.)

A:	B:	C:
<A>texta <B>list <C>elem1</C> <C>elem3</C> <C>elem4</C> </B> </A>	<A>texta <B>list small <C>elem2</C> </B> <B>list large <C>elem1</C> <C>elem2</C> <C>elem3</C> </B> </A>	<A>texta <add> <B>list small <C>elem2</C> </B> <endadd> <B>list <add>large<endadd> <C>elem1</C> <add><C>elem2</C><endadd> <C>elem3</C> <del><C>elem4</C><enddel> </B> </A>

2.2 Atomic parts of the document

The smallest unit of a document that is used during the comparison is called the atomic part. In section 2.1 the atomic unit was a character; but there may be good reasons to take as an atomic unit not a character, but a word, a sentence or even the complete text within a #pcdata.
4 types of atomic units are implemented: character, word, sentence and pcdata.
A word is defined as a string between two spaces.
A sentence is defined as a string between two dots, including the second dot.
Pcdata is defined as the complete text in between two tags (start or end tags), and not containing tags within itself. This definition of "pcdata" differs slightly from the formal XML/SGML definition of "#pcdata" which may contain floating elements. For instance, in '<a>pqrst' the strings 'pqr' and 'st' are called "pcdata", even if element a was defined in the DTD as #pcdata with an inclusion of element b.

Example:

A:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA>Some of the latched indicators have an orange circle around
them. This shows that they are part of a system included in the
master minimum equipment list (MMEL). These systems must be
serviceable
before the aircraft is permitted to start a flight.</TPARA>
```

B:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA>Some of the latched indicators have a blue circle around
them. This shows that they are part of a system included in the
master minimum equipment list (MMEL). These systems must be
serviceable
before the aircraft is permitted to start an initial flight.</TPARA>
```

C if compared at the character level:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA>Some of the latched indicators have a<del><enddel>
<chg>blu<endchg>e circle around
them. This shows that they are part of a system included in the
master minimum equipment list (MMEL). These systems must be
serviceable
before the aircraft is permitted to start a<add>n initial<endadd>
flight.</TPARA>
```

C if compared at the word level:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA>Some of the latched indicators have <chg>a blue<endchg>
circle around
them. This shows that they are part of a system included in the
master minimum equipment list (MMEL). These systems must be
serviceable
before the aircraft is permitted to start <chg>an initial<endchg>
flight.</TPARA>
```

C if compared at the sentence level:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA><chg>Some of the latched indicators have a blue circle around
them.<endchg> This shows that they are part of a system included in
the
master minimum equipment list (MMEL).<chg> These systems must be
serviceable
before the aircraft is permitted to start an initial
flight.<endchg></TPARA>
```

C if compared at the pcdata level:

```
<TULPARA>The indicators change from black to white.</TULPARA>
<TPARA><chg>Some of the latched indicators have a blue circle
around them. This shows that they are part of a system included in
the
master minimum equipment list (MMEL). These systems must be
serviceable
before the aircraft is permitted to start an initial
flight.<endchg></TPARA>
```

If the switch "no_ignore_diff_in_tags" is set in the settingsfile (see section 5) it is assumed that text that is surrounded by different tags in **A** and **B** is different from each other. For instance, in the example above, if **B** would start with "`<TULPARB>The indicators change from black to white.</TULPARB>"` then **C** would get "`<chg><TULPARB>The indicators change from black to white.</TULPARB><endchg>".`

2.3 Revision tracking of attributes

If two tags are compared then their attributes, if any, are compared also. In the current implementation the result of the comparison is a list of the names of attributes which differ either by name or by value. The list is surrounded by the tags '<revattrs>...</revattrs>' (which may be redefined). The names in the list are in uppercase. The sequence of the attributes in **A** and **B** could be different. Their sequence in **C** is the same as in **B**.

During the comparison of values eventual quotes are ignored. **C** will retain the eventual quotes of **B**.

Example:

A:

```
<A at='x y' au="u v" av=aaq>abcd
   <B eetje='leeg'>xy</B>
   <D>d</D>
</A>
```

B:

```
<A at="x y" av=aap au='v' aw='w'>abcd
   <B eetje='loog'>xyz</B>
   <B aatje='leeg'>pqr</B>
</A>
```

C:

```
<revattrs>AV AU AW</revattrs><A at="x y" av=aap au='v' aw='w'>abcd
   <revattrs>EETJE</revattrs><B eetje='loog'>xy<add>z<endadd></B>
   <chg><B aatje='leeg'>pqr</B><endchg>
</A>
```

2.4 Discriminating attributes

Discriminating attributes effect the skipping of the comparison of elements. The use of "discriminating attributes" may drastically decrease runtime.
For instance, suppose a book contains a number of chapters, each chapter being an element with the chapter number as an attribute, e.g. CH. Chapters have to be compared only if their chapter numbers are equal. In RevXml CH may be defined as being "positive discriminating".
The reverse situation may occur in the case of an attribute, e.g. VERSION, which indicates a revision. The value of the attribute may have been set e.g. by a content management system or by an authoring system. In that case chapters have to be compared only if their revision numbers are different. In RevXml VERSION may be defined as being "negative discriminating".
The following actions are taken:

	if attribute values are equal:	*if attribute values are different:*
attribute is not discriminative	compare	compare
attribute is positive discriminative	compare	don't compare; nodes may be considered as different
attribute is negative discriminative	don't compare; nodes may be considered as equal	compare

An element may contain positive as well as negative discriminating attributes. Assume an element contains the attributes CH and VERSION as defined above. In this case the action table becomes

	if value of VERSION is equal	if value of VERSION is not equal
if value of CH is equal	!compare, equal	compare
if value of CH is not equal	!compare, different	!compare, different

Moreover, an element may contain any number of positive as well as negative discriminating attributes. The decision to compare or not is taken according to the following rules.
If for all positive discriminative attributes holds that the nodes have to be compared then the comparison has to take place; else the nodes are considered as being different. For ease of reasoning one may imagine that all values of pos. discr. attributes are appended together to form one value of one pos. discr. attribute.
If for all negative discriminative attributes holds that the nodes do not have to be compared then the comparison does not have to take place and the nodes are considered as equal. Again, for ease of reasoning one may imagine that neg. discr. attributes are being appended together to form one value of one neg. discr. attribute.
For the combination of the imagined resulting pos. and neg. discr. attribute the table above displays the decision.

The definition of discriminating attributes (and empty SGML elements) has to be made in the file RevXml.emp. The syntax is akin to the syntax of Xpath:

```
Emp_line            ::= '//' Element '[@ Wildcard]' (S '{' S
                        Discr_attr (S '|' S Discr_attr)* S '}')?
Element             ::= Name (Empty_indicator)? | Wildcard
Empty_indicator     ::= '\'
Discr_attr          ::= ('^')? ( Name | Wildcard )
S                   ::= (space | tab| newline)*
Wildcard            ::= '*'
```

In a future version of RevXml the syntax for the selection of elements and for the attributes within '[' and ']' will be extended, using the syntax of XPath.
Currently, elements with discriminating attributes have to be specified each on a separate line, where a wildcard stands for all elements that are not specified on other lines. Between square brackets all attributes have to be selected for processing by a wildcard. Between the braces the positive and negative discriminating attributes are specified. A wildcard is allowed. The wildcard stands for all attributes that are not otherwise specified between the same braces.

An elaborate example follows.

RevXml.emp contains:

//EL1\	indicates the empty element EL1 (for SGML)
//* [@*]{ @CH }	all elements that are not specified on other lines, with positive discr. attribute CH
//EL2A [@*] { @CH }	the element EL2A with positive discr. attribute CH
//EL3 [@*] { ^@VERSION }	the element EL3 with negative discr. attribute VERSION
//EL3A [@*] { ^@VERSION }	idem for element EL3A
//EL4 [@*]{ @CH | ^@VERSION }	the element EL4 with positive discr. attribute CH and neg. discr. attribute VERSION
//EL4A [@*]{ @CH | ^@VERSION }	idem for element EL4A
//EL4B [@*] {@CH | ^@* }	the element EL4B with positive discr. attribute CH and all other attributes as neg. discr. attribute
//EL4C [@*] { @CH | ^@VERSION }	idem for element EL4C
//EL4D [@*] { @CH | ^@VERSION1 | ^@VERSION2 }	the element EL4D with positive discr. attribute CH and neg. discr. attributes VERSION1 and VERSION2

Document **A** contains:

<EL2 A1=V1 CH=2>..EL2..</EL2><!-- compare -->
<EL2a A1=V1 CH=2>..EL2..</EL2a><!-- !compare, !equal -->
<EL3 A1=V1 VERSION=V3>..EL3..</EL3><!-- !compare, equal -->
<EL3a A1=V1 VERSION=V3>..EL3..</EL3a><!-- compare -->
<EL4 A1=V1 CH=2 VERSION=V3>..EL4..</EL4><!-- !compare, equal -->
<EL4a A1=V1 CH=2 VERSION=V3>..EL4..</EL4a><!-- !compare, !equal -->
<EL4b CH=2 VERSION=V3>..EL4..</EL4b><!-- compare -->
<EL4c A1=V1 CH=2 VERSION=V3>..EL4..</EL4c><!-- !compare, !equal -->
<EL4d A1=V1 CH=2 VERSION1=V1 VERSION2=V2a>..EL4..</EL4d><!-- compare -->
<EL4d A1=V1 CH=2 VERSION1=V1a VERSION2=V2a>..EL4..</EL4d><!-- !compare, equal -->

Document **B** contains:

<EL2 A1=V1 CH=2>..EL2a..</EL2><!-- compare -->
<EL2a A1=V1a CH=2a>..EL2a..</EL2a><!-- !compare, !equal -->
<EL3 A1=V1 VERSION=V3>..EL3a..</EL3><!-- !compare, equal -->
<EL3a A1=V1a VERSION=V3a>..EL3a..</EL3a><!-- compare -->
<EL4 A1=V1 CH=2 VERSION=V3>..EL4a..</EL4><!-- !compare, equal -->
<EL4a A1=V1a CH=2a VERSION=V3>..EL4a..</EL4a><!-- !compare, !equal -->
<EL4b CH=2 VERSION=V3a>..EL4a..</EL4b><!-- compare -->
<EL4c A1=V1c CH=2a VERSION=V3a>..EL4a..</EL4c><!-- !compare, !equal -->

<EL4d A1=V1c CH=2 VERSION1=V1a VERSION2=V2a>..EL4a..</EL4d><!-- compare -->
<EL4d A1=V1c CH=2 VERSION1=V1a VERSION2=V2a>..EL4a..</EL4d><!-- !compare, equal -->

Document **C** becomes:
<EL2 A1=V1 CH=2>..EL2<?OPENADD>a<?ENDADD>..</EL2><!-- compare -->
<?OPENDEL><EL2a A1=V1 CH=2>..EL2..</EL2A><?ENDDEL><?OPENADD><EL2a A1=V1a
CH=2a>..EL2a..</EL2a><?ENDADD><!-- !compare, !equal -->
<EL3 A1=V1 VERSION=V3>..EL3a..</EL3><!-- !compare, equal -->
<?revattrs>A1 VERSION<?endrevattrs><EL3a A1=V1a
VERSION=V3a>..EL3<?OPENADD>a<?ENDADD>..</EL3a><!-- compare -->
<EL4 A1=V1 CH=2 VERSION=V3>..EL4a..</EL4><!-- !compare, equal -->
<?OPENDEL><EL4a A1=V1 CH=2
VERSION=V3>..EL4..</EL4A><?ENDDEL><?OPENADD><EL4a A1=V1a CH=2a
VERSION=V3>..EL4a..</EL4a><?ENDADD><!-- !compare, !equal -->
<?revattrs>VERSION<?endrevattrs><EL4b CH=2
VERSION=V3a>..EL4<?OPENADD>a<?ENDADD>..</EL4b><!-- compare -->
<?OPENDEL><EL4c A1=V1 CH=2
VERSION=V3>..EL4..</EL4C><?ENDDEL><?OPENADD><EL4c A1=V1c CH=2a
VERSION=V3a>..EL4a..</EL4c><?ENDADD><!-- !compare, !equal -->
<?revattrs>A1 VERSION1<?endrevattrs><EL4d A1=V1c CH=2 VERSION1=V1a
VERSION2=V2a>..EL4<?OPENADD>a<?ENDADD>..</EL4d><!-- compare -->
<?revattrs>A1<?endrevattrs><EL4d A1=V1c CH=2 VERSION1=V1a
VERSION2=V2a>..EL4a..</EL4d><!-- !compare, equal -->

2.5 Supported XML/SGML constructions

The documents **A** and **B** should be XML documents or non-minimized SGML documents. That means that the SGML minimization constructions: tag omission, short references and datatag should have been resolved, for instance by a SGML parser.
The documents may start with an XML DTD or a SGML declaration and/or a DTD. In the last case the differences between the DTD's will also be tracked. At the other hand it is not required that a document conforms to a DTD. For SGML a list of tagnames is required which function as EMPTY elements. All other tags are assumed to appear in pairs. The XML convention for empty tags, e.g. <A/>, is supported. For SGML, the syntax of the tags and attributes themselves should conform to the reference concrete syntax of ISO 8879. Therefore, it is assumed that the tag delimiters are '<' and '>', as in XML.
It is also possible to compare flat files without any markup.

2.6 Technique

RevXml uses an extension to the known techniques for the determination of the longest common subsequence of **A** and **B**. It calculates the least number of edit-operations which are required to come from **A** to **B**. The extension concerns the applicability to tree-structured texts instead of flat texts. The functionality of RevXml has been influenced and refined by a number of customers. Propriatry techniques for caching, preprocessing, the handling of discriminating attributes and the implementation in C make RevXml one of the fastest and functionally most advanced products in the market for revison tracking of XML and SGML documents. Customers reported that RexXml can handle documents of the size of 50 MB or more.

2.7 Customers

RevXml, or its predecessor Sgdiff, has been sold, without marketing, to companies like Arbortext, Cisco, Citec, Corena, Fokker, Haufe Verlag, Pharmasoft, Wolters Kluwer and Wolters Samsom. They are satisfied with the product, as may be verified by their testimonials. Some installations are running for their tenth year. The change tracking capability of Arbortext's Epic Editor 4.2 is based upon the technology of RevXml. From 2002 on the product is offered to the general XML market.

3 Embedding in the environment

The program, called RevXml, is written as a function in ANSI C.
On the Windows platform RevXml is delivered as a dll, on other platforms as an object file.
RevXml can be called in two modes (indicated by the compiler switch "#define operate_on_files"):

1. operating on external documents, with a call to the function "revxml_on_files"
2. operating on internal buffers which contain the documents in internal memory, with a call to the function "revxml_on_buffers"
   - for demonstration purposes a C function RevXml_clnt is provided which reads the documents **A** and **B** first into internal buffers; care is taken to remember the positions of new lines; a new line is, if appropriate, replaced by a blank
   - RevXml is called as "revxml_on_buffers"
   - the file C is written, with new lines on the same positions as in file B.

RevXml_clnt is provided as an example. It demonstrates the use of the two API's.
RevXml and its predecessor SGDIFF have been ported to a number of Unix flavours and to an IBM mainframe, where a Cobol program holds the tree structures and calls the C function RevXml.

4 Use of the program

The program RevXml_clnt runs in batch under Windows95/98/Mill/NT/2000 and under Solaris.

It can be called as follows:

RevXml_clnt DIR A B **C**

where:

- 'DIR' is the directory where the files A, B and C reside
- 'A' is the filename of the original document (input)
- 'B' is the filename of the revised document (input)
- 'C' is the filename of the revised document with revision markers (output)

Input to RevXml_clnt are:

- the documents **A** and **B** as XML or SGML files
- a file called 'RevXml.emp' which contains the names of empty tags and discriminating attributes, in uppercase
- a file called 'RevXml.def' which contains the settings
- a file called 'RevXml.lic' which contains a license key

Output from RevXml_clnt are:

- the document **C** (= the document **B** with revision markers)
- a logging file, called 'RevXml.log'
- eventually a tracing file, called 'RevXml.trc'

- a cache memory file, called 'RevXml.cm'. Usually this file will be deleted upon a normal exit. However, if RevXml has been halted manually there is a possibility that it is not deleted. Upon the next execution RevXml will try to delete the file. The file may also be deleted manually.

In the file RevXml.log a report is given about the specified numbers and the amounts of memory which have been actually used.
As already described: the empty elements and discriminating attributes have to be summed up in the file RevXml.emp, in uppercase.

RevXml_clnt gives a report on the elapsed time so that measurements may be done. Please note that the first time that RevXml_clnt is run with a large amount of heap space Windows will use its time to set up the required swapping space, if it did not claim it before.

5 Settings

'**RevXml.def**' is assumed to be the name of the file which contains a number of settings. An example of RevXml.def is (with linenumbers for the subsequent explanation):

```
 1: <del>                  revision starttag for deletions from A
 2: <enddel>               revision endtag for deletions from A
 3: <add>                  revision starttag for additions from B
 4: <endadd>               revision endtag for additions from B
 5: <chg>                  revision starttag for changes
 6: <endchg>               revision endtag for changes
 7: <revattrs>             revision starttag for attributes
 8: </revattrs>            revision endtag for attributes
 9: combtags               combine revision tags, e.g. del+add into
                           chg
10: noignore_diff_in_tags  do not ignore the difference of tag names
11: nocompare_pi           do not compare processing instruction
12: nocompare_comment      do not compare comment
13: printa                 print also the deleted text of A
14: 2                      atomic_unit_of_text: 1=char 2=word
                           3=sent. 4=pcdata
15: preproc                preprocess with
                           atomic_unit_of_text=pcdata
16: notrace                no tracing required in file RevXml.trc
17: 2000000                textsize
18: 5000000                heapsize in bytes
19: 200000                 d_size in int (= 4 bytes); 4*d_size <
                           heapsize
20: 300                    max_nr_of_names_of_elements
21: 300                    max_nr_of_names_of_attributes
22: 2000                   max_nr_of_names_of_attribute_values
23: 64                     pagesize (APS) for cache memory buffer; 1
                           page will be of size APS * 512 bytes
```

The settings within RevXml.def are organized as follows.

- The settings have to be stated in a fixed sequence
- On a line only the string up to the first space is significant
- On the first 6 lines the revisionmarkers which will appear in document C are listed, e.g.:

    <del> -- start of a deletion

<enddel> -- end of a deletion (could also be </del>)

<add> -- start of an addition

<endadd> -- end of an addition (could also be </add>)

<chg> -- start of a change

<endchg> -- end of a change (could also be </chg>)

The revision markers may be changed into other strings with one restriction: the length of all the 3 start markers should be the same and the length of all the 3 end markers should be the same. The replacing strings may be arbitrarily chosen; e.g. as processing instructions

- Lines 7 and 8 contain the strings which will be used for the indication of a change in the name(s) and/or the value(s) of attributes.

<revattrs> -- start of a list of changed attributenames or -values

</revattrs> -- end of the list of changed attributenames or -values

These names may be changed also, e.g. in “<?revattrs“ and “>”

- Line 9 contains a switch which indicates if a sequence of '<del></enddel><add>Y</add>' has to be replaced by '<chg>Y</endchg>': 'combtags' or 'nocombtags' (the switches 'combtags' and 'printa' can not be combined)
- Line 10 contains a switch which indicates if the contents of unequal tags have to be compared or not: 'ignore_diff_in_tags' or 'noignore_diff_in_tags'
- Line 11 contains a switch which indicates if processing instructions (PI) have to be compared or not; if not, in the marking of the deleted text for doc A the PI's of A are left out; also, added PI's to doc B are not marked as being added
- Line 12 contains a switch which indicates if comment has to be compared or not; the treatment is the same as for PI's
- Line 13 contains a switch which indicates if the deleted text of A has to be included in C, or not: 'printa' or 'noprinta'
- Line 14 indicates the atomic unit within the document: 1 = char 2 = word 3 = sentence 4 = pcdata
- Line 15: the switch for preprocessing, e.g.: ‘preproc’ or 'nopreproc'; preprocessing improves the runtime for larger documents
- Line 16: the switch for tracing on and off, e.g.: 'trace' or 'notrace'; tracing produces a lot of output
- Line 17: the maximum size of a text buffer (variable "textsize"); should be an upper bound for the size of the documents in bytes
- Line 18: the maximum amount of heap space to be used; RevXml has its own memory manager; at the start it allocates the maximum amount of heap space and reuses it in a stack wise manner; furthermore, a cache memory system is used; 50 MB should be sufficient, even for large documents
- Line 19: the maximum size of the stack for matrices d (variable "d_size"); 200K should be enough; however, very large chunks of #pcdata with many comparisons on the character level may require more; in that case RevXml will give a message.
- Line 20: the maximum number of different names for tags
- Line 21: the maximum number of different names for attributes

- Line 22: the maximum number of different attribute values
- Line 23: the pagesize for cache memory, in multiples of 512; 64 should be enough; however, very large chunks of #pcdata with a lot of comparisons on the character level may require more; in that case RevXml will give a message.

RevXml gives a report on the elapsed time so that you can do your own measurements. Please note that the first time you run RevXml with a large amount of heap space the operating system will use its time to set up the required virtual memory, if it did not claim it before.

6 Terms and conditions

RevXml is meant to cooperate with other systems. It is envisioned that some adaptations may be required, for instance for a change of

- the functionality, like the indication of differences in attributes;
- the manner of exchange of information with the calling program;
- the platform, or even the programming language.

These adaptations can be made by Palstar. Please consult the website of Palstar, www.palstar-nl.com, for the conditions.

To all the offers and agreements of Palstar bv the General Terms and Conditions of Business of FENIT are applied, unless deviations are stated expressly in writing. (FENIT is the Federation of Dutch Trade Associations for Information Technology.)

7 Contact

Dr Gert J. van der Steen, Palstar bv,
Winkelsteeg 5a, 7975 PV Uffelte, The Netherlands.
Tel. +31-654774547, Fax. +31-521351078.
E-mail: info@palstar.nl
Website: www.palstar-nl.com