Previous Analysis of ODF 1.2 Deletion Change-Tracking Model
In demonstrating the implementation of single deletions that tear (and heal) the hierarchical XML structure, it was seen how the healing of excisions occurs to produce well-formed XML fragments in the `<text:deletion>` element. In addition, there is healing across the scar that is left at the point of excision. Orchestrating both of those properly allows acceptance of the deletion and rejection of the deletion to occur without incident.

By coincidence, the chosen single-deletion case also demonstrated a bug in deletion tracking. In a variety of deletions, there can be silent corruption of the `<text:deletion>`, with inability to restore the original text if the deletion is rejected.

The Interesting Question of Insertion-Deletion Symmetry
Change-tracked insertions appear to be far simpler than tracked deletions. Insertion is certainly easier to describe in the specification. A marker is placed at the beginning of the inserted material. Another marker is placed at the ending of the inserted material.

But wait, these markers are not tied to the XML hierarchy. And they represent two scars: One that is healed to bridge into the insertion from before the start point, one that is healed from the insertion to beyond the end point.

This note explores how insertions and deletions have similar complexity and require comparable precision of specification for ODF 1.0-1.2 change-tracking to be successfully and independently implementable.

Format of Change-Tracking Document Samples
The present document includes three examples by which the same insertion is attempted. To observe how the change tracking is presented by a particular ODF implementation, you will need to use an ODF 1.0-1.2 consumer that implements tracked changes. Although this document was produced with changes set to visible, you may need to adjust settings in your software to see the changes. If you cannot observe the change-tracking, the narrative and screen captures present the whole story either way.

Because there have been change-tracking bugs and fixes over releases of ODF implementations, you may or may not observe the same tracked changes that were observed by the author.
For reference and comparison, there are also screen captures of various stages of the tracked-change material. In those samples, the following conventions apply:

- **Insertions are shown with** green highlighting
- **Deletions are show with** red highlighting
- **Font changes shown with** double-underlining
- **Comments provide** observations about the changes

With software in the OO.o code-base family, you can use option settings such as those above to view the changes in this document in the same form as in the screen captures.

This document was produced using Libre Office 3.3.2 on Microsoft Windows 7. It is identified as an ODF 1.2 Document.

**The Basic Insertion Idea**

The idea is to create an insertion that is essentially the symmetric counterpart of the deletion that was explored in the *Forensic Analysis of ODF Deletion Tracking*. Instead of taking a three-item list and following paragraph down to a two-item list, we go in the opposite direction.

We start with a two-item list and insert, in a single/continuous operation, the text and styling that creates a three-item list and following paragraph of text.
We start with

1. This is list-item-a

2. This is text following the list.

We want to end up with

1. This is list-item-a

2. This is list-item-b

   This is some text hanging below the list-item-b

3. This is list-item-c

   This is text hanging below the list-item-c

   This is a paragraph of text following the list.

**Directly-Typed Insertions**

One way to make the insertion is to place an insertion point right after “is” in the second list item and introduce the desired text from the keyboard. I think of this as the least likely to work.

1. This is list-item-a

2. This is list-item-b

   This is some text hanging below the list-item-b

3. This is list-item-c

   This is text hanging below the list-item-c

   This is a paragraph of text following the list.

It took some effort, but the result has the intended appearance. We won't know how much this insertion is similar to a deletion reconstruction until we look deeper into the XML of the document.

[Note: The Forensic Analysis of Deletion Tracking did not attempt to do a text character-by-character deletion from the keyboard. This points out a valuable case that I missed.]

Here's what the change-tracked presentation of the insertion was when I made it:
Directly-Typed Insertions

One way to make the insertion is to place an insertion point right after “is” in the second list item and introduce the desired text from the keyboard. I think of this as the least likely to work.

1. This is list-item-a

2. This is list-item-b

This is some text hanging below the list-item-b

3. This is list-item-c

This is some text hanging below the list-item-c

This is a paragraph of text following the list.

It took some effort, but the result has the intended appearance. We won’t know how much this insertion is similar to a deletion reconstruction until we look deeper into the XML of the document.

Before the Insertion

Here is the XML for the sample of text as it appears without any insertion:

```xml
<office:text text:use-soft-page-breaks="true">
    <text:list xml:id="list38237886" text:style-name="L4">
        <!-- This is the version of the list that is given without any insertions as the starting point of all of the attempts. Note how <text:list> is a top-level text-content child of the <office:text>. -->
        <text:list-item>
            <text:p text:style-name="P13">This is list-item-a</text:p>
        </text:list-item>
        <text:list-item>
            <text:p text:style-name="P13">This is text following the list.</text:p>
        </text:list-item>
    </text:list>
</office:text>
```

The markup for the initial list is straight-forward.
After the Insertion

In this case, the insertion is treated as three consecutive insertions. Because I backed out of insertions that weren’t creating new list items properly, and then resumed, I think my manual efforts triggered closing off of one insertion and then following by another. This is not remarkable.

What is important to observe in the markup for the insertions is that the start and end points of insertions are in sibling elements, not following the hierarchy of the XML itself:

```xml
<office:text text:use-soft-page-breaks="true">
  <text:changed-region xml:id="ct308338856" text:id="ct308338856">
    <text:insertion>
      <office:change-info>
        <dc:creator>Dennis Hamilton</dc:creator>
        <dc:date>2011-04-17T18:24:00</dc:date>
      </office:change-info>
    </text:insertion>
  </text:changed-region>
  <text:changed-region xml:id="ct308338960" text:id="ct308338960">
    <text:insertion>
      <office:change-info>
        <dc:creator>Dennis Hamilton</dc:creator>
        <dc:date>2011-04-17T18:25:00</dc:date>
      </office:change-info>
    </text:insertion>
  </text:changed-region>
  <text:changed-region xml:id="ct308339064" text:id="ct308339064">
    <text:insertion>
      <office:change-info>
        <dc:creator>Dennis Hamilton</dc:creator>
        <dc:date>2011-04-17T18:26:00</dc:date>
      </office:change-info>
    </text:insertion>
  </text:changed-region>
</text:tracked-changes>
```

Here is the copy of the two-item list that was modified by direct-entry insertion. Note that we now have a three-item list and a following paragraph. Some end tags had to be changed in order to accomplish that.

```xml
<text:list xml:id="list38361488" text:style-name="L6">
  <text:list-item>
    <text:p text:style-name="P15">This is list-item-a</text:p>
  </text:list-item>
  <text:list-item>
    <text:p text:style-name="P15">This is list-item-b</text:p>
  </text:list-item>
  <text:list-item>
    <text:p text:style-name="P15">This is list-item-c</text:p>
  </text:list-item>
</text:list>
```
What should be evident in this example is that major characteristics of deletions also apply to insertions:

1. Insertions can cut across the hierarchical structure of the XML that corresponds to the text structure of which the insertion is a part.
2. Insertions can require alteration of the structure of the XML beyond the end of the insertion in order to knit subsequent material into the structure that the insertion adds.

3. The revocation of an insertion can require different healing to knit that which precedes the start point of the insertion together with that which follows the end point of the insertion. This must reconstruct the text as it existed prior to the insertion having been made.

There's another facet of production-quality change-tracking as part of the interactive editing of a document. The software will recognize that runs of keyboard entries are treated as single actions, even though conducted through multiple keystrokes.

Deletions by repetitive backspace and delete keyings are also collected into single deletions.

Finally, the editing of an existing insertion or deletion tends to adjust the insertion or deletion and not lead to a pile-up of change-tracking with change-trackings.

None of these characteristics are determined by the ODF change-tracking provisions. It is simply the case that the change-tracking provisions do not interfere with delivery of such convenient behaviors.

For me, the most startling, taken-for-granted quality of production-quality change tracking is the amount by which the progressive entry of an insertion (or deletion) continually changes how text beyond the current end of the change has its XML structure modified in order to splice with the end point of the change.

That such transformation is also required to be reversible depending on change acceptance/rejection choices later is something that is generally far below the attention of the user. (I also assume that the internal model in which the under-modification document is held is organized to support such adjustments easily.)

**How About Copy and Paste?**

In the next example, a section of text that has the desired form is used for selection of material to copy into the short version of the list.

Here is the source:

1. This is list-item-x

2. This was list-item-b

   This is some text hanging below the list-item-b

3. This is list-item-c
This is some text hanging below the list-item-c

This is a paragraph of nonsense unrelated to the list.

The selection that was copied has been highlighted subsequent to the copy and paste.

Here is the destination and the insertion:

1. This is list-item-a

2. This is list-item-b

   This is some text hanging below the list-item-b

3. This is list-item-c

   This is some text hanging below the list-item-c

4. This is a paragraph of text following the list.

As can be seen with changes visible, this didn't work out so well. I had the notion that this might work better than the typed insertion, but I was mistaken:

There are some remarkable problems that resulted with this insertion,

- The indentation of the complete text has been lost with the list items numbered 2-3 brought left for no clear reason

- The fact that “This is a paragraph of” was not treated as being outside the list is perhaps understandable yet it seems that it should have been easily avoided.
There is some hang-over from the insertion that has new paragraphs be made into additions on the list. Although I forced “As can be seen” to not be a list item, the software continued to propose new paragraphs as continuations of the list. (There may be a related bug where if a list is copied to a later part of the document, the copy is renumbered as if it is a continuation of the original list. I also confess to not noticing that there is more than one place to turn off list-creation.)

My initial speculation was that insertion has many of the qualities exhibited by the revocation of a deletion. That would, I speculated, be an interesting way to bundle up a copied selection and then make it an insertion.

This is an imperfect notion, because the insertion point does not reflect the systematic healing process of having had that material excised. Without that to count on, the insertion requires some other way to convey the structural characteristics of the source text for reconciliation at the point of insertion.

I still expected the paste case to be closer to the directly-typed insertion. Considering how badly it worked, I wonder why the operation was allowed.

This case is sufficiently discouraging that I don't want to know what the XML looks like.

**Cut and Paste Perhaps?**

This version should be self-explanatory after what has gone before:

1. This is list-item-x

2. This was list-item-b
   - This is some text hanging below list-item-b

3. This is list-item-c
   - This is some text hanging below list-item-c

4. This is a paragraph of nonsense unrelated to the list.

(This time I made certain to ensure that the last line is not viewed as part of the list, an error that may be invisible but influencing the previous result.)

1. This is list-item-a

2. This is list-item-b
   - This is some text hanging below list-item-b
3. This is list-item-c
   This is text hanging below list-item-c

4. This is a paragraph of text following the list.

It is clear that the origin of pasted material doesn't seem to influence the pasted insertion any differently. Unfortunately, the already-identified defect in deletion tracking arises above, as expected.

When the text was cut, there was not indication of a list-item numbered 4. This is how things looked until this document was saved:

| 1. This is list-item-x
| 2. This was list-item-b
| 3. This is list-item-c
   This is some text hanging below list-item-c
   This is some text hanging below list-item-c
   This is a paragraph of nonsense unrelated to the list.

(This time I made certain to ensure that the last line is not viewed as part of the list, an error that may be invisible but influencing the previous result.)

| 1. This is list-item-a
| 2. This is list-item-b
| 3. This is list-item-c
| 4. This is a paragraph of text following the list.

We have shown the basic nature of insertions and how they cross hierarchies. We have also shown that pasting of insertions does not bring over text structure from the source in any predictable manner. I thought to find more magic. Instead, chaos.