Tracking Changes: Technical and UX Challenges

March 13nd 2010

laurens@xopus.com

Introduction

- Laurens van den Oever
- CEO, Xopus BV

- The browser based XML editor ...
- ... for non-technical authors.

Introduction

Today:

- The Challenges of Change Tracking
  - Technical
  - User Experience
- Why Xopus has no full implementation (yet)

Why Would We Track Changes?

- Collaboration

  While authoring

- Reviewing

  After authoring

- Auditing

  Any time

Collaborating

- Supports creative process
- Large group of authors
- Typically working consecutively
  - Maybe even realtime
- Communication method
- Accept/reject functionality not as important
  - No final responsibility
- Authors can see tracked changes

Reviewing

- To ensure quality
- Author/editor roles
- Accept/reject functionality the goal
  - Editor has final responsibility
- Authors may not see tracked changes

Auditing

- History of all changes
  - Including deleted content
  - Rejected inserts
- Detail and reliability is vital
- Accept/reject functionality irrelevant
- Authors may not see tracked changes

Tracking vs Comparing

- Comparing (diff)
  - Compares two documents
  - Works on existing content
  - Supports a version tree
- Tracking
  - Supports multiple authors
  - Must be built into editing tool
  - Consecutive versioning
  - No canonical XML issues

Tracking vs Comparing

V1: “A sentences in need of desparate rephrasing”

“A sentence~~s~~ i~~n ~~n~~eed of ~~desp~~a~~erate need of rephrasing.”

“A sentence~~s~~ in desperate need of~~ desparate~~ rephrasing.”

V2: “A sentence in desperate need of rephrasing.”

- Compare doesn’t ‘understand’ the change

Applying changes

- Compare (typical)
  - 3 way merge
  - Line based
  - Can convert to change tracking markup

- Accept/reject in the UI
  - Integrated in familiar authoring interface
  - Adds author and date, maybe comment info

Tracking vs Validation

- Store changes either in
  - Processing instructions or in
  - Elements

- Attribute changes stored in attributes
  - Only complete changes
    (there is no attr substructure to store more)
  - So prevent content in attributes

Processing Instructions

```
<p>A<?begin?> new<?end?> sentence</p>
```

- No validation interference
- Difficult to render using XSL
- Hard to maintain integrity
- Easy to ‘accept’: remove all PIs
- No standard to access substructure
  (‘attributes’, ‘child nodes’)

Elements

```
<p>A<ct:add> new</ct:add> sentence</p>
```

- Interferes with validation
- Easier to render using XSLT
- Inheritant integrity
- DOM access substructure

Elements

```
<meta>
   <creationDate>...</creationDate>
   <ct:add><description/></ct:add>
   <keywords/>
</meta>
```

- Hard to validate order
- Filtering?
  - Not with realtime validation

Real vs Perceived Changes

- Not all changes are atomic DOM actions.
- Authors perceive one interaction as one change.
- Example:

```
<p>Sentence on no particular subject. |Followed by
   another random sentence.</p>

<p>Sentence on no particular subject.<split1/></p>
<p><split2/>|Followed by another random sentence.</p>
```

Real vs Perceived Changes

- Event based DOM API makes it harder
- Contract numbering example:

```
<list>
  <item><nr>1.</nr><p>An item</p></item>
  <item><nr>2.</nr><p>Another item|</p></item>
  <item><nr>3.</nr><p>Last item</p></item>
</list>
```

Real vs Perceived Changes

```
<list>
  <item><nr>1.</nr>
    <p>An item</p></item>
  <item><nr>2.</nr>
    <p>Another item<split1 id=“a”/></p>
    <split1 id=“b”/></item>
  <item>
    <split2 id=“b”/>
    <add><nr>3.</nr></add>
    <p><split2 id=“a”/>|</p></item>
  <item>
    <nr><del>3.</del><add>4.</add></nr>
    <p>Last item</p></item>
</list>
```

Real vs Perceived Changes

- Possible to handle changes individually
- But that is not logical to author
  - Why reject only change 3. -> 4.?

- Also consider adding a table column
- Followed by adding a new row
  - Reject one cell?
  - Row insertion before the column?

Changes vs WYSIWYG

- Challenges rendering changes WYSIWYG
  - WYSIWYG is result of transformation
  - Changes live in source XML
  - Author lives in transformed output
- Approaches to get changes in the output
  - Compare output
  - Adapt transformation
  - Magic

Changes vs WYSIWYG

- Compare output
  - Hidden changes are lost (metadata)
  - Stylesheet output is treated as content (auto numbers, generated text)
  - Sorting and multiple occurences interfere
  - Output far more complex than input

Changes vs WYSIWYG

- Adapt transformation
  - Requires rich transformation language

    CSS may not suffice
  - Makes (XSL) transformation more complex
    - How do we maintain it?
  - When do we have full coverage?

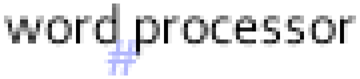
Changes vs WYSIWYG

- Magic
  - Black box feature of the authoring tool
  - Xopus keeps change info separately
    - No validation interference
    - No transformation interference
  - Changes are rendered over WYSIWYG output
  - Can’t render deleted content (~~strike through~~)

Exotic Requirements

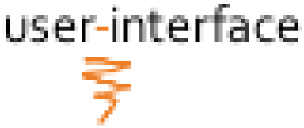
Small changes may be very important:

- Added whitespace



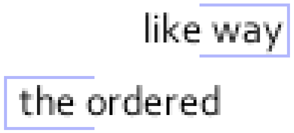
word processor

- Added punctuation



user-interface

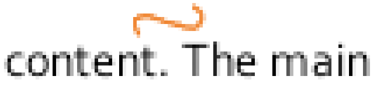
- Split



like way

the ordered

- Merge



content. The main

Exotic Requirements

- Auditing has interesting requirements
- Keep information about
  - Rejected inserted content

    By whom, when and why
  - Accepted deleted content

    Again: by whom, when and why
- In general in must answer why
  - Content is published
  - Content is not published

Concluding

- Tracking changes very complex
- No one size fits all solution

- We continue our research
- Welcome your feedback

Questions?

March 13nd 2010

laurens@xopus.com