Taming the ODF Dragon: Complexity Reduction by improved Tooling

Svante.Schubert@gmail.com

ROME | 11 October 2017
Agenda

- Why?
- What?
- How?
Why?
Why? Leverage ODF standard!

- ODF is..
  - Blueprint for all ODF applications
  - Telling what are valid ODF documents
  - An OASIS and ISO standard (government argument)
  - created against vendor lock-in (interoperability by format)
What?
Improve ODF for developer!

- Ease ODF application development
  - Safe time by avoid repeated work!
  - Reducing complexity!

- Share as much as possible among ODF app developers..
  - Feature tests and test documents at standard level (similar JPEG2000 ISO standard)
  - Make specification machine readable..
  - Generate (as much as possible)
<table>
<thead>
<tr>
<th>Illustration</th>
<th>Krups Countline KH442</th>
<th>Tefal TT 5500</th>
<th>Severin AT 2509</th>
<th>Severin AT 2514</th>
<th>WMF Stello Toaster</th>
<th>Severin AT 2287</th>
<th>Grundig TA 6330</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>comparison result</strong></td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
<td>Vergleich.org</td>
</tr>
<tr>
<td>Review</td>
<td>1.2</td>
<td>1.3</td>
<td>1.5</td>
<td>1.6</td>
<td>1.7</td>
<td>1.8</td>
<td>Excellent</td>
</tr>
<tr>
<td>Rating</td>
<td>very good</td>
<td>very good</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>Good</td>
<td>good</td>
</tr>
<tr>
<td>Date</td>
<td>09/2017</td>
<td>09/2017</td>
<td>09/2017</td>
<td>09/2017</td>
<td>09/2017</td>
<td>09/2017</td>
<td>09/2017</td>
</tr>
<tr>
<td><strong>Customer rating at Amazon</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Reviews</td>
<td>4</td>
<td>564</td>
<td>2</td>
<td>983</td>
<td>158</td>
<td>511</td>
<td>44</td>
</tr>
<tr>
<td><strong>Sheet per pass</strong></td>
<td>6</td>
<td>8th</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td><strong>Power (watts)</strong></td>
<td>720 watts</td>
<td>1,200 watts</td>
<td>1,400 watts</td>
<td>850 watts</td>
<td>900 watts</td>
<td>700 watts</td>
<td>850 watts</td>
</tr>
<tr>
<td><strong>Dimensions (LxWxH)</strong></td>
<td>33.2 x 24.4 x 20.2 cm</td>
<td>40 x 23.4 x 22.8 cm</td>
<td>12.5 x 37.1 x 18.2 cm</td>
<td>27.1 x 15.5 x 18.3 cm</td>
<td>32.5 x 20 x 27.5 cm</td>
<td>32 x 18 x 18.5 cm</td>
<td>34 x 21.5 x 24.5 cm</td>
</tr>
<tr>
<td><strong>mass</strong></td>
<td>0.8 kg</td>
<td>3.1 kg</td>
<td>1.0 kg</td>
<td>1.5 kg</td>
<td>1.9 kg</td>
<td>1.2 kg</td>
<td>2.0 kg</td>
</tr>
<tr>
<td><strong>Heat insulation</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>rolls rust</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Broad disc centering</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Toast lifting function</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>defrost</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>manual stuffing function</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Testzüger Foundation Warentest 04/2016</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Including egg cooker and egg paws</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Housing is not hot</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Also suitable for bread</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Very quiet</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Tans evenly</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Illuminated key labels</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Integrated roll holder</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Good workmanship</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Stable crumb tray</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
CSS Paged Media (@page)

CSS at-rule (@page) to define page-specific rules when printing web pages, such as margin per page and page dimensions.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>iOS Safari</th>
<th>Opera Mini</th>
<th>Android Browser</th>
<th>Chrome for Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>11</td>
<td></td>
<td></td>
<td>10.1</td>
<td>all</td>
<td>4.4</td>
<td>61</td>
</tr>
<tr>
<td>Edge</td>
<td>15</td>
<td>56</td>
<td>11</td>
<td>11</td>
<td></td>
<td>55</td>
<td>61</td>
</tr>
<tr>
<td>Firefox</td>
<td>56</td>
<td>61</td>
<td>47</td>
<td>11</td>
<td></td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>Chrome</td>
<td>49</td>
<td>60</td>
<td>47</td>
<td>11</td>
<td></td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>Safari</td>
<td></td>
<td>10.1</td>
<td>47</td>
<td>11</td>
<td></td>
<td>55</td>
<td>61</td>
</tr>
</tbody>
</table>

Notes

Currently no browsers appear to support the `marks` & `bleed` properties from the latest version of the specification.

* Does not support the `size` property.
Acid3

94/100

To pass the test, a browser must use its default settings, the animation has to be smooth, the score has to end on 100/100, and the final page has to look exactly, pixel for pixel, like this reference rendering.
We shall aim for..

- Conformance comparison of ODF Applications (like Toasters)
- Automated ODF conformance tests (like ACID3 for HTML/CSS)
  - only work as *HTML has a browser Run-Time-Model (DOM)*
- No *ODF XML Run-Time-Model*, but common *Semantic Model*
- Need for *ODF Feature* description
  - There is more than a monolithic ODF block
- Need for *ODF Feature change* description
  - User already expect to change ODF documents
  - Do no longer 'send the repository', send an ODF DIFF
- Some feature example: Page layout… (see PageLayout.odt)
An ODF feature example..

- Page Layout test document (PageLayout.odt)
  - Two pages (one portrait, one landscape layout)
  - Three paragraphs (two on first page, third on second)
  - Different header/footer on each page
Hello World!
The Semantic Model vs. XML

- **ODF XML Logic (XML Model)**
  - Paragraph → Style → Master Page → Page Layout → Text/Size

- **User Logic (Semantic Model)**
  - Align *paragraph* with *page layout* (*header & footer* and *size*)
  - Decouple XML from user logic
  - Define feature machine readable in spec & grammar
Specify Semantic Model!

- Define semantic model in ODF spec & grammar
  - How does the ODF XML change, when we add / modify / delete the feature
  - Allow to send DIFF instead of documents
  - Make Libreoffice Online, Office365, Google Docs and GIT interoperable!

- Enable to generate the model from the ODF spec & grammar
  - How the run-time-model is changed when adding / modifying / deleting the feature
  - Import & export of ODF XML into application are dependent like inverse functions and should be able to be generated as well
Specify Semantic Model!

- Problem:
  - ODF grammar is more than 18,000 lines text file
  - Hard to read / analyze
How?
Tricks to analyse ODF

- **Original Idea**: Chaos Computer Congress on Source Code Analysis

  — Instead of using 'grep' on source code text files

  — Mapping source code semantic to graph database (*TinkerPop 3*)

  — Analyze the graph with stored procedure (*Gremlin script*)
Tricks to analyse ODF

- **Original Idea:** Chaos Computer Congress on Source Code Analysis
  - Instead of using 'grep' on source code text files
  - Mapping source code semantic to graph database (*TinkerPop 3*)
  - Analyze the graph with stored procedure (Gremlin script)

- **Copied Idea:** Transformed to our ODF domain:
  - Instead of looking into *ODF grammar text file* (>18k lines)
  - Map the ODF grammar into a graph database (*Tinkerpop 3*)
  - Analyze the ODF grammar graph with reproducible queries (using Gremlin script within GraphDB)
How? - Mapping into GraphDB

- Map the ODF grammar into a graph database (Tinkerpop 3)
  - ODF grammar (RelaxNG XML) file is the source
  - Instead of parsing and resolving it myself, reuse OpenSource MultiSchemaValidator and parse its memory dump :)
  - Written a tool by using ANTLR 4.0 (parser generator) mapping the MSV input to a Graph XML file (to load into any GraphDB)
    
    see JIRA #458 of Apache ODF Toolkit (incubating)
Reproducible Analysis

- Analyze the ODF grammar graph
- Having reproducible queries
  - using Gremlin script within GraphDB
  - What is the difference between a heading (text:h) and paragraph (text:p)?
  - Is it possible that paragraphs are being nested?
  - What is the minimal text document?
  - ...

@Developer: To Do List

- Identify all ODF patterns (semantic model, feature change API)
- Make all ODF information machine readable in the specification
  - e.g. annotate ODF elements in grammar, which start a logical user object
- Generate as much as possible from ODF spec
  - Move feature tests and test documents to ODF standard
Thank you!