Due to Popular Demand - SVG Import!

A long story in 45 minutes

Fridrich Strba & Thorsten Behrens
Novell, Inc.

Novell®

Why SVG Import?

- users *desperately* want it in OOo – missing SVG import is one of the highest-voted issues
- and then some:
  - the most prevalent format for vector cliparts
  - the most prevalent vector graphics exchange format
  - an Open Standard and
    - native format for one of the best vector drawing apps out there – Inkscape
  - a subset of SVG is even copied in ODF for the drawing shapes

The History

- In the beginning, there was `void` SVG support
- Then came about Issue 2497
- Followed by WindLi's XSLT implementation
- And Bernhard Haumacher's excellent Batik-based import extension
- The final wisdom is in Issue 49991
- Not to forget the piece presented here: the built-in component

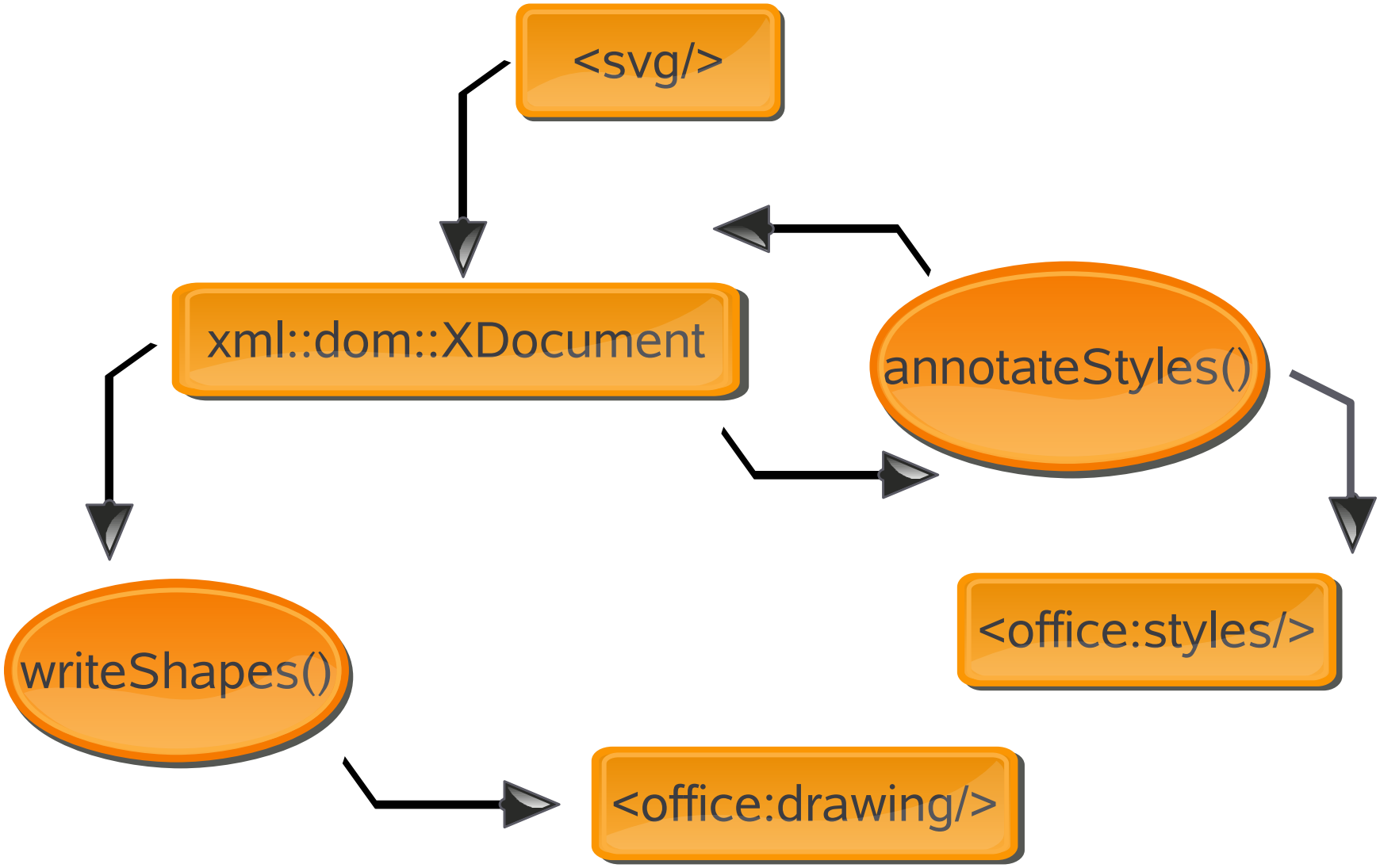
Why XSLT Won't Work Out

- ODF & SVG are mutually incompatible:
  - transformation not possible by mapping xml entities
  - instead, *graphical* entities need to be mapped, sometimes with intimate knowledge of pixel & geometry
- though ODF re-uses a subset of SVG, semantics & expressiveness is different enough to make xml-based conversions fail

What Did We Do?

- Took generic SAX DOM parser
- Two-pass DOM tree visiting:
  - First pass collects style information, serializes styles to ODF & annotates elements with style names
  - Second pass maps SVG shapes to ODF shapes, possibly emulating via polygons
- No need for custom xml parser, very quick results
- Easy to extend over time, no cross-cutting changes necessary

Architecture



Cliparts from: http://openclipart.org/people/skadge/skadge_SVG_widgets_for_diagrams.svg

DEMO: ODF Import

Why ODF Import is Suboptimal

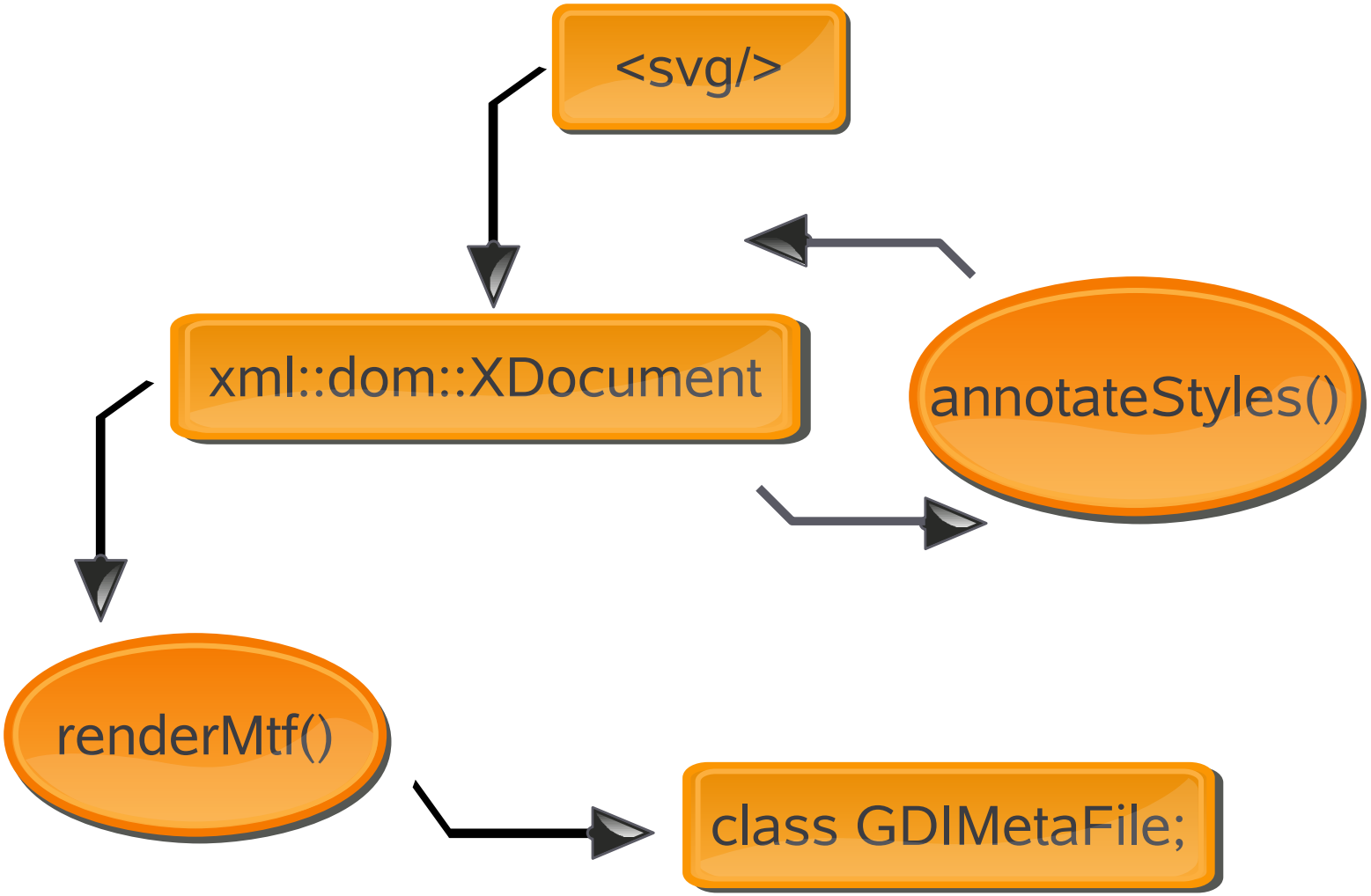
- Inevitable information loss:
  - Inferior gradients
  - Inferior fill rules
  - Inferior line stroking & dashing
  - Inferior line end markers
  - Grouping much less expressive
  - No 'template' shapes
  - No clipping
  - No filters
- i.e. all non-representable features need to be emulated via custom rendering

SVG Graphic Import

The “Insert->Picture” variant

- most code shared with ODF import
- Adding render commands to a metafile, instead of streaming ODF
- Yields an atomic, non-editable image like PNG or EMF

Architecture



Cliparts from: http://openclipart.org/people/skadge/skadge_SVG_widgets_for_diagrams.svg

DEMO: Graphics Import

What's Wrong With Graphic Import?

- Duplicates effort – librsvg is there & works like a charm
- Problem is, OOo does not easily lend itself to custom graphic renderers
  - Historically, all OOo graphic filters convert to either
    - Bitmap, or
    - StarView Metafile (comparable to WMF)
  - There are two notable exception to the rule:
    - EPS tunneling for PostScript printers
    - Rodo's EMF+ import, rendered via Xcanvas
- Hacking librsvg to pseudo-render to StarView Metafile is a waste – too little expressiveness

Planned Graphic Import Improvements

- better gradient & text support for the ODF import
- use librsvg to render directly:
  - either with the cairo backend
  - or via rendering::XCanvas
- get Sun to accept it under non-SCA conditions

Roadmap

- Improve SVG to ODF mapping on demand
- bin SVG to SVM mapping mid-term, and replace with librsvg binding
  - if possible, provide a generic render plugin mechanism for other vector formats:
    - WMF/EMF (join forces with Abiword/libwmf)
    - EMF+
    - SVG (librsvg)
    - EPS/PDF (gs/poppler)
    - SVM (extract own code to lib)
    - PICT (extract own code to lib)
- Long-term, decouple all graphic filters from core OOo

Q & A