OOXML: NOW WHAT?

PRE-BRM DISCUSSION

CT 173 / Portugal
01/2008

DISPOSITIONS

- over 6000 pages
- 3522 comments
- around 1000 unique issues
- around 2000 pages of dispositions
- 118 dispositions for Portugal

DISPOSITIONS

- 70% of PT comments correctly addressed
- many **critical** issues pending
- clarifying discussion took place
- most concerns relate to interoperability, scope and correctness of OOXML

FACTS

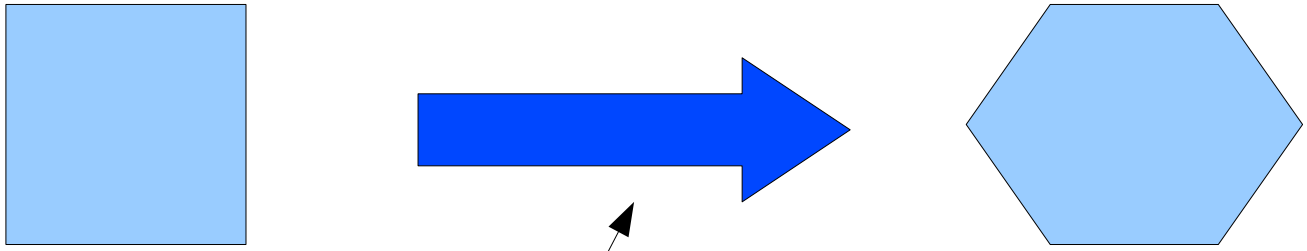
- very powerful incumbent
- deprecated features only recently documented: not enough time to evaluate correctness and feasibility
- application defined behavior opens the way for potential abuse of dominant position (*hmm, let's quietly change the behavior on the next service pack...)*

WHY DO WE NEED DEPRECATED FEATURES?

TO REPRODUCE OBSOLETE BEHAVIOUR?!

DEPRECATED BEHAVIOUR SHOULD BE MAPPED IF DESIRED AT IMPORT TIME

here we had deprecated behavior, but no specific tags for it (it's by design/bug)



legacy format

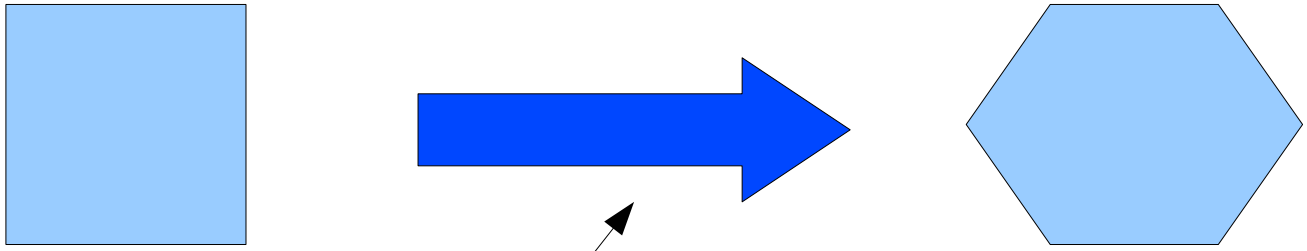
OOXML 1.0

the import filter should compensate the deprecated / buggy behavior by adapting the resulting XML so that the final document looks the same as seen with the old software
(unless proven impossible, should be checked individually)

here we should have correct behaviour as a function of the XML content

EXAMPLE

Word 95 used to incorrectly insert/forget spaces



legacy format

OOXML 1.0

Instead of AutoSpaceLikeWord95 tag on OOXML the import filter should insert/remove the necessary spaces on the non-deprecated XML
(can be made optional via import GUI)

spaces will be correctly inserted – there will be a different aumount than on the original document to render the same result

WHY

- a brand new standard should not implement bugs where unnecessary
- each behavior a valid OOXML producer is allowed to implement sets the need for implementation on all consumers
- more behaviors = more bugs
- import filters are optional: a perfect OOXML consumer/producer shouldn't have to deal with old software oddities

PROPOSAL

- get the deprecated features **totally** out of normative scope
- software that produces deprecated tags should **not** be able to claim OOXML compliance
- let the information about old behaviors remain available
- let the OOXML implementors decide whether or not they want their filters to use such information on import time

NOTE

- yes, we know that the deprecated features are already “recommended against” in the spec
- but with such a huge incumbent recommendations may prove of little value
- only enforcement is reliable

APPLICATION DEFINED?

SPEC UNDEFINED

WHAT?

- application defined behavior leaves a degree of freedom for the software producers
- no guaranteed interoperability
- the limiting case is having an empty spec!
- have we learned something from void abuse in other standards? (HTML, ACPI,....)

PROPOSAL

- the correct meaning of all markup must be described
- no generic extensions: specify what extensions may be used for and how
- the possible formats of all contained objects must be defined with reliable pointers to the corresponding specs
- an OOXML validator should be created and maintained by a reliable entity

OOXML SHOULD ONLY BECOME AN ISO STANDARD IF THE FINAL VERSION IS PROVEN TO BE PORTABLE, COMPLETE AND LEGACY FREE *

* EXCEPT WHERE IMPOSSIBLE; EACH LEGACY INCLUSION SHOULD BE CAREFULLY JUSTIFIED

BONUS TRACK

“YOU GUYS WANT EVERYTHING FIXED ON OOXML BUT THE ODF 1.0 SPEC ALSO HAS PROBLEMS X, Y AND Z...”

BONUS TRACK

“YOU GUYS WANT EVERYTHING FIXED ON OOXML BUT THE ODF 1.0 SPEC ALSO HAS PROBLEMS X, Y AND Z...”

IF THAT'S SO, ODF MUST BE IMPROVED.
THIS IS NOT A REASON TO ALLOW OOXML BAD QUALITY

- ODF has a **solid open source cross platform reference implementation**
- unlike OOXML it has **proven** interoperability
- there is **no incumbent** for ODF

RISK OF ABUSE IS LOWER WITH ODF BUT IF PROBLEMS ARE FOUND THEY MUST BE FIXED AS WELL!