

# Specification of OOPS

Chris Johnson 12/12/2007- 11/03/08

## Document history and progress

v0 20/12/07

into subversion 21/12/07

plan and outline v1 13/12/07

specification

input specification v1 20/12/07 v1.1 5/3/08

output spec v2 20/12/07 semi-formal indent, justify, maximal fill

separated from Design doc 11/3/08

info requests and sources

retrieved Open Document Format v1.1 / v1.0 pdf

asked Ian Barnes for design/spec notes 20/12/07 email – got a little info in reply

## Overview

Oops is the Open Office Document Publication System. Oops provides a “pre-flight” check of Open Office documents that have been prepared using the ACM Journal template, written for Microsoft Word. Oops reads an Open Office word processor file (Open Office version 1 format (.sxw source file extension) or version 2 format (.odt extension) converted from a Microsoft Word file that was prepared using the ACM journal template.

Oops scans and parses the OpenOffice document's content, builds a parse tree representing the logical structure of the document, and then outputs:

- some metadata extracted from the document (title, authors' names and affiliations)
- a .txt file containing a formatted plain text version of the content of the document,
- a .html file containing a simple HTML web page version of the document, and
- a .xml file that shows the XML source with indentation to indicate the tree structure of the document.

## Development History

The program was designed and written in the Department of Computer Science, ANU, by Ian Barnes in 2003 (in Eiffel) and rewritten in Java by Alexei Khorev, 2006. It was intended as a teaching example for the course COMP2100 Software Construction, and it has been used in that course each year 2004-2007. Chris Johnson created the written specifications and did significant design and implementation extensions and refactoring in 2007-08.

## Table of Contents

### **Table of Contents**

|   |    |
|---|----|
| Specification of OOPS.....              | 1  |
| Chris Johnson 12/12/2007- 11/03/08..... | 1  |
| Document history and progress.....      | 1  |
| Overview.....                           | 1  |
| Development History.....                | 1  |
| Table of Contents.....                  | 2  |
| Specifications.....                     | 3  |
| Product description.....                | 3  |
| Functional Requirements.....            | 3  |
| Data Requirements.....                  | 3  |
| Non-functional requirements.....        | 11 |
| System Requirements.....                | 11 |
| Performance Requirements.....           | 11 |
| Appendix A.....                         | 11 |
| XML References.....                     | 11 |

# Specifications

## Product description

Oops is a utility program to extend the toolkit of document authors who are writing for publication in a journal or conference proceedings. The existing toolkit starts from a Microsoft Word template supplied by ACM (the *Association for Computing Machinery*, one of the two largest publishing institutions in the computing discipline). The author applies the template to create a template document, working in Microsoft Word or in Open Office, and uses Open Office to read the word document and converts it to an open XML representation for further editing and tools to be applied. Oops assists the author and publisher by extracting and displaying metadata from a document, applying simple transformations to “tidy up” the representation. It assists the author by producing alternative formats for the document for presentation or extraction of text.

## Functional Requirements

Oops shall run as a command that takes a single filename argument with .sxw or .odt extension. The named file shall be in Open Document Format. Oops shall apply a number of defined transformations to the ODF “content” portion of the file, and shall output a pretty-printed plain text version of the XML; a plain text version of the text data in that content; and a partial HTML marked up version of the content.

## Data Requirements

### 1. *Input format*

#### Background

The OpenOffice.org application uses the OpenDocument file format (also known as ODF) version 1.0 OASIS OpenDocument/ISO/IEC 26300 defined *in 738 pages* at [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=office#technical](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office#technical) as its native file format.

Documents are represented as a compressed (zipped) archive containing XML files representing the metadata, default styles, document content, and settings for the application. The content is in a file named `content.xml`. This file contains an XML representation of the document styles and its contents. [see References for a description of XML elements, tags, attributes and content]. An Open Office text-document contains “any sequence of text content elements, which includes paragraphs (and headings), text sections, and indices.” Of these only the paragraphs, headings, and text-sections are used in Oops.

The set of paragraph styles is determined to some extent by the use of the ACM publication template to prepare a Microsoft Word document, converted to ODF, that is the source of the text documents processed by OOPS. The conversion process is outside the scope of this specification, but is assumed to have been performed by Open Office Writer. The ACM template is found at

[http://www.acm.org/publications/latex\\_style/ms\\_word\\_template.doc](http://www.acm.org/publications/latex_style/ms_word_template.doc)

## Metadata, styles, settings

1.0. The ODF document metadata, default styles, and settings shall be ignored.

## Content

The content is assumed to be well formatted XML. Some constructs will be ignored and whitespaces will be compressed. The following specifications use EBNF (Extended Backus Normal Form) notation, described in Appendix A.

1.1. All XML document type declarations shall be ignored (Regular expression syntax in EBNF: `<!DOCTYPE.*>` )

1.2. All XML comments shall be ignored (syntax: `<!--.*-->` )

1.3. All XML processing instructions shall be ignored (syntax: `<?.*?>` )

1.4. All XML tags and attributes shall be accepted.

1.5 An end-tag with the same name shall match each start-tag, properly embedded.

1.6 Any sequence of whitespace embedded in an attribute value shall be compressed to a single space. (whitespace sequence syntax: `[:space:]*` )

1.7. All data content shall be accepted.

1.8 Any sequence of whitespace embedded in data content shall be compressed to a single space.

1.9 Any data element containing only a sequence of underscores, or a sequence of hyphens, shall be ignored. *Note:* this specification is changed in Assignment 1, 2008.

## Data model: paragraph styles

ODF represents a text document using elements with the tags `text:h` (for headings), `text:p` (for plain paragraphs), the self-describing `text:unordered-list`, `text:orderedlist`, `text:listitem`, `text:a` (anchor), and `text:span` (to associate a style with a sequence of text embedded in paragraph).

In ODF a text paragraph is represented as an XML element with the tag “`text:p`”. A text paragraph has an associated *style* if the “`text:p`” element has an attribute “`text:style-name`”. The name of the style is the value of this attribute.

The *properties* of a style named X is the set of attribute-value pairs that is contained in the attribute “`style-properties`” of a “`style:style`” element that also has the attribute “`style:name`” with value equal to X.

The `text:span` element may also have an associated style name.

## Output formats

Oops shall produce output in four formats:

1. metadata in plain ASCII
2. plain text content
3. HTML content

#### 4. XML content

Metadata is produced to standard output (console). Each of the other forms of output is produced in a separate file in the same directory as the input file. The filename of the output file is formed from the input filename by replacing the extension suffix of the input filename by `txt`, `html`, `xml` respectively.

## 2. Metadata output

For the purposes of Oops, **metadata** is distinguished as data in the document content which is in any text paragraph with the named styles "Author's name", "Title", "Affiliation", "Abstract", or "Categories".

2.1 Oops shall output to the standard output console the Title metadata, then the Author's Name metadata, then the Affiliation metadata. These outputs shall be labelled and be on separate lines.

## 3. Plain text output

3.1 The plain text output shall contain all of the data elements of the input, in the same order.

3.2 The layout (line breaks and indentation) of the output is determined by selected XML tags in the input according to Table 1.

3.3 The output shall be in blocks of ASCII text. Each block of text shall contain all of the non-whitespace characters in a character string in a data element in the input. It shall contain a single space in place of each sequence of whitespace characters on the input.

3.4 The layout of each block of text shall be left justified, bounded (unless it overflows), maximally filled, indented from the left. This is described in detail in 3.6 below.

3.5 Elements shall be from the set of ODF element tags defined in column 2 of table 3.5.1. Other elements shall be reported and subsequently ignored.

3.6 Elements shall be transformed and output as described in Table 3.5.1.

|        | <b>ODF element tag</b> | <b>element class</b> | <b>plain text output transform</b>   |
|--------|------------------------|----------------------|--|
| 3.5.1  | office: body           | Body                 | children   |
| 3.5.2  | text: h                | Heading              | level 1: uppercase(children)<br>level 2: children  |
| 3.5.3  | text: p                | Paragraph            | children <i>followed by</i> emptyline  |
| 3.5.4  | text: unordered-list   | UnorderedList        | content  |
| 3.5.5  | text: ordered-list     | OrderedList          | content  |
| 3.5.6  | text: list-item        | ListItem             | ordered list item: consecutive arabic numbers of 1, 2 or 3 digits, starting at 1 for each list[1]<br><i>followed by</i> increaseIndent(children)<br><br>unordered list item: literal characters " o " (space, lowercase 'o', space)<br><i>followed by</i> increaseIndent(children) |
| 3.5.7  | text: span             | Span                 | content  |
| 3.5.8  | text: a                | Anchor               | content  |
| 3.5.9  | text: s                | Space                | single space   |
| 3.5.10 | text: line-break       | LineBreak            | content  |
| 3.5.11 | <i>plain text</i>      |                      | <i>indent, left justify</i>  |

**Table 3.5.1 Acceptable input elements and Transformations into Plain Text output**

Note [1] the presence of the *consecutive numbering* attribute on the input list element shall cause the output numbering to run on from the preceding list, if any, otherwise the list numbering will start at 1.

Note [2] all style elements shall be ignored

Note [3] no tags or attributes shall appear in the plain text output.

The notation "children" signifies all of the contained/children elements of this element, in the original input order. "uppercase" denotes the transformation of all letter characters in its argument to uppercase letters.

Children elements are run on into the output by default, i.e. no additional interspacing is output unless explicitly specified in this scheme.

### 3.6 Justification and indentation scheme

The output shall consist of consecutive lines of text. Each line of output shall be empty, a sequence of words, or a single word as follows. It shall be

- **an empty line**
- *or* a sequence of words with the following constraints
  - (*left justified and indented*) the first word of the line shall be preceded by a number of spaces corresponding to the current **indentation level** (see 3.7 below)
  - the words in the sequence shall be separated by single space characters
  - (*bounded*) the total length of the line including the indentation, the characters in the words, and all separators shall be less than the **maximum line length**
  - (*maximally filled*) if the current line is not the last line of this block, and there is any word on the following line, then the length of the current line plus the length of that first word plus one separating space would exceed the maximum line length
- *or* a single word (**long word overflow**)
  - the word shall be preceded by a number of spaces corresponding to the indentation level
  - the total length of indentation plus this word's length is equal to or greater than the maximum line length.

The **maximum line length** shall be 64 characters.

### 3.7 Indentation scheme

3.7.1 A block of plain text shall be output with the same indentation level for every line.

3.7.2 The indentation of the first line of a block shall be the same as that of the preceding plain text block, if any.

3.7.3 The indentation level of the first block shall be 0.

3.7.4 Each item in an *ordered list* shall start on a new line. It is preceded by a list item number right justified into 3 character positions and a space. The item numbers increment continuously within a list. The item numbering is reset to 1 at the start of a new list, except that a “continuous numbering” list continues incrementing from the preceding list. The indentation level is then incremented by 1.

3.7.5 each item in an *unordered list* is preceded by 2 spaces, a bullet (represented as the 'o' character) and a space. The indentation level is then incremented by 1.

3.7.6 a *block element* (in the HTML sense) is followed by an empty line

3.7.7 an *inline element* (in the HTML sense) has no additional separation characters.

3.7.8 a *level 1 heading* is output transformed into uppercase characters.

3.7.9 Other levels of heading are output in unchanged case.

3.7.10 the copyright symbol © is output as the copyright symbol in extended ASCII. No other special characters are recognised.

3.7.11 The quantum of indentation shall be 4 character positions for each level of increment.

## **4. HTML output**

The HTML output shall contain transformations of selected XML elements of the input, and all of the data elements of the input, in the same order. These transformations have been defined pragmatically from inspecting particular documents to get an effective output for the author to view readily.

### **4.1 Paragraph styles**

Each `text:p` element shall be transformed to HTML elements depending on its style-name as in Table 4.1.

|        | <b>Paragraph style name</b>                  | <b>HTML element</b>             |
|--------|--|---------------------------------|
| 4.1.1  | Title  | centred, H1                     |
| 4.1.2  | Abstract                                     | Blockquote, in smaller font     |
| 4.1.3  | Author's Name<br>Author_27_s_20_Name         | paragraph, centred, bold        |
| 4.1.4  | Affiliation                                  | paragraph, centred              |
| 4.1.5  | Categories                                   | paragraph, smaller font         |
| 4.1.6  | Text body<br>Text_20_body                    | paragraph                       |
| 4.1.7  | Quoted Text<br>Quoted_20_Text                | blockquote                      |
| 4.1.8  | Numbered List<br>Numbered_20_List            | numbered list OL                |
| 4.1.9  | Footnote                                     | paragraph, smaller font         |
| 4.1.10 | Figure Caption<br>Figure_20_Caption          | centred paragraph, smaller font |
| 4.1.11 | Table Head<br>Table_20_Head                  | centred paragraph               |
| 4.1.12 | References                                   | paragraph                       |
| 4.1.13 | Primary Head<br>Primary_20_Head              | H2                              |
| 4.1.14 | Secondary Head<br>Secondary_20_Head          | H3                              |
| 4.1.15 | Displayed Equation<br>Displayed_20_Equation  | blockquote, italics             |
| 4.1.16 | Initial Body Text<br>Initial_20_Body_20_Text | paragraph                       |

**Table 4.1 Transformation map for paragraph styles**

Other ODF elements shall be transformed as in Table 4.2.

| <b>ODF element tag</b> | <b>element class</b> | <b>relevant attributes</b> | <b>HTML element</b>  |
|------------------------|----------------------|----------------------------|--|
| text:h                 | Heading              | text:level                 | H (value of text:level)  |
| text:p                 | Paragraph            |                            | see Table 4.1  |
| text:ordered-list      | OrderedList          | text:continue-numbering    | UL<br>if continue:numbering then set start= explicitly to continue numbering |
| text:unordered-list    | UnorderedList        |                            | OL   |
| text:list-item         | ListItem             |                            | LI   |
| text:span              | Span                 | style-name                 | SPAN<br>set inline style for Italic, Bold and SmallCaps attributes           |
| text:a                 | Anchor               | xlink:href                 | A HREF=  |
| text:s                 | Space                |                            | single space   |
| text:line-break        | LineBreak            |                            | line break   |

**Table 4.5 Relevant element types for HTML Output**

### ***XML output***

Output each ODF element, in order, according to the following scheme.

| <b>ODF element tag</b> | <b>element class</b> | <b>relevant attributes</b> | <b>HTML element</b>  |
|------------------------|----------------------|----------------------------|--|
| text:h                 | Heading              | text:level                 | H (value of text:level)  |
| text:p                 | Paragraph            |                            | see Table 4.1  |
| text:ordered-list      | OrderedList          | text:continue-numbering    | UL<br>if continue:numbering then set start= explicitly to continue numbering |
| text:undered-list      | UnorderedList        |                            | OL   |
| text:list-item         | ListItem             |                            | LI   |
| text:span              | Span                 | style-name                 | SPAN<br>set inline style for Italic, Bold and SmallCaps attributes           |
| text:a                 | Anchor               | xlink:href                 | A HREF=  |

## Non-functional requirements

### System Requirements

Expects Word and Open Office to be available, but not strictly required (input document can be imported from another system).

### Performance Requirements

not specified.

## Appendix A

### EBNF Notes

EBNF (Extended Backus Naur Form, or Backus Normal Form) is a notation for the restricted form of syntax called *regular expressions*. The notation varies widely. We apply a subset of the notation used in the Unix tool `egrep`.

| Oops Input Syntax (EBNF notation) |   |
|-----------------------------------|---|
| <code>input</code>                | <code>= ( element   dataContent ) *</code>      |
| <code>element</code>              | <code>= startTag input endTag   emptyTag</code> |
| <code>startTag</code>             | <code>= '&lt;' name attribute* '&gt;'</code>    |
| <code>emptyTag</code>             | <code>= '&lt;/' name attribute* '&gt;'</code>   |
| <code>endTag</code>               | <code>= '&lt;' name '/&gt;'</code>              |
| <code>name</code>                 | <code>= [^ &gt;/=]*</code>                      |
| <code>attribute</code>            | <code>= name '=' value</code>                   |
| <code>value</code>                | <code>= ''' [^"]* '''</code>                    |
| <code>dataContent</code>          | <code>= [^&lt;]+</code>                         |

Using the notation in the form for the Unix utility `egrep`:

[1] Characters in single quotes represent themselves. For example, `<` denotes literal character `<` (less-than). `</` denotes the string of two characters `<` followed by `/`.

[2] `|` denotes alternatives, with low precedence.

[3] `*` denotes repetition (0 or more times) of the preceding element;

`+` denotes repetition 1 or more times.

[4] `[...]` denotes any member of a set of characters, as in regular expressions [e.g. see manual for Unix `egrep` command]

[5] `[^ ...]` denotes any member of the negation of the set of characters [as for `egrep`]

[6] Parentheses (round brackets: `()`) enclose groups of items to be taken together.

## XML References

- [A Technical Introduction to XML](#) by Norman Walsh, who is one of the major figures in the

development of XML.

- [The Official XML 1.0 Specification](#) from the W3C. (A difficult document to read, but all the information is there - more than you will ever need.)
- [XML Basics](#), an article from Linux Mag that gives a good gentle introduction.