

FOP Alternative Design

Alternative Design Approach to FOP

\$Revision: 1.4 \$ \$Name: \$

by Peter B. West

Table of contents

1 Alternative Design.....	2
1.1 Status and availability.....	2

1. Alternative Design

This section of the FOP web site contains notes on approaches to an alternative design for FOP. The individual documents here are fragmentary, being notes of particular issues, without an overall framework as yet.

The main aims of this redesign effort are:

- full conformance with the Recommendation
- no limitation on the size of files
- increased performance
- reduced memory footprint

Secondary aims include:

- increased performance
- reduced memory footprint

In order to achieve these aims, the primary areas of design interest are:

- Representing properties, for most purposes, as integers.
- Implementing a top-down processing model for each of the processing components.
- Distributing FOP processing over a number of threads with single-point downstream communication and flow control by means of traditional producer/consumer queues. The threads so far under consideration are:
 - XML parser
 - FO tree builder
 - layout engine
 - Area tree builder
- Redesigning XML parsing and FO tree building using a **pull-parsing** methodology with integrated FO input validation.
- Representing vital relationships among the elements with appropriate data structures. These include:
 - Representing trees with explicit Tree objects, rather than as implicit relationships among other objects.
 - Drawing threads through the tree nodes to represent linear layout relationships for resolving keeps, breaks and space specifiers.
- Caching integrated into the tree node access methods.

1.1. Status and availability

The *ALT DESIGN* effort is not taking place on the main line of development, represented by the *HEAD* tag on the CVS trunk. The source is available via the FOP_0-20-0_Alt-Design tag. This code has only a non-*Ant* build environment based on some small unix shell scripts and the *jikes*

FOP Alternative Design

compiler. The parser stage and the FO tree building code is present. The first example of producer/consumer binding is working, the Tree class and the Node class with inner *iterator* classes are available and working. Property handling is almost complete, and all FO classes are present and sufficiently complete to allow for FO tree building.

[Preliminary results](#)

(<http://marc.theaimsgroup.com/%3Ff=fop-dev%26m=103890259919360%26w=2>) and

[follow-up testing](#)

(<http://marc.theaimsgroup.com/%3Ff=fop-dev%26m=103918886413611%26w=2>) of FO tree building shows memory reductions of almost 50% compared to the most recently tuned version of the maintenance version of the code (FOP 0.20.5 RC). Alt-Design FO tree building was also slightly faster, in spite of the use of pull parsing implemented on top of SAX.

Currently, only [Peter West](mailto:pbwest@powerup.com.au) (mailto:pbwest@powerup.com.au) is working on the ALT DESIGN sub-project.