

The sooner you start coding, the longer
it takes
—Anonymous

CHAPTER 6

FOSI Methodology

The methodology and best practices outlined in this chapter are designed to streamline FOSI stylesheet development, help ensure on-time and on-budget delivery, and create top quality, easy-to-maintain, state-of-the-art FOSI stylesheets capable of supporting “lights out” print/PDF publishing with Arbortext Editor.

A FOSI that has not been designed with maintenance in mind is costly and time-consuming to maintain. DTDs as well as formatting specifications can change over time, so FOSI maintenance is an issue for most organizations. The methodology described here is designed to make FOSI development and maintenance as easy as possible.

The components of the author’s methodology are detailed in the following sections.

- **Project information** on page 685
- **FOSI design** on page 690
- **FOSI development** on page 691
- **FOSI quality assurance** on page 719
- **FOSI documentation** on page 720

JARGON

“Lights out” refers to batch formatting that does not require operator attendance or intervention.

TIP

The best part about following the methodology described in this book is that you write less rather than more code. And, if you adhere to the recommended naming conventions and best practices, few comments are needed.

TIP 

Identifying expectations ahead of time helps ensure client satisfaction with the completed project.

TIP 

"Requirements" are needed to estimate the effort. "Specifications" are needed to do the work. It's usually best to wait until the project is approved before digging into the "specs."

Project information

It would be nice if you could count the number of elements and attributes in the DTD, multiply them by some magic number, and get a good estimate of how long it will take to develop the FOSI. However, it's not that simple. The time needed for FOSI development depends on the complexity of the desired formatting and the requirements of the SGML/XML project.

To scope the FOSI development effort, you need project information, which includes printed samples, project requirements, and formatting specifications. This information is vital for developing a FOSI on schedule and on budget.

Printed samples

Printed samples are needed to estimate a FOSI project. Get as many as you can. However, printed samples do not provide sufficient information for coding a stylesheet. Formatting requirements and specifications are needed to develop a FOSI stylesheet, as described in the sections below.

NOTE: It is important to keep in mind that a FOSI is developed for a DTD, not a document. The FOSI stylesheet is used to format all documents that follow the DTD, which includes documents that have not yet been written. It is possible to format nonexistent documents because the elements and structure defined in the DTD limit the possibilities.

Project requirements

To estimate the effort and answer the question of how long it will take, you need to know the project requirements. The questions below are designed to determine the scope of a FOSI project. They collect the information needed for an accurate estimate of FOSI development efforts and uncover issues — hidden "time bombs" — that require further investigation. Note that some are not applicable to screen FOSIs.

NOTE: In addition to scoping the project for the estimate, requirements information is part of the acceptance criteria for the completed project.

1. How much information — volumes, books, reports, pages — is to be printed? Please provide samples of the documents you want to print.
2. What is your deadline for print/PDF publishing?
3. What version of Arbortext Editor are you using?

4. How many DTDs are involved in your project
5. Do you need a FOSI to format screen display in Arbortext Editor?
6. What languages are used in your documents?
7. How many languages are used in one document?
 - Multiple languages in one document
 - One language per document
8. What paper sizes will be used? Please check all that apply:
 - 8-½×11
 - 8-½×14
 - 11×17
 - A4
 - A5
 - Other; please specify:
9. Are different versions of the same document needed? Please check all that apply.
 - Draft/Final
 - Student/Instructor
 - Public/Confidential
 - Standard-size/Pocket-size
 - Other; please specify:
10. Will documents always be printed in their entirety? Or will partial documents be printed? For example, does a single section of a manual need to be printed separately using the same section numbering as in the complete manual?
 - Always print entire document
 - Sometimes print part of a document
11. What DTD attributes must be supported by the FOSI? For example, a *foldout* attribute on a <figure> tag may identify foldout figures on oversized pages, which must be published in a separate volume.
12. What DTD elements are inline elements that occur within a paragraph or title and require no special formatting? Such elements are quickly and easily supported in the FOSI.
13. What DTD elements are block elements and what formatting do they require? Paragraphs may be straightforward, but other block elements may have complex formatting requirements.
14. What list markers (numbering, lettering, bullets, etc.) are required?

TIP 

For a "screen FOSI" to format the Edit window display, be sure to check with authors and other end users of Arbortext Editor for their needs. Keep in mind that FOSI can display messages and guidelines that do not appear in print/PDF output.

15. How many levels of nested lists must the FOSI support? The DTD may allow unlimited nesting, so it is important to find out what is used in practice.
16. What text structures, such as a table of contents or index, must be generated?
17. What page layouts are needed? Some page layouts are simple, others are quite complex and therefore time-consuming to support.
18. Other than lists, what elements must be numbered or lettered? Most documents number chapters, others require complete hierarchical numbering. Figures and tables may also be numbered.
19. What elements reset numbering or lettering? For example, is figure numbering reset for each chapter?
20. What elements require generated text other than numbering or lettering? For example, “Figure” can be output before each figure number. Keep in mind that the effort required for this item may be increased when more than one language is supported by the FOSI.
21. What elements require generated graphics? For example, a logo graphic can be automatically output on the title page.
22. What elements should be suppressed from output?
23. What elements should be output in a different order?
24. For any element with an ID attribute, what should be output when the element is cross referenced?
25. Do your documents require any of the following? Please check all that apply.
 - Footnotes
 - List of Effective Pages
 - Security classification
 - Change pages (point pages)
 - Change bars (revision marks)
 - Other; please specify
26. Are you planning to use File->Save As HTML to create HTML for browser display?
27. Will documents be printed to PDF files?

Formatting specifications

Formatting specifications provide the details about the documents to be formatted. This level of detail is generally not needed to estimate the effort but is needed to develop the FOSI.

PDF and printed samples will provide some of the formatting specifications you need. Other sources are corporate identity manuals, style guides, spec sheets, and existing stylesheets.

To code the FOSI, you need to collect the formatting specifications listed below. Keep in mind that some formatting is not supported for Edit window display. Refer to individual categories for details.

- Exact formatting specs for all elements and attributes to be formatted
- Exact generated text (in all languages to be supported) and formatting for it
- Generated numbering styles, formatting, incrementing, and reset information
- Gentext and formatting for cross references
- Floated objects and float locations
- Page dimensions
- Page margins
- Column and gutter widths
- Source and formatting for all page headers and footers
- Page numbering styles
- Security markings text, priority, and formatting
- Revision bars weight and location in the page margin

NOTE: FOSI may not directly support all desired formatting. Also, the DTD may not have the structure needed by the FOSI. The following figure shows how to handle such situations.

TIP

If you measure printed samples to determine specs, you'll probably find they vary from sample to sample. Before coding the FOSI, ask for confirmation of your measurements.

TIP

You may find that the measurements of a bound printed sample are not nice round numbers because you are measuring the "trim size." For example, page dimensions may measure $5\frac{7}{8} \times 8\frac{3}{8}$ rather than 6×9 or $8\frac{3}{8} \times 10\frac{7}{8}$ rather than $8\frac{1}{2} \times 11$. Check with your client to find out what page dimensions you should code in the FOSI.

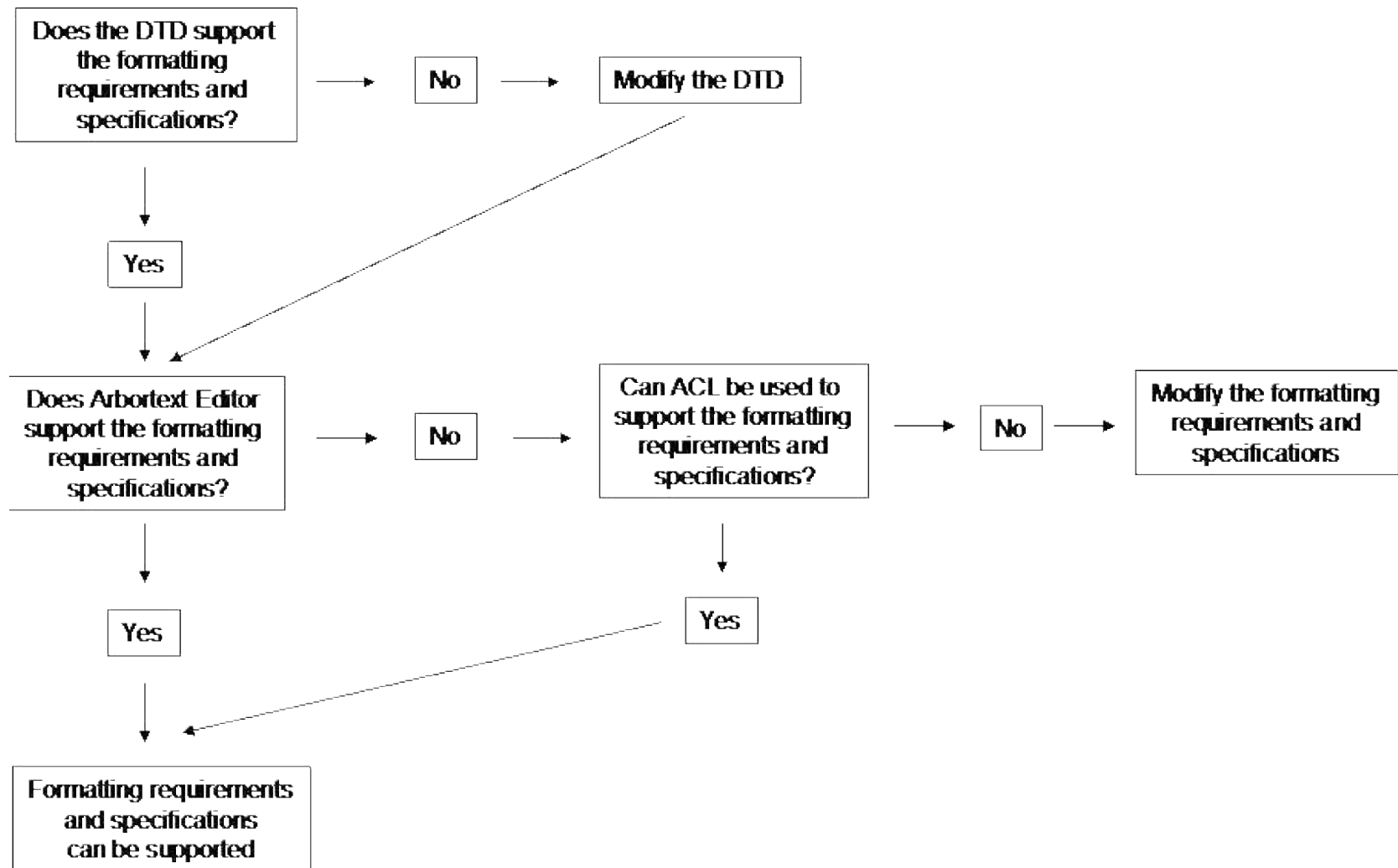
FOSI TIP

Most organizations have a style guide and/or a corporate identity manual, which are excellent sources of information for formatting requirements and specifications.

TIP

When the markup needed to support the desired formatting cannot be added to a "set-in-stone" DTD, consider creating a "Publishing DTD" that is used only for print/PDF production. Authors can continue to use the "Authoring DTD," while the Publishing DTD can be modified as needed without affecting other processes.

Figure 292 Formatting analysis



FOSI design

Before you start coding the FOSI, you need to consider the following issues related to the SGML/XML project.

- If formatting is need for print/PDF as well as for the Edit window, should you develop separate FOSIs for screen and print, or should one FOSI support both using the “editor-only” feature?
 - If authoring and printing are to be done at different geographic locations, separate FOSIs may make more sense.
 - If the formatting for the screen and print/PDF is very different, two FOSIs should be considered.
 - If the formatting for the screen and print/PDF have much in common, one FOSI may make more sense.
 -
- If several languages must be supported, should each language have its own FOSI, or should all languages be handled by one FOSI?
 - If multiple languages will be used in each document, one FOSI is needed.
 - If each language will be used at a different geographic location, consider developing a FOSI for each language.
- If more than one DTD must be supported, must two FOSIs be developed and maintained?
 - Each DTD and doctype must have its own FOSI. But if DTDs share elements and formatting, the FOSIs can be designed to share common formatting, as described in **How to create modular FOSIs** on page 652.

FOSI development

Once you have collected the requirements and specifications information and decided FOSI design issues, as outlined above, you are ready to begin coding. First a word about tools you will need.

NOTE: FOSI development is essentially a one person task. There is no easy way for multiple developers to code independently and then merge their code into a single FOSI. Others who want to help can best contribute by developing test documents and testing the FOSI and application. Refer to **Developing test documents** below for details.

Tools

It is important to keep in mind that an organization's public documents are part of its brand. Your print FOSI is probably expected to duplicate current formatting exactly.

Figure 293 Clear plastic ruler and type gauge

You will need one or more clear plastic rulers to measure points, picas, inches, and possibly millimeter. Choose rulers with leading and rule weight gauges as well as E-scales for gauging type sizes.

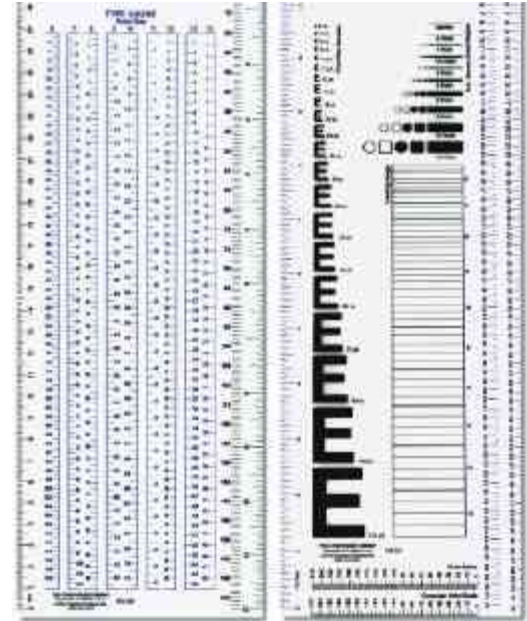


Figure 294 Loupe

You may also want to acquire a loupe or magnifying glass. A type-size finder may also come in handy.

**Figure 295 Rubber finger tip**

Another useful tool is a rubber finger tip, which makes it easy to turn each page in a document without wetting your finger or getting a paper cut. Rubber fingers are inexpensive and can be found at any office supply store.

**TYPE TIP** 

An easy way to check if the output exactly matches the current formatting is to put a page of output over a similar page of the original, align the text at the top left, and hold the pages up to a strong light. It will be immediately obvious if the margins and leading don't match.

If you need to do a lot of this sort of verification, consider buying or making a light box.

TIP 

You can search the Internet for "graphic arts supplies" or specific tools to find sources for them.

TIP 

For a screen FOSI, test documents are needed when or soon after FOSI development begins.

TIP 

You may find it helpful to include text that identifies the formatting to be tested. For example: `<title>THIS LONG TITLE TESTS WHAT HAPPENS IF IT DOESN'T FIT ON ONE LINE</title>`

Test documents

The following sections described how to code the `docdesc` and basic components of the FOSI. This work can be done with a single sample file. However, after this initial coding is done, test documents are needed to continue work on the FOSI. The test suite will also be used to test future FOSI modifications due to changes to the DTD, formatting specifications, boilerplate text output by the FOSI, etc.

NOTE: Along with written project requirements and formatting specifications, test documents generally constitute acceptance criteria for the project.

It is important to realize that just as printed samples are not sufficient for estimating FOSI effort, samples of existing SGML/XML files are not sufficient for testing a FOSI. Several robust test documents are needed to thoroughly exercise the DTD and formatting specifications. They should be “enriched” files that include more markup than would typically occur in an actual file. For example, a table may rarely be used in your documents, but if tables are included in the DTD and are to be supported in the FOSI, then a table should be included in at least one document. Similarly, lists that are nested to three levels may rarely occur, but if the DTD allows nested lists and if the FOSI is to support three levels, then three levels of nested lists should be included in one or more test documents.

NOTE: For best results, test documents should not be developed by the person developing the FOSI. Ideally, the client should develop test documents. Either way, when you start working on a FOSI, someone else should be working on developing test documents.

NOTE: When the DTD and formatting change in the future, the test documents must be updated so they continue to thoroughly test all requirements.

Developing test documents

The following sections provide guidelines on developing test documents that will thoroughly test your FOSI. First are examples of testing general formatting such as quadding, indenting, and page layouts, followed by examples of testing SGML/XML-related formatting. For each example, unless otherwise specified, the elements to be tested may be inserted in the same test document.

General Formatting

In your test documents, include at least one of each element that is to be supported in the FOSI. If the formatting for the element is different in different contexts, be sure to cover all possibilities in the test documents. The tables below provide examples of how to test general formatting in your test documents.

Table 100 Justification

Formatting Requirements	Test Document(s)
A <legend> should be justified with the last line centered	Include enough text to fill at least three lines of output to show both justification and centering
<title> should be flush right with any turnover lines also flush right	Include enough text to require at least two lines of output to show flush right for multiple lines

Table 101 Indent

Formatting Requirements	Test Document(s)
<list-item> should have a 1-em hanging indent	Make the <list-item> long enough to output at least three lines to show the hanging line and the indented lines

Table 102 Wraparound graphic

Formatting Requirements	Test Document(s)
<para> should wraparound a <graphic>	Make the <para> long enough to wrap around the entire graphic. Test different graphics and different sizes of graphics.

Table 103 Page layouts

Formatting Requirements	Test Document(s)
An index includes an opening recto page (right-hand page), a verso page (left-hand page), and a facing recto page	Include enough index entry elements to generate at least three pages of index material

SGML/XML-based formatting

FOSI formatting is designed to work directly with SGML/XML, including optional elements, repeatable elements, attributes, context, and nesting. In

addition, a FOSI can apply formatting based on an element's occurrence within its parent's structure. Examples are shown in the tables below.

Table 104 Optional elements

Formatting Requirements	Test Document(s)
When an optional index is included in the document, it should start a new recto page, but when there is no <index>, a new page should not be started	Use two test documents. In one, insert an <index> element. Do not include it in the second test document.
Each optional, repeatable <name> in <attendees> should be separated from the previous name by a comma and a space	Use three test documents. In one, insert just one <name>. In the second test document, insert multiple <name> elements. In the third, do not include <name>

Table 105 Repeatable element

Formatting Requirements	Test Document(s)
Repeatable element <copyright-year> is to output a comma and space between each year if there is more than one element within its parent <title-page>	Assuming one or more <copyright-year> elements are allowed in only one location in the document structure, use two test documents. In one test document, include a single <copyright-year> element within <title-page>. In the other, include at least two <copyright-year> elements

Table 106 Context

Formatting Requirements	Test Document(s)
<para> should be italic when in context of <intro> but roman when in context of <section>	Include <para> in <intro> and also in <section>
<emphasis> is roman when in context of <para> in context of <intro>, but italics when in context of <para> in context of <section>	Include <emphasis> around text in <para> in <intro> and <emphasis> around text in <para> in <section>
<section> in context of <book> should generate a section title page, but when one <section> is output on its own and not part of <book>, then no section title page should be generated	Use two test documents. Include <book> and multiple <section> elements in one test document. In the other, include only one <section>

Table 107 Nesting

Formatting Requirements	Test Document(s)
The desired format for an <code><item></code> in a <code><numbered-list></code> differs at each nested level, with a maximum of three nested levels: level one is Arabic numbers, level 2 is lowercase alphabetic characters, and level 3 is lowercase roman numerals	Include three nested levels of the <code><numbered-list></code> element

Table 108 Occurrence

Formatting Requirements	Test Document(s)
The first occurrence of element <code><para></code> within its parent <code><section></code> should have no indent on the first line, but subsequent <code><para></code> elements should have a 1 em indent on the first line	Include more than one <code><para></code> in <code><section></code>
The first <code><section></code> within its parent <code><book></code> should start a new recto page, but each following <code><section></code> should continue on the current page and not start a new page	Include more than one <code><section></code> within <code><book></code>

Table 109 Attributes

Formatting Requirements	Test Document(s)
When its attribute <i>box</i> is set to <i>yes</i> , the entire <code><figure></code> should be boxed, but it should not be boxed when the attribute is set to <i>no</i>	Include two <code><figure></code> elements. Set <i>box</i> to <i>yes</i> on one, and <i>box</i> to <i>no</i> on the other
When its <i>style</i> attribute is set to <i>bullet</i> , a bullet should be output before each <code><random-item></code> , but when <i>style</i> is set to <i>dash</i> , a dash should be output	Include two <code><random-list></code> elements. Set the <i>style</i> attribute on one to <i>bullet</i> and on the other to <i>dash</i>
The desired format for the document depends on the <code><book></code> <i>style</i> attribute, which has six possible settings	Assuming <code><book></code> is allowed only once in the document structure, six separate test documents are needed, each with the <i>style</i> attribute set differently
When the <code><workbook></code> <i>version</i> attribute is set to <i>instructor</i> , every <code><answer></code> should be output, but when <i>version</i> is set to <i>student</i> , <code><answer></code> should be suppressed	Assuming <code><workbook></code> is allowed only once in the document structure, use two test documents. Set one document to <i>instructor</i> , and set the other document to <i>student</i>