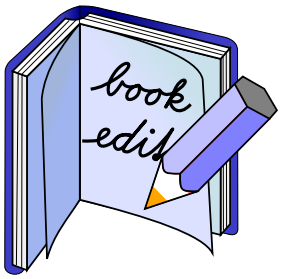
The RichDoc Framework



Michal Ševčenko <sevcenko@vc.cvut.cz> , 10. January 2007

Table of contents

Table of contents

Part I User's Guide **4**
Chapter 1 Introduction 5

Part I
User's Guide

Chapter 1
Introduction

The RichDoc Framework is a comprehensive system that can be used for authoring, managing, exchanging and presenting documents, with emphasis on scientific documents. It is an experimental system developed within the KSMSA project. Its main goal is to validate the experimental results of Michal Ševčenko's dissertation project "Knowledge Support for Modeling and Simulation", but it is published under the GNU license in the hope it will be useful for other users, too. Please refer to Chapter 14 for list of current limitations of the system. Users are also encouraged to contribute to the system, see Chapter 18.

1.1 Basic Philosophy of the RichDoc Framework

What is the difference between RichDoc framework and similar document management systems? There are two main groups of document management systems. The first type of systems, also called word processors, are based on the WYSIWYG (What You See Is What You Get) idea. That is, what you see on a computer screen during authoring a document is very similar, if not equal, to what you get when you print the document. This implies that word processors are *presentation-oriented* – they encourage the user to focus on visual rather than structural aspects of the document being edited. On the other hand, word processors have intuitive user interface, and are thus easy to use even by unexperienced users.

At the other side, there are markup systems, based on some markup language, such as LaTeX or HTML. You do not prepare the document visually, but using a text editor, where you enter the document text mixed with processing instructions (markup) needed to define structural and visual aspects of the document. Documents prepared with such systems usually are usually much more *structure-oriented* – the document data primarily represents the structure of the document, and the visual aspects (font size, style, margins etc) is inferred from this structure automatically using mechanism that is external to the document. This separation enables using the same document in very different contexts. For example, you may generate from the same document output in both HTML and PDF formats. However, using such systems is less comfortable and requires the knowledge of usually complex markup language.

We have tried to combine these approaches, and created a system that is semi-WYSIWYG. It is WYSIWYG in a sense that you are authoring the document visually, using graphical user interface. On the other hand, what you see on the screen primarily represents the structure of the document, not its final visual appearance. The fine visual aspects of the document may be modified later without modifying the actual document content. The content structure is thus much more similar to the traditional markup systems. In fact, there is a markup language behind the user interface of the system, but is designed to thoroughly support the visual nature of editing. Another design feature of our system, that makes it similar to markup language, is the "document consistency principle". The principle rules that the document representation contains only those properties that the user actually explicitly specified. Properties that have not been specified explicitly get some reasonable default values, changeable without modification of the document. This clean structure is then propagated to output formats that can be generated by the framework, such as HTML.

One particularly useful feature of the RichDoc framework is that it forms a compact, portable software component that can be easily embedded into other software systems. The framework can not only create stand-alone documents, but also small documentation chunks that may be embedded to some superordinate function units, such as calendar entries, engineering files, or any other units for

which it is useful to attach documentation to. All this with advanced features like tables, graphics or mathematics.

1.2 What RichDoc Framework is *Not*

The RichDoc Framework may be successfully used in situations when you need to create structurally rich documents with simple, consistent mapping between the document structure and its final visual appearance. If you need, for instance, to frequently change visual properties of document elements in an ad-hoc manner, you may find the RichDoc framework too cumbersome for accomplishing such task. The recommended best practice is to define for certain document elements, such as paragraphs or tables, a small number of classes (see Chapter 5), and merely assign that classes to existing elements. If this scenario does not fit to you, you may consider using more visually oriented text processors, such as Microsoft Word.

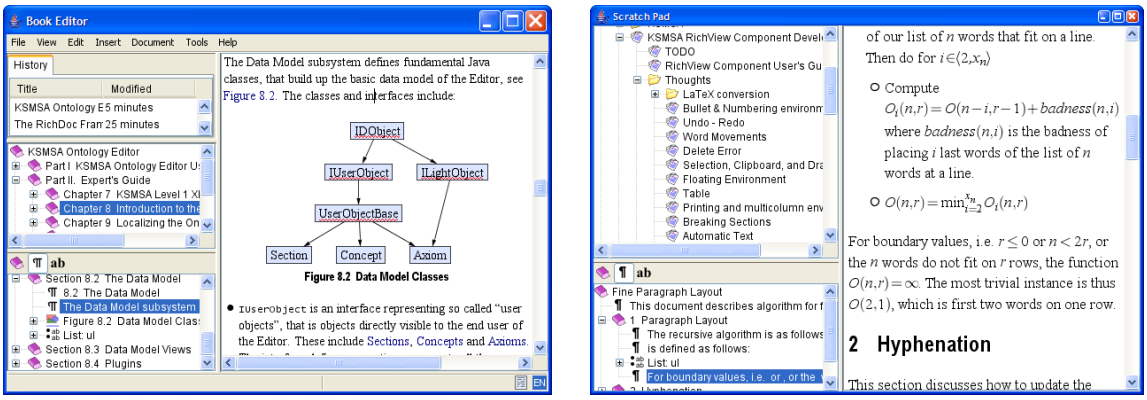
1.3 Requirements and Installation

The RichDoc Framework is pure Java application. It means that it runs on any platform that supports Java technology, including Microsoft Windows, many versions of the Unix operating system, and others. It has been tested on Microsoft Windows XP and RedHat Linux 9.1, but it is expected to run on other Java-enabled platforms as well.

Prior installation of the RichDoc Framework, you need to install the Java Runtime Environment (JRE) from Sun. Just visit http://www.java.com and download the JRE. If you have a JRE installed already, make sure that it is version 1.5 or later. If not, you should install the latest version.

Second step is to download and install a binary distribution of the RichDoc Framework. Two binary distributions are provided: Windows Installer to be installed on Windows, and compressed archive to be installed under UNIX. Installation under Windows is quite straightforward: just run the installer, and follow the instructions. After installation, the start menu is populated with appropriate shortcuts to run RichDoc applications. To install under UNIX, you need to unpack the distribution archive to an appropriate directory, and add the `bin` subdirectory to your `PATH`. Also make sure you have set the `JAVA_HOME` environment variable to the directory of your JRE installation. Then you can run any of the RichDoc programs by typing either `bookEditor`, `scratchPad`, or `ioTool`.

1.4 RichDoc Applications



(*a*) (*b*)

Figure 1.1 Two RichDoc applications: (*a*) BookEditor, (*b*) ScratchPad

There are two basic applications based on the RichDoc framework. **BookEditor**, a program intended for authoring compact, standalone, scientific documents like books and articles, and **ScratchPad**,

a handy tool for making personal notes. The two applications differ in a way they organize documents and their content, but the general content development procedures are the same. The rest of this document deals mostly with the BookEditor program. The ScratchPad program is then described in Chapter 13.

1.5 Getting Started

To start with the BookEditor, you must first create new document. To do this, use the File–New command from the pull-down menu. A New Document Wizard pops up, see Figure 1.2.

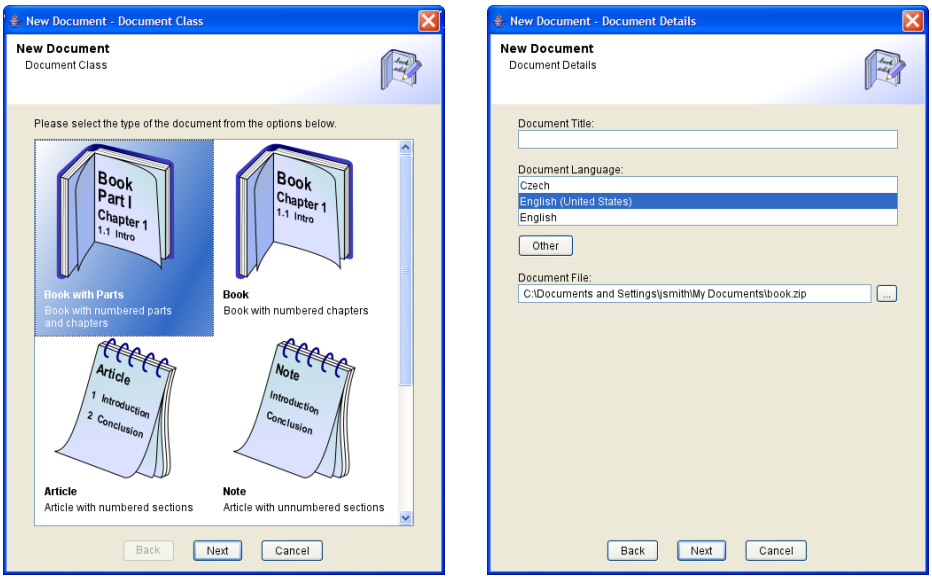


Figure 1.2 New Document Wizard

In the first step, you should specify the *document class* of the document, which represents its overall character. It may be either a **Book**, a **Book with Parts**, **Article** or **Note**. Setting the document class affects many aspects of the document, including its visual style, or the way the BookEditor numbers document sections. If you want to create your own document class or modify existing class, see Document Classes in the Programmer's Guide.

In the second step of the wizard, you specify the document title, language, and file. The *document language* specifies the primary language of the document. You may either select the language from the list, or press the Other button and select a language from the list of all languages. Note that although arbitrary language may be selected, most languages may have limited or no support, in which case English language rules apply to the document. To learn more about language support of the BookEditor, see Chapter 9.

1.6 Managing the Structure of the Document

The BookEditor frame shows the document content in the right part of the main window. The left part shows the document structure, see Figure 1.3.

Note that in this manual, we call any structural part of the document, i.e. Part, Chapter, Section etc., with a generic name *document section.* The upper-left part of the main window shows the Document Global Structure pane, displaying overall document structure, i.e. all its sections, in a hierarchical way. You may use this pane to navigate to a section, or to change the document structure. To add new document section, right-click the parent section, and invoke Create *Section* from the popup menu, where *Section* stands for one of Part, Chapter, etc. To delete existing sections, select them with mouse, and press the Delete key. To move sections, drag them with mouse.

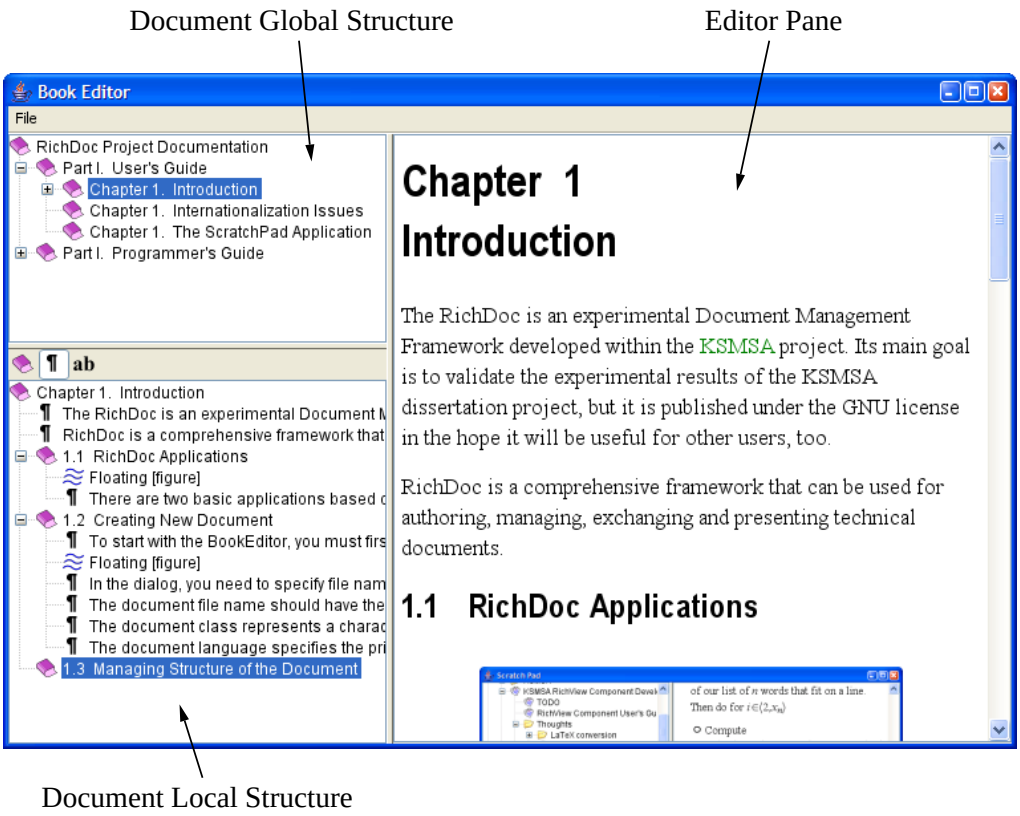


Figure 1.3 The BookEditor Frame

The Editor Pane in the right part of the main frame displays a part of the document selected in the Document Global Structure Pane. Note that the BookEditor always splits documents into editable parts at the level of chapters, that is, the Editor Pane displays one chapter at a time, if the document being edited is a book, or the whole document, if the document being edited is an article.

The Document Local Structure pane, shown in the bottom-left part of the main frame, displays the structure of the selected chapter in more detail. You may select the detail of the view using a toolbar at the top of the pane. Press to display the structure on section level, ¶ to display the structure on paragraph level, or **ab** to display the structure on the word level. This pane can also be used to change the document structure – you may drag nodes with mouse to move them around, drag them while holding the `Ctrl` key to copy them, or delete them by pressing the `Delete` key.

Chapter 2
Editing Basics

This chapter describes basic procedures of document content development.

2.1 Document Sections

In the previous chapter, we have shown how to create new document, and how to add major sections (parts and chapters). This section describes how to further structure the chapter displayed in the editor.

As you will see, many commands may be invoked using the context-sensitive popup menu. In particular, if you want to create a new subsection, just right-click the mouse at the position where the new subsection should be. A menu pops up, see Figure 2.1.

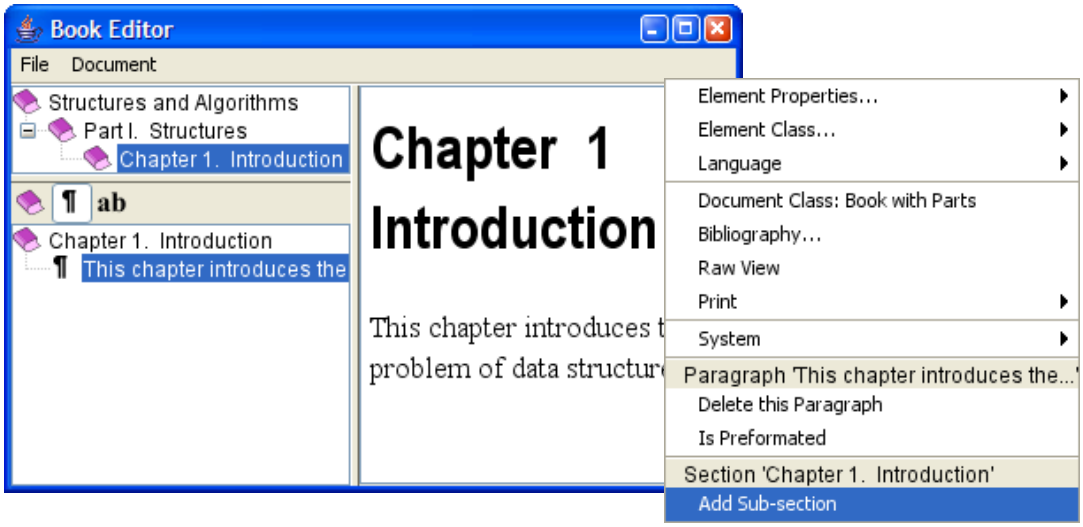


Figure 2.1 Creating new Subsection Using Popup Menu

In the upper part, the popup menu shows available commands that may be applied to the whole document. Below, there are commands available for the current part of the document, from the most specific part—the current paragraph, to more general part—the chapter. If you activate the last command, a new section is added to the document, see Figure 2.2.

If you display a popup menu again, you have three sets of context-sensitive commands: for the current paragraph, for the subsection 1.1, and for the chapter. You may create new section sibling to 1.1, or new subsection, i.e. section 1.1., or join the section 1.1 with the main section. You may also change the structure of the document by dragging sections, paragraphs etc. in the local structure pane, located at the bottom-left part of the main frame.

2.2 Moving and Copying Objects

Frequently, you need to move parts of the document to another location, or create a copy of some document part. The BookEditor offers two ways of moving and copying material. The traditional method is to use the system clipboard to exchange material. The second option is to drag & drop objects in the Local Document Structure pane.

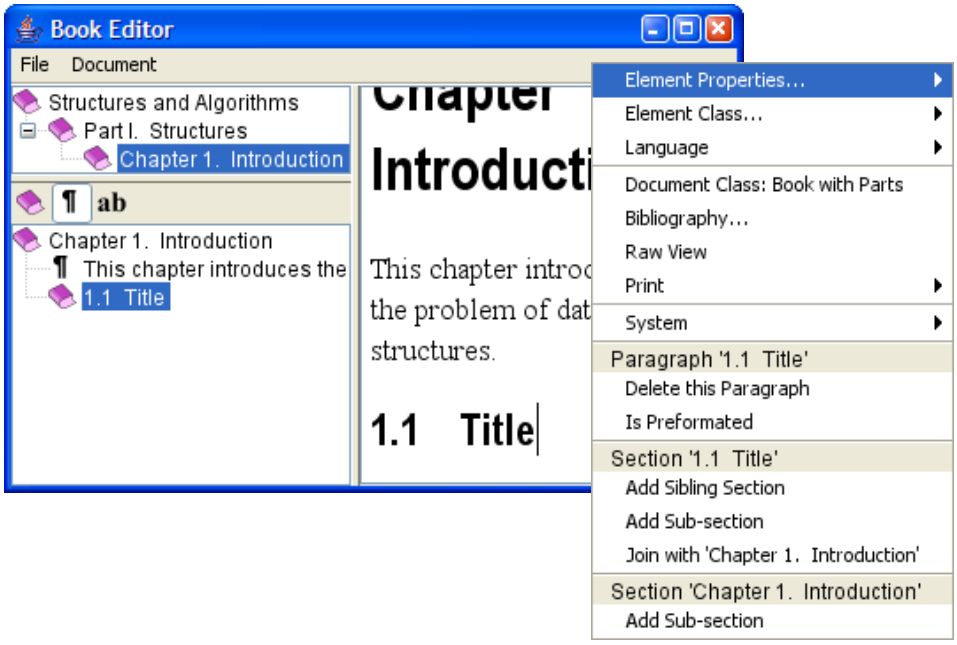


Figure 2.2 Subsection Created

2.2.1 Copying and Moving using the Clipboard

The "Copy & Paste" is a traditional method of moving and copying material. First, you need to make a selection, either by dragging the mouse over a document portion, or by moving the cursor by arrow keys while holding the `Shift` key. The selection is indicated by changed foreground and background color of the selected text. Then you may either copy the selected material to the clipboard, by pressing `Ctrl-Insert`, or cut it (copy it to the clipboard while deleting the original selection) by pressing `Shift-Delete`. Once you have copied the material onto the clipboard, you may move the cursor to a place where you want to paste the material, and press `Shift-Insert`. You may create as many copies of the clipboard as you need.

2.2.2 Copying and Moving using the Document Structure Pane

Alternative method to using clipboard is moving and copying objects by dragging them in one of the document structure panes. This method is more convenient when you want to move or copy well-separated objects, such as whole paragraphs, tables, sections etc. To move an object, just drag it in one of the Document Structure panes. You may drop an object either *beside* existing object, which is indicated by a line above or below the target object, for example you may put a paragraph after existing paragraph. Alternatively, you may put an object *inside* existing object, which is indicated by a rectangle around the target object. For example, you may put a paragraph or a section inside existing section.

2.3 Paragraph Formatting

When you are editing a paragraph, you may apply various formatting styles to the text. You can use toolbar buttons **B** for bold, *I* for italic, T for typewriter, U for underline, and ~~S~~ for strike-through. Alternatively, you may use corresponding keyboard shortcuts `Ctrl+U`, `Ctrl+I`, `Ctrl+T`, `Ctrl+U` or `Ctrl-Shift-S`. If there is some text selected, the formatting is applied to the selection. If nothing is selected, but the text cursor is in the middle of a word, the formatting is applied to the word. Other-

wise, the formatting is not applied to any existing text, but to the newly typed text. If you apply some formatting to text that already bears the formatting, the formatting is turned off.

There are more special-purpose formatting actions. The language formatting (toolbar button EN) can be used to set language for a phrase. If you want to cite a foreign-language phrase, you may want to explicitly set its language, to avoid spelling errors, and to ensure correct hyphenation. The IDX button can be used to associate selected text with index entry, see Section 6.1. The button can be used to cancel formatting under text cursor.

2.4 Creating Cross-References

If you need to insert a hyperlink, also known as cross-reference, to a numbered document element (section, table, figure, equation, or list item), use Insert Hyperlink toolbar button , or just press `Ctrl-L`. A Select Cross-Reference Target window shows up, see Figure 2.3.

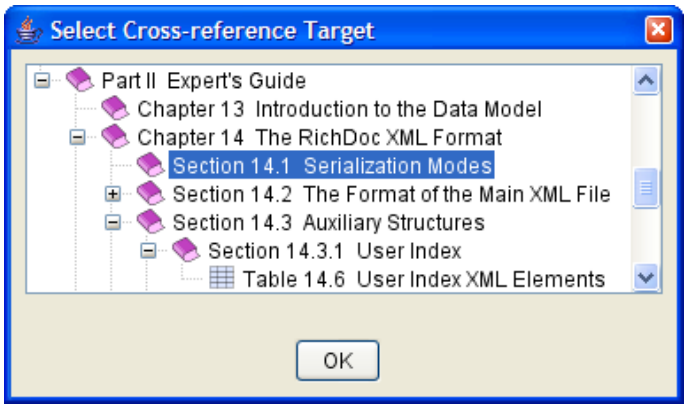


Figure 2.3 Select Cross-reference Target Window

After you select an element in the window, either by double-clicking it or by pressing `Enter`, a *hyperlink* pointing to the the target element is inserted to the document at the cursor position, displaying the title of the target, such as “Chapter 5” or “Equation (3.2)”. The hyperlink is associated with the target element, which means that it automatically changes when the number of the target element changes.

You can also use the hyperlink to navigate to the target element using mouse click. If you want to navigate the hyperlink in editing mode, you need to hold the `Ctrl` key before clicking, otherwise the hyperlink is just selected for editing.

If you want the hyperlink to have text different that the automatically generated title of its target, click it with mouse, and invoke the Editable command from the popup menu. Then just change the hyperlink text as required. This time, however, the hyperlink text does not change when the hyperlink target number changes.

Note that the automatically generated hyperlink text contains, besides the target number, also the type of the target, such as “Chapter” or “Equation”. This is particularly useful when the context in which the document can be used is not known during the document authoring. For instance, you may not know whether particular document section will be at chapter or section level in final document configuration, so you don't want to write the Chapter or Section words directly to section cross-references, but let the framework to synthesize these words automatically. However, in some languages, such as Czech or German, you may need to insert an inflected form of the Chapter or Section words, such as “V Kapitole 5”, which the framework cannot do for you automatically. In this case, toggle the Include Target Name state of the hyperlink, using the respective command from the popup menu. The target text then includes only the number of the target element. The target name, which is no longer a part of the hyperlink text, may be typed manually then.

It is also possible to create a hyperlink to some external page on the Internet, via its Uniform Resource Locator (URL). One possibility is just to type the page's URL to your document. When you

type the start of the URL, such as `http://`, the text is automatically turned into hyperlink, which is indicated with its green color. This is useful if the title of the hyperlink is equal to its URL. Alternatively, you may type a title of the hyperlink, which may be any text, select it, and use the toolbar button. The title is turned into a hyperlink, and you are prompted to provide its URL. The URL may be changed later by the Set URL popup menu command.

2.5 Setting Properties of Objects

2.6 Creating Lists

The Editor offers four types of lists: unordered lists, ordered lists, description lists, and TO-DO lists. Each list has corresponding button in the Paragraph toolbar: , , and , which may be used to insert an empty instance of the respective list into the document. The unordered list uses various types of bullets to indicate its items. The ordered list uses various types of numbering (Arabic numbers, Roman numbers, Latin or Greek letters, etc.) The description list has items consisting of term, typeset in bold, and its definition. The TO-DO list decorates its items with an empty box, indicating uncompleted item, or box with check mark, indicating completed item. You may toggle the uncompleted/completed state by clicking the box.

To add new item to existing list, put the cursor at the end of an existing item, and press `Enter`. A new empty item is added after the selected item. You may also use one of commands from the context menu—you may insert item before or after the current item, or add a new empty paragraph after the list containing the current item.

2.7 Creating Tables

To insert new table to your document, use the or toolbar button. The first option should be used when you want to insert a titled, numbered table, the second is for inserting just a table without a numbered caption. By default, a table with a single empty cell is inserted. If you want to insert a table with more cells, click the button, and drag mouse below and right the button, until the table has the desired size.

When the text cursor is inside a table, the toolbar provides buttons for changing the structure of the table. The or buttons may be used to add new column to the right edge of the table, or new row to the bottom edge of the table, respectively. The and buttons insert new column or row left to or above the current cell, respectively. The and buttons delete the column or row containing the current cell, respectively.

There is also a toolbar for manipulating the current cell. The and merges the current cell with the cell right to it or below it, respectively. You may set the horizontal alignment of a cell to left (), center (), right (), decimal () or justified (). The vertical alignment may be top (), middle () or bottom ().

You may also specify the borders of tables. By default, tables do not have borders, but note that in editing mode a light border is always displayed as a visual aid. You may assign a table some class (see Section 2.5) which adds a border to the table. For instance, the table class `border` causes all cells of the table to have regular border.

You may also override the default border properties individually for each border line. You may choose from a set of predefined border types, or use border with an ad-hoc properties. To modify borders of table cells, press the button in the Table Border toolbar, and select a predefined border type from the list right to the button. Then contour borders you want to set to the selected type, by mouse dragging. The empty item in the list of border types restores the borders to their default, which is usually regular border or empty border, depending on the table class. *You may also specify ad-hoc properties of the border, i.e. its width, color and line style.*

Chapter 3
Equation Editor

The RichDoc framework contains an embedded equation editor, which can be used to insert equations and other mathematical structures into the document. You may insert either an *inline equation* using the $\sqrt{x}$ toolbar button, which does not interrupt the flow of words in a paragraph, such as $e^{i\pi}+1=0$, or a *displayed equation* using the $\overline{\sqrt{x}}$ toolbar button, that is placed on its own line, separated from its neighborhood by a vertical space, such as

$$\int \frac{1}{1+x^2}\,dx = \tan^{-1} x$$

For more information about displayed equations, see Section 3.7.

3.1 Mathematic Text

The most simple material that can be inserted into an equation is text. Text may denote variables, constants, numbers, functions etc. The text is classified into several semantic categories, described in Table 3.1. Each category gets some default typographic style, which can be overridden if necessary. See Chapter 5 for managing visual styles of elements.

Table 3.1 Categories of Mathematical Text

Category	Description	Default Style	Example
`mtext`	generic text	normal	text
`var`	variable	italic	*x*
`const`	constant	normal	*e*
`num`	number	normal	10
`dim`	dimension	normal	kg
`vec`	vector	bold	**a**
`mat`	matrix	bold	**A**
`dom`	domain (e.g. real numbers)	blackboard	$\mathbb{Z}$
`mathSf`	sans-serif	sans-serif	A

If you need to use category that is not listed in the table, you may use the generic category “Generic Text”. Then you may assign it a class using the generic class mechanism, see Chapter 5.

Note that when you start typing text, the class is selected automatically. If you type a letter, the class is set to variable, if you type a digit, the class is set to number. Other classes must be set manually.

It is also possible to insert regular paragraph text into the equation, using ¶ toolbar button. You may apply all properties applicable to normal paragraphs, such as formatting, adding inline graphics, or even inline math. You cannot, however, insert line breaks into the paragraph.

3.2 Operators and Symbols

The equation editor can insert a variety of operators and symbols. The a and Γ button groups may be used to insert lowercase and uppercase Greek letters, respectively. Note that Greek letters are in fact normal mathematic text. The lowercase Greek letters get the variable class by default, while the uppercase letters get the dimension class by default. The $\neq$ and ∞ button groups may be used to insert mathematical operators and symbols, respectively.

TeX/LaTeX users, that are familiar with macro names for symbols and operators, may insert these by typing the macro names. For example, after typing '\equiv' on the keyboard, the symbol $\equiv$ is inserted at the cursor position. The Table 3.2, Table 3.3, Table 3.4 and Table 3.5 summarize the Greek letters, operators and symbols supported by the Equation Editor.

The Table 3.5 shows only special operators that do not have a corresponding key on a regular English keyboard. Operators that do have such key, such as '$<$', '$=$', '$+$', etc., as well as punctuation symbols like ',', ';', '!' etc., can be inserted just by typing their corresponding key.

Table 3.2 Lowercase Greek Letters

Operator	Shortcut	Operator	Shortcut
α	\alpha	β	\beta
γ	\gama	δ	\delta
ε	\epsilon	ζ	\zeta
η	\eta	θ	\theta
ι	\iota	κ	\kappa
λ	\lambda	μ	\mu
ν	\nu	ξ	\xi
π	\pi	ρ	\rho
ς	\varsigma	σ	\sigma
τ	\tau	υ	\upsilon
φ	\varphi	χ	\chi
ψ	\psi	ω	\omega

Table 3.3 Uppercase Greek Letters

Operator	Shortcut	Operator	Shortcut
Γ	\Gama	Δ	\Delta
Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi
Σ	\Sigma	Φ	\Phi
X	\Chi	Ψ	\Psi
Ω	\Omega		

3.3 Structures

Mathematical formulas may contain, besides simple structures like variables, operators and symbols, more complex structures like fractions, radicals, integrals etc. These can be inserted using the mathematical toolbar, or by typing their LaTeX equivalent. When a structure is inserted, a number of empty

Table 3.4 Symbols

Operator	Shortcut	Operator	Shortcut
∞	\infty	∂	\partial
ϑ	\vartheta	ϱ	\varrho
ϕ	\phi	$\Re$	\Re
$\Im$	\Im	$\prime$	\prime
■	\blacksquare	□	\Box
∇	\nabla	\	\backslash
ℓ	\ell	$\surd$	\surd
$\wp$	\schwellL	©	\copyright
↵	\crlf		

Table 3.5 Operators

Operator	Shortcut	Operator	Shortcut
$\geq$	\ge	$\leq$	\le
$\gg$	\gg	$\ll$	\ll
$\geqslant$	\geqslant	$\leqslant$	\leqslant
$\neq$	\ne	$\times$	\times
$\pm$	\pm	$\mp$	\mp
$\cup$	\cup	$\cap$	\cap
$\vee$	\vee	$\wedge$	\wedge
$\forall$	\forall	$\exists$	\exists
$\in$	\in	$\approx$	\approx
$\simeq$	\simeq	$\sim$	\sim
$\cdot$	\cdot	$\equiv$	\equiv
$\propto$	\propto	$\perp$	\perp
$\odot$	\odot	$\oplus$	\oplus
$\leftarrow$	\leftarrow	$\gets$	\gets
$\rightarrow$	\rightarrow	$\leftrightarrow$	\leftrightarrow
$\to$	\to	$\Leftarrow$	\Leftarrow
$\Rightarrow$	\Rightarrow	$\Leftrightarrow$	\Leftrightarrow
$\dots$	\dots	$\vdots$	\vdots
$\ddots$	\ddots	$\iddots$	\iddots
$\cdots$	\cdots	$\circ$	\circ
$\bullet$	\bullet	$\star$	\star
$\leadsto$	\leadsto		

boxes appear, which serve as placeholders for material to be inserted into the structure. You may use arrow keys or mouse to navigate between the components of the structure. It is, of course, possible to nest structures into each other. The Table 3.6 summarizes the variety of mathematical structures supported by the Equation Editor.

Table 3.6 Mathematical Structures

	Shortcut	Description	Example
x_1^2	_ or ^	script	$x_1, 2^n$
$\frac{a}{b}$	\frac	fraction	$\frac{a+b}{a-b}$
$\sqrt[3]{x}$	\sqrt	radical	$\sqrt[3]{x+1}$
Σx	\sum	sum	$\sum_{i=1}^{n} x_i$
Πx	\prod	product	$\prod_{i=1}^{n} x_i$
$\int x$	\int	integral	$\int_0^1 x^2 dx$
$\oint x$	\oint	circular integral	$\oint_0^1 x^2 dx$
$\iint x$	\iint	double integral	$\iint_A x dA$
$\iiint x$	\iiint	triple integral	$\iiint_A x dA$
$\iiiint x$	\iiiint	quadruple integral	$\iiiint_A x dA$
$\int\cdots\int x$	\idotsint	integral group	$\int \cdots \int_A x dA$

not. In the displayed state, they are bigger and take more vertical space. Compare $\frac{x}{x+1}$ and $\dfrac{x}{x+1}$, or $\int_0^1 x dx$ and $\displaystyle\int_0^1 x dx$. The displayed state is enabled when the structure is in an displayed rather than inline equation. Nesting structures also disables the displayed state, i.e. structures nested in other structures are no longer displayed. If you need to set the displayed state manually, check the Displayed box in the Structure toolbar. The unchecked state (☐) indicates that the displayed state is always disabled, the checked state (☑) indicates that the state displayed is always enabled, and the intermediate state (■) indicates that the displayed state is inherited from the outer context.

The *Limits* state determines whether indexes of structures, which denote its limits, are above and below the structure symbol like $\int_a^b x$, or right to the structure symbol, like $\int_a^b x$.

3.4 Fences

We call *fences* mathematical structures like parentheses, brackets, curly brackets etc. Types of fences are summarized in Table 3.7. You can insert the empty fence box either by a toolbar button under the fence button group (x), or by typing its shortcut. By default, the fence box is automatically completed so that the left symbol matches the right symbol.

If you place the text cursor inside the fence, the Fence toolbar shows up in the toolbar window. Here you can set the type of each fence side separately. This allows you to create asymmetric fences, such as $\langle 0,1\rangle$.

By default, the fence scales automatically with the size of its content. You may, however, set the fence size manually, by selecting a Size value in the Fence toolbar. This allows you to create equations like this: $\big((x+1)(x+2)\big)^2$. Normally, the outer fence would have the same size as the inner fences.

Table 3.7 Fences

	Shortcut	Description	Example
(x)	(	Round Brackets	(x)
$[x]$	[	Square Brackets	$[x]$
$\{x\}$	{	Curly Brackets	$\{x\}$
$\|x\|$	|	Pipes	$\|x\|$
$\langle x\rangle$	`\langle`	Angle Brackets	$\langle x\rangle$
$\lfloor x\rfloor$	`\lfloor`	Floor Brackets	$\lfloor x\rfloor$
$\lceil x\rceil$	`\lceil`	Ceil Brackets	$\lceil x\rceil$

3.5 Arrays

Arrays are one or two-dimensional structures similar to tables. The process of creating arrays is thus similar to the process of creating tables, described in Section 2.7. You may use arrays to create matrices and vectors, or just to visually structure mathematical material. For instance, you may easily create equations like this:

$$\begin{pmatrix} 1 & 3 & -1 \\ 2 & 1 & 3 \\ -5 & 8 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 7 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

To insert a new array containing single empty cell to the cursor position, either use the toolbar button, or type `\array` from the keyboard. If you want to create an array with more cells, drag the toolbar button with mouse as if creating a table. To change the structure of an array, use the toolbar buttons as described in Section 2.7. The borders of arrays are also specified in the same way as borders of tables.

Note that it is not possible to specify the vertical alignment of array cells. The alignment is always according to the mathematical typographic style, i.e. the cells in the same row are aligned to the mathematical axis, running mostly through the vertical center of the cells. It is, however, possible to specify the horizontal alignment of the cells.

3.6 Spacing

The equation editor tries to insert appropriate spacing between elements of equations, according to typographical conventions for typesetting mathematic material. However, you may sometimes want to insert explicit horizontal spacing between elements, either positive, i.e. to extend the existing space, or negative, to reduce the existing space. To insert a space to the cursor position, use a toolbar from the | button group, or type the corresponding shortcut. To insert several spaces at once, click the toolbar button, and drag to the right. Types of spaces are summarized in Table 3.8. The space width value is relative to the font height. That is, the largest space, `\quad`, is as wide as a character cell height.

3.7 Displayed Equations

This section describes options that can be used only with displayed equations.

A displayed equation may be numbered. To toggle a numbered/unnumbered state of a displayed equation, right-click the equation with mouse, and select Has number from the popup menu. You may also insert equation that is numbered by default, using the toolbar button.

Displayed equation may be split into several rows. Equation rows are inserted and removed much the same way as the table rows, i.e. using the toolbar buttons (add row), (insert row) and (delete row), but from the Displayed Equation toolbar. If the equation is numbered, you may de-

Table 3.8 Horizontal Spaces

Width	Button	Shortcut	Example	Width	Button	Shortcut	Example
1/16		\, or \thinspace	$\Rightarrow\Leftarrow$	−1/16		\negthinspace	$\Rightarrow\!\!\Leftarrow$
1/8		\: or \medspace	$\Rightarrow\Leftarrow$	−1/8		\! or \negmedspace	$\Rightarrow\!\!\Leftarrow$
1/4		\; or \thickspace	$\Rightarrow\Leftarrow$	−1/4		\negthickspace	$\Rightarrow\!\!\Leftarrow$
1		\quad	$\Rightarrow\quad\Leftarrow$				

cide whether the whole equation has a number, or whether each of its rows gets its own number, using the Number per Row command from the popup menu.

The equation rows may be split into separately aligned groups using *tabs*. You may insert either right-aligning tab or left-aligning tab using the toolbar buttons and , respectively. A right-aligning tab aligns the group before it to the right, while the left-aligning tab aligns to the left. You must put a tab into each row to take effect – the group of corresponding tabs get aligned on the same horizontal position. A row may contain several tabs, but all the rows should contain the same number of tabs. For example, the following equations use right-aligning tabs before the equality signs:

$$\begin{aligned} x+1 &= 10 \\ x-y+2 &= 2 \\ y &= 6 \end{aligned}$$

Note that you may use arrays (see Section 3.5) to achieve a similar effect. However, aligned rows are more appropriate for specifying independent equations, that are only mutually aligned. With arrays, it is also not possible to assign numbers to rows.

Besides rows corresponding to equations, you may also insert rows containing paragraphs, using buttons and , for adding and inserting paragraph rows, respectively. This is useful if you need to insert paragraph material in the middle of a displayed equation without interrupting it, to preserve numbering or alignment of the equation rows. For instance, you may create displayed equations like this:

$$\begin{aligned} A_1 &= N_0(\lambda;\Omega)-\phi(\lambda;\Omega'), \\ A_2 &= \phi(\lambda;\Omega')-\phi(\lambda;\Omega), \end{aligned}$$

and

$$A_\infty = \mathcal{N}(\lambda;\omega).$$

Note that the equations are properly aligned, which wouldn't be possible without putting the interrupting paragraph *inside* the displayed equation.

3.8 Editing Equations

The Equation Editor supports moving and copying objects, as described in Section 2.2. Both methods are supported, that is, you can move or copy objects either using the clipboard, or using the Local Document Structure tree. You must, however, switch the Local Document Structure to the finest resolution, using the **ab** button. The pane is also useful for displaying the structure of complex equations in a hierarchical way. For instance, the equation below is displayed as shown in Figure 3.1.

$$\int \frac{1}{1+x^2}\,dx = \tan^{-1}x$$

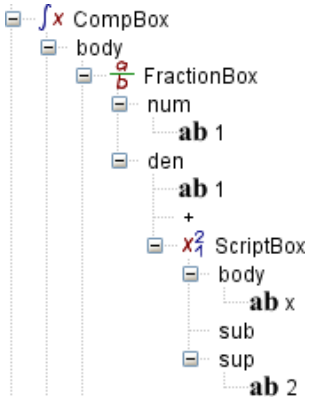


Figure 3.1 Hierarchical View of a Portion of an Equation

instance, in the example in Figure 3.2, there are two empty boxes at the very left part of the formula. Although it is not obvious, there is a difference: the first box is *outside* the script box, whereas the second is *inside* the script box. The Equation Editor distinguishes these two positions, although the distinction has no effect on the final appearance of the formula. In this particular example, you should put material in the former box, i.e. outside the script box, because the reader of the formula would understand the material this way. That is, you should always use fences to give visual guidance to the priority of operations, even if the Equation Editor allows you to explicitly specify the priorities without giving that visual guidance.

$$\left(\frac{x+1}{x-1}\right)^{2}$$

Figure 3.2 Empty Boxes of Complex Formula

Chapter 4
Drawings

The BookEditor includes embedded editor for creating figures containing 2D graphics. With the editor, you may create figure containing curved shapes, boxes with or without text text, links, arrows, or raster images.

4.1 Creating New Drawing

To insert new empty drawing into the document, use Insert Drawing ( ) or Insert Titled Drawing ( ) toolbar command. An empty drawing area is added to the document, see Figure 4.1. If you click the drawing area, a Drawing toolbar appears in the toolbar window. Using the toolbar, you can add drawing elements to the drawing.

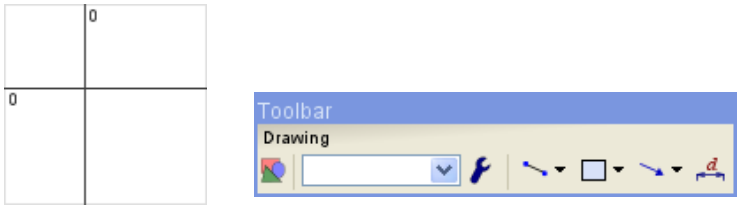


Figure 4.1 Empty Drawing Area and Drawing Toolbar

4.2 Adding Graphical Elements

To add new elements to the drawing area, you must first select one of the *drawing tools* using the Drawing Toolbar. This can be done using one of the drawing tool buttons of the toolbar, see Figure 4.1. The tools are described in detail in subsequent sections.

When you select a particular tool, the New Element Toolbar is added to the toolbar window, see Figure 4.2. You may use the toolbar to specify properties of drawing element *before* it is added to your drawing. This is useful if you want to add several objects with certain properties. The properties include background color ( ), foreground color ( ), and more element-specific properties accessible using the button. You may, of course, also change the properties of elements that have been added to the drawing already.

Figure 4.2 The New Element Toolbar

The first icon on the toolbar indicates the kind of object to be added to the drawing. The yellow star suggests that the properties apply to the new, prototype element, that has not been added to the drawing yet. When you move the mouse cursor over the drawing area, the cursor changes to indicate the kind of tool that is currently active. When you are finished with the tool, you may press the ESC key or the right mouse button to return to the normal mode.

4.2.1 Curved Shapes

To add line, quadratic curve, Bézier curve or arc, use one of the drawing tools, , , or , respectively. Then click the left mouse button to designate the segment start, and drag to specify its end. With this tool, you may create shapes containing only one segment.

To modify existing shape, make sure that no tool is active, and click a shape to select it. The Shape Toolbar shows up in the toolbar window. A number of boxes appears along the selected shape, see Figure 4.3. These boxes represent *manipulation points* that can be dragged to change the selected shape.

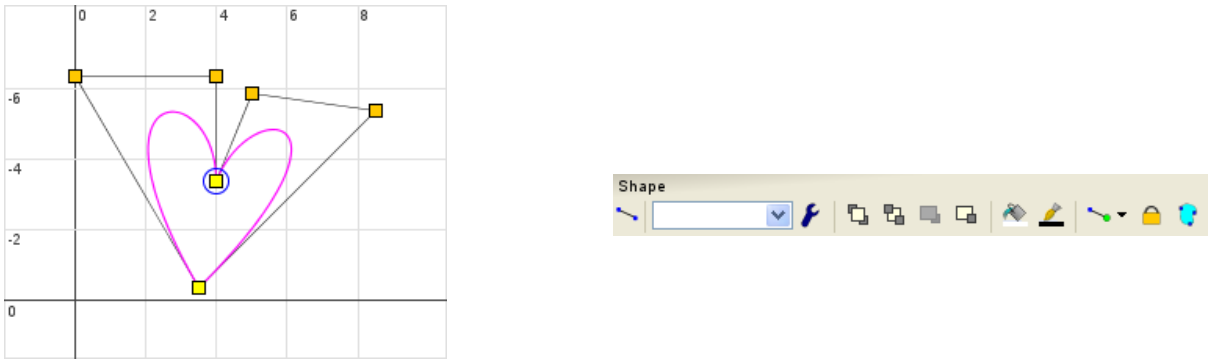


Figure 4.3 Manipulation Points of Shape Segments

To add more segments to the selected shape, use any of buttons of the Shape toolbar: , , , or . Then use mouse clicks to add segments to the selected shape. Please note the difference between the and tools: the former creates new shape, while the latter adds new segment to the selected shape. To delete a segment, right-click it with the mouse, and select Delete Segment from the pop-up menu.

By default, shapes are *open*, meaning that they have the beginning and the end. You may *close* a shape using the button, i.e. permanently connect the beginning and the end. Press the button again to open a closed shape. Closed shapes may be *filled* with a color. To toggle filled/transparent state, press the toolbar button.

4.2.2 Boxes

You can add boxes of various shapes to the drawing. The boxes may optionally contain text. To add new box to the drawing, use any of tools according to Table 4.1. Then click mouse to define upper-left corner of the box, and drag to the right and bottom to define the box size. You may toggle the plain / text state later using the A button.

Table 4.1 Add Box Tools

	Plain	With Text
Rectangular		a
Rounded		a
Elliptic		a

The block with text may contain any block content, e.g. a paragraph, series of paragraphs, list, etc. You can set the preferred size of a block by dragging its yellow control point, see Figure 4.4. The text is automatically wrapped to the preferred width. If the preferred height is too small to accommodate the text, the box is automatically widened so that the text fits into the box.

To create a box containing a table, use the tool. The tool creates an invisible box containing a table. Again, you may drag the yellow point to set the preferred width of the table.

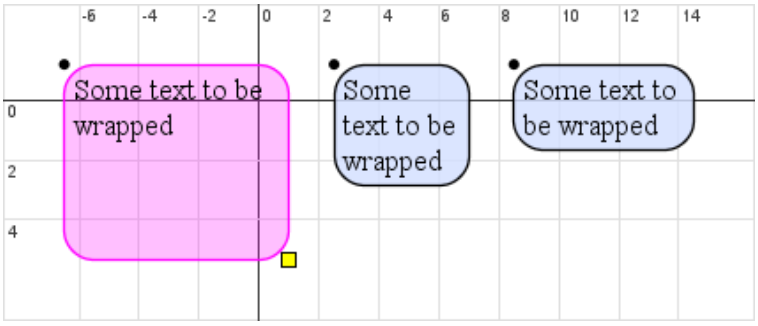


Figure 4.4 Wrapping Text in a Box

4.2.3 Bitmaps

To add a raster image (also known as bitmap) to your drawing, just select the drawing, and paste the image previously copied to the clipboard, by pressing `Shift` + `Insert`. You may also insert an image from file, using the toolbar button. The supported image formats are GIF, JPEG and PNG. You can use the popup menu to change the properties of the bitmap. Use the Set Scale command to enlarge or reduce the natural size of the bitmap. The Crop command cuts away margins having the same color, see Figure 4.5. To mark some color as transparent, use the Set Transparent Color, and click the image to select a color that you want to make transparent.

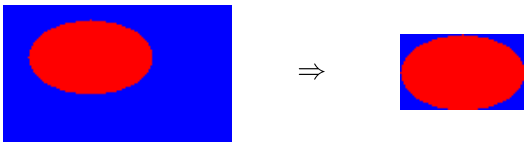


Figure 4.5 Cropping a Bitmap

4.2.4 Links

Solid elements, i.e. boxes, bitmaps and closed shapes, may be connected with *links*. A link is a straight, angular or curved connector line connecting two points. The endpoints may be associated with host elements, which causes the link to change automatically when the host element moves. Link endpoints may also have decorations such as arrows or bullets.

To add new link to the drawing, use one of the following toolbar buttons: for straight link, for curved link, for corner link, or for rectangular link. You may drag the endpoints of a link to designate the link start and end. If you drag a connector over another element, the link endpoint snaps to the geometrical center of the selected element. If you drop the endpoint like this, the link endpoint is permanently associated with the element. If you want to prevent snapping of the endpoint while dragging over an element, hold the `Ctrl` key during dragging.

Note that when link endpoint is associated with some element, the link virtually ends in the center of the host element, is automatically trimmed by the boundary of the host element, see Figure 4.6. Also note that the curved link may have the start point identical to the end point, see Figure 4.6*e*. In this case, the link has, besides a control point controlling its shape, a control point for rotating the link around its collapsed endpoints.

If you create a complex diagram containing many boxes and links, you may tell the drawing to automatically layout the diagram. To do this, select the root element of the diagram, and invoke the Auto Layout command from the popup menu, see Figure 4.7. From now on, the position of all boxes except the root box is controlled automatically. You may disable the auto-layout state by invoking the command again, and then manually refine the structure of the diagram.

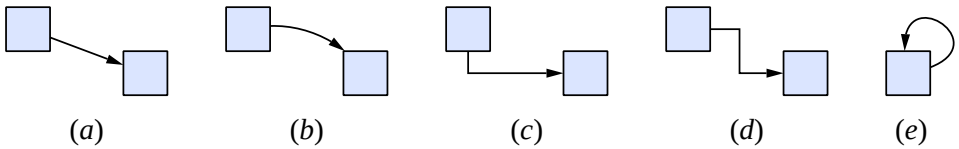


Figure 4.6 A Variety of Links. (*a*) Straight Link, (*b*) Curved Link, (*c*) Corner Link, (*d*) Rectangular Link, (*e*) Loop Link

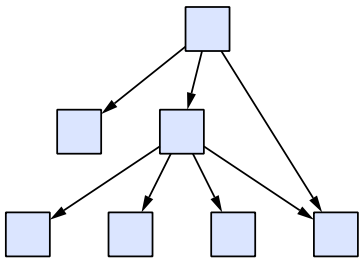


Figure 4.7 Automatically Laid-out Diagram.

4.2.5 Linear Dimensions

You may add add linear dimensions to your drawings using the toolbar button, see Figure 4.8*a*. Four control points affect the shape of the linear dimension, see Figure 4.8*b*. Two points control the position of the arrow, other two points control endpoints of the perpendicular extension lines, and the green point control the relative position of the text. Note that as with text, you may snap the endpoints of the extension lines to some objects, to permanently associate the dimension with the object being measured. When the position or size of the measured object changes, the dimension is updated accordingly.

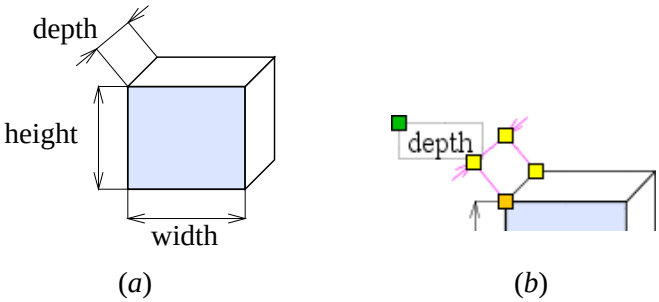
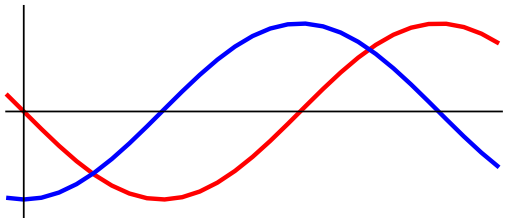


Figure 4.8 Linear Dimensions Example. (*a*) Variety of Linear Dimensions, (*b*) Control Points of a Linear Dimension

4.2.6 Custom Boxes

At times you might need to add material to your drawing that is better described using an algorithm, or script, rather than by interactively drawing it. In this case, you may want to use the Custom Box tool. To add the Custom Box to your drawing, use the tool, and drag mouse to define position and size of the box. Then use the to open the script editor, and enter the script that draws the content of the box. For an example of using Custom Box, see Figure 4.9. The detailed description of the script language is beyond the scope of this document. Also note that this feature is still under construction.



```
List sin = list(); List cos = list();
translate(0,height/2); double scale=3;
width = width/scale-0.2; scale(scale);
translate(0.2,0);
for (double i=-0.2; i<width; i+=0.2) {
  sin.add(p(i, sin(i)));
  cos.add(p(i, cos(i))); }
setColor(red);   draw(poly(sin));
setColor(blue);  draw(poly(cos));
setColor(black); setStroke(0.02);
draw(line(0,-1.2,0,1.2));
draw(line(-0.2,0,width,0));
```

Figure 4.9 Custom Box Example

4.3 Manipulating Elements

When you have drawing containing a number of elements, its easy to move, copy or delete them. To move an element, just drag it with mouse. If the element has manipulating points, be sure to click point outside the manipulating point to initiate move gesture instead of manipulation. If you want to drag more elements at once, select them by clicking outside of any element, and extending the selection using mouse dragging. Then drag any of the selected objects to move the whole selection. You may also toggle object selection by clicking it while holding the `Ctrl` key. To delete selected objects, just press the `Delete` key.

Note that you cannot change the size of the drawing area; the area resize automatically to accommodate all objects in it. You may drag an object outside the area; the object temporarily disappears, but when you drop it, the area resizes so that the distant object fits to it. Also note that the relative position of objects in the drawing coordinate system has no effect on final visual appearance of the drawing. That is, if you move *all* objects to any new position, there is no visible change in the drawing.

To copy selected objects, you have two options. Either drag one of selected objects with the mouse, or use the clipboard, i.e. use `Ctrl` + `Insert` to copy selection to the clipboard, and `Shift` + `Insert` to paste it. The clipboard way of copying objects can be used also to copy elements between different drawings. Note that you can also copy entire drawing using the Local Document Pane, as described in Section 2.2.

All elements in the drawing have its *Z-order*, i.e. position on the imaginary axis perpendicular to the screen. The later the element is added to the drawing, the closer it is to you. The Z-order is particularly important when the elements overlap; the elements that are closer obscure the elements that are behind them. The Z-order of object can be additionally changed. To move object one position up or down, use the buttons or , respectively. Button brings object above all objects, while button sends object beneath all object.

4.4 Setting Visual Properties

4.5 Text Measurement

Since the BookEditor is not strictly WYSIWYG as described in Section 1.1, the text style used in drawings can be changed by applying different visual style to the drawing. This may break the structure of the drawing. It is thus not recommended to change fonts used in drawings from the setting used to create the drawing.

Moreover, the BookEditor paints text in a way optimized for good appearance on low-resolution displays. This means that each letter has integer width and height, causing that the text does not scale

uniformly. For example, if you scale a letter 11 pixels wide to 50% size, its width will be either 5 or 6 pixels, a value quite different from the correct width of 5.5 pixels. The measurement of text may be thus inaccurate with errors of order up to tens of percent, which may easily break your drawing. If your drawing requires accurate measurement of text, you may set its Fractional Metrics attribute. With this attribute, the texts scale uniformly, but the spacing of letters may be uneven.

4.6 Transformations

Existing objects in the drawing may be transformed by applying transforming operations to the drawing. This feature is particularly useful in connection with animations, described in Section 4.7. The four basic transforming operations are demonstrated in Figure 4.10.

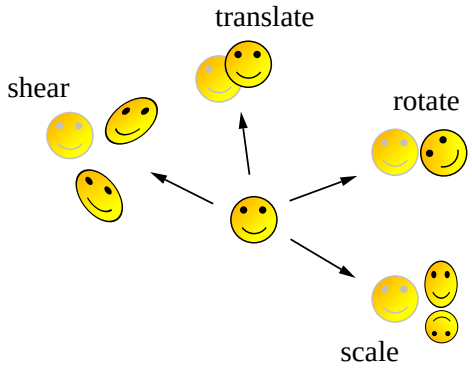


Figure 4.10 Transforming Operations

To create new transform in a drawing, select the drawing, and use its toolbar button. The Transform Properties dialog shows up, see Figure 4.10. Initially, the list of operations is empty. Press the New button, and select one of the four transformation types. The list of operations is populated with parameters corresponding to the created operation. Each transforming operation has different parameters, as specified in Table 4.2.

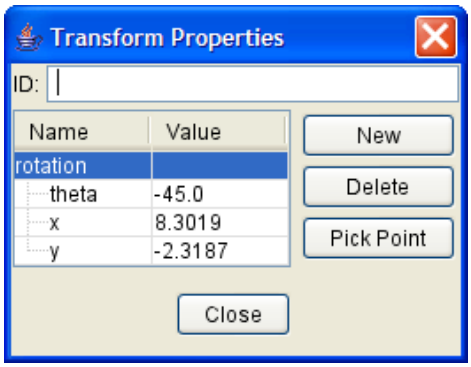


Figure 4.11 Transform Properties Dialog

Each transformation may have an identifier, which can be set in the Transform Properties dialog. This identifier may be later used e.g. to programmatically change the parameters of the transform, to achieve an effect of animation, see Section 4.7.

Note that it is possible to add more operations to single transformation object. The operations are then applied to objects subsequently. For operations having center point, i.e. all operations except translation, it is useful to specify the center point by selecting it directly in the drawing. This can be done by pressing the Pick Point button in the Transform Properties dialog, and then clicking with

Table 4.2 Parameters of Transforming Operations

Operation	Description	Parameters
Translation	Moves objects by a vector	x, y... translation vector
Rotation	Rotates objects by an angle around a point	x, y... center point, *theta*... rotation angle in degrees (counterclockwise)
Scale	Scales objects	x, y... center point, sx... horizontal scale, sy... vertical scale; negative scale values may be used to flip the object
Shear	Shears objects	x, y... center point, sx... horizontal shear, sy... vertical shear

mouse any point in the drawing underneath the dialog. The x, y variables of the currently selected operation are then selected appropriately.

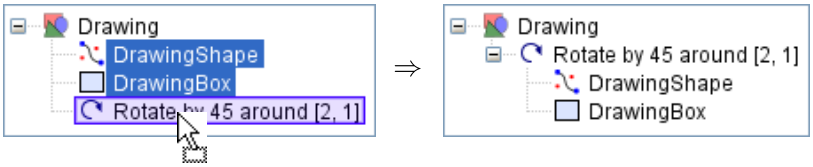


Figure 4.12 Moving Objects into Transformation

When transformation object is prepared, we need to associate it with drawing objects to be transformed. This can be done by dragging the objects into the transformation in the Document Structure Pane, see Figure 4.12. When we drop objects into transformation, the transformation's operations begin to apply to its child objects. It is also possible to compose transformations, i.e. a child of transformation may be another transformation.

4.7 Animations

You may turn a still-image prepared with the drawing editor into an animated image, by introducing animation rules. To add animation rules to the drawing, select the drawing, and use the Animation command from the popup menu. The Animation Rules dialog pops up, see Figure 4.13.

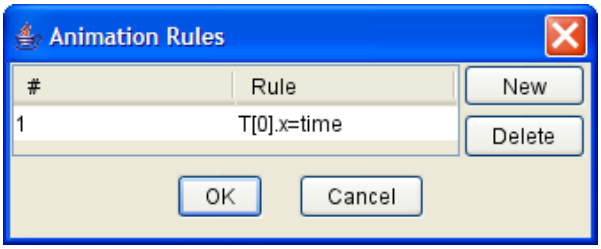


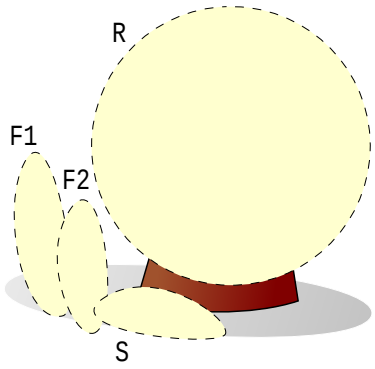
Figure 4.13 Animation Rules Dialog

Every rule has the form *object* = *value*, where *object* is an expression that evaluates to object that can receive a value, and *value* is an expression that evaluates to object that can provide a value. During animation, the rule is executed periodically to evaluate the expression on the right side and update the state of the object at the left side. The objects that can appear on the left side of an animation rule are summarized in Table 4.3. The right side of a rule may be an arithmetic expression, involving mathematical operators, functions, and the animation parameter `time`. The parameter represents the animation real time in seconds, starting from zero at the time animation started, i.e. the first frame of the animation has been displayed.

Table 4.3 Scriptable Properties

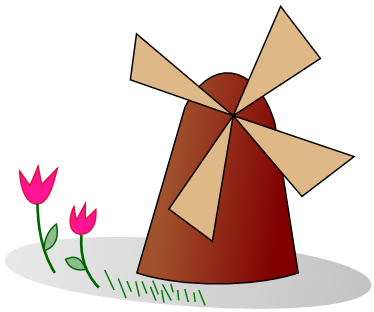
Property	Description
transform`[`*n*`]`.*parameter*	Parameter of *n*-th operation of *transform*. The parameters depend on the type of transforming operation, see Section 4.6.
object.*color*.*component*	Sets specific color of given *object*. The *color* may be one of `foreColor`, `backColor`, `backColor2` or `borderColor`. The *component* may be one of *r*, *g*, *b* for specifying color components in the RGB color model, or *c*, *m*, *y* for specifying components of the CMY color model. The color components should have value in the interval $\langle 0,1 \rangle$.

the picture. Figure 4.14*b* shows the animation script. Figure 4.14*c* shows the final animated picture.




```
R[0].radTheta=time
F1[0].radTheta=sin(time*2)/10
F2[0].radTheta=sin(time*2)/10
S[0].sx=sin(time*2)/(0-8)
```

(*a*) (*b*)



(*c*)

Figure 4.14 Sample Animated Figure: (*a*) Parametrized Picture, (*b*) Animation Rules, (*c*) Animated Picture

Chapter 5
Managing Visual Styles

As we already mentioned, the key difference between the RichDoc framework and classical strictly WYSIWYG word processors is sharp separation of document structure from its visual representation. To get some specific visual representation of a document, we must apply some visual formatting rules to the document. These rules are defined in a structure called *style sheet*. Since the style sheet does not depend on the document, we may use different style sheets for the same document to get alternative visual representations of the document, see Figure 5.1. It is, of course, also possible to apply the same style sheet to a group of documents to provide uniform look and feel for such group.

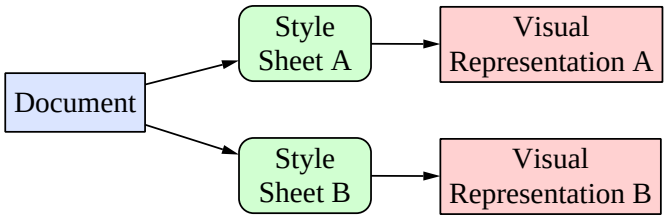


Figure 5.1 Applying Visual Rules to a Document

The style sheet structure may be split into several *cascades*, to provide rules of different levels of generality, see Figure 5.2. There may be general style files providing rules applicable for wide variety of situations, which may be refined with more specific style files providing exceptions from general cases. When the visual presentation module needs to find a specific visual formatting rule, it first checks the more specific style files, and continues up the cascade until an appropriate rule is found. The style language used to define style files is described in detail in Chapter 19.

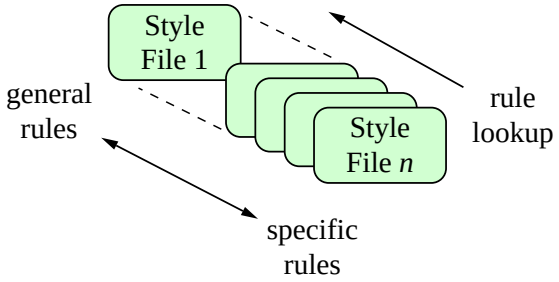


Figure 5.2 Cascading Style Sheets

As described in Section 2.5, it is possible to assign each document element some visual attributes such as font and color, in an ad-hoc manner. This is convenient for handling special cases, but for regularly occurring patterns, it is recommended to treat visual attributes more systematically.

For each document element type, such as paragraph, paragraph fragment, table or section, has some standard set of visual properties that is acquired from the style sheet currently in use. We may, however, declare special named variants, called *classes* for any element type, and override the visual properties for these. For some elements, there are already some predefined classes, such as class `border` for tables, but you can create your own classes. For each class, you may define a set of attributes in the same way as you define attributes for a concrete document elements. You can then *associate* instances of document elements with an appropriate class, which causes the element to *inherit*

all visual attributes from its class. You can later change the attributes of the class, and all elements associated with that class immediately reset their appearance to reflect the changes. It is of course possible to override the inherited values with ad-hoc settings for each individual element.

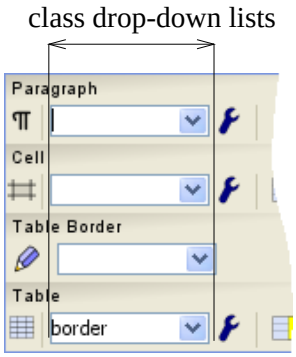


Figure 5.3 Specifying Class from the Toolbar

We can specify the name of element class using the toolbar, see Figure 5.3. Select the element for which you want to set the class, and access the drop-down list on its toolbar. Here either type the class name, or select it from the list. When the class is specified, the element view immediately acquires visual attributes from the class and resets its appearance accordingly.

Chapter 6
Index, Glossary and Bibliography

This chapter describes how to add index and glossary to your books, and how to manage database of bibliographical references.

6.1 Creating Index and Glossary

Quality printed books have index and/or glossary sections in their end. The BookEditor framework supports authoring the index and glossary, and generates the corresponding sections automatically. The index section usually contains alphabetically sorted list of concepts, with references to pages where concepts are mentioned or defined. The glossary section contains a list of concepts, each accompanied with short definition.

To create an index, you should define the list of concepts, and associate them with elements in the document body. We shall call a definition of single concept an *index entry*. An index entry is associated with one or more elements in the document body, see Figure 6.1. An index entry may also provide a *glossary definition*. To define index entries, select the Index... command from the popup menu. An Index Editor window pops up, see Figure 6.2.

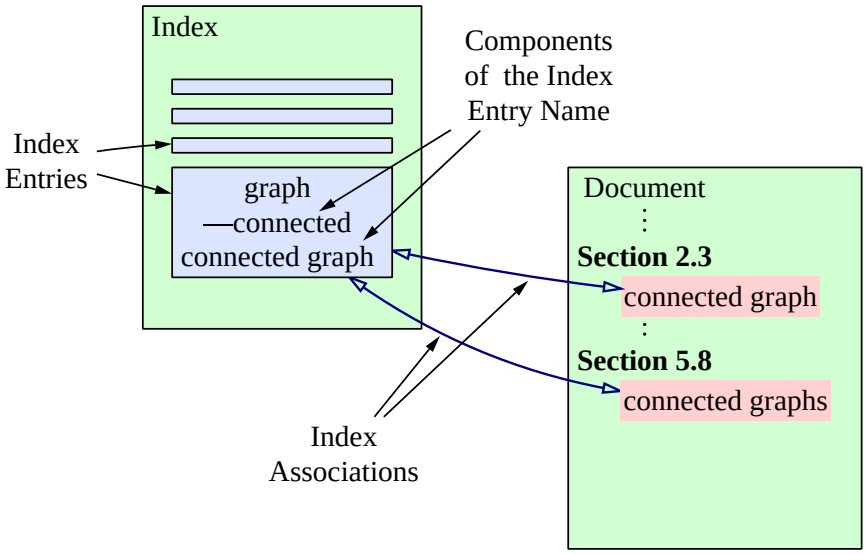


Figure 6.1 The Index Structures

The window contains four panes; the Index Entries pane contains a list of all index entries in the index. The Selected Entry pane contains an editor where you can edit the components of the selected entry. The Glossary Definition pane provides a space for optional glossary definition corresponding to the selected entry. The Associations pane displays a list of elements in the document that are associated with the selected entry.

As we can see in the figure, the entries may be split into several components, specified from the most important, to the less important ones, such as *graph—connected*. Some entries may share some of the first component(s), and differ only by the less significant components (e.g. *search—breadth-first*, *search—depth-first*). Such component sharing affects the final visual appearance of the index section in usual way; the common component(s) appear only once, and the sec-

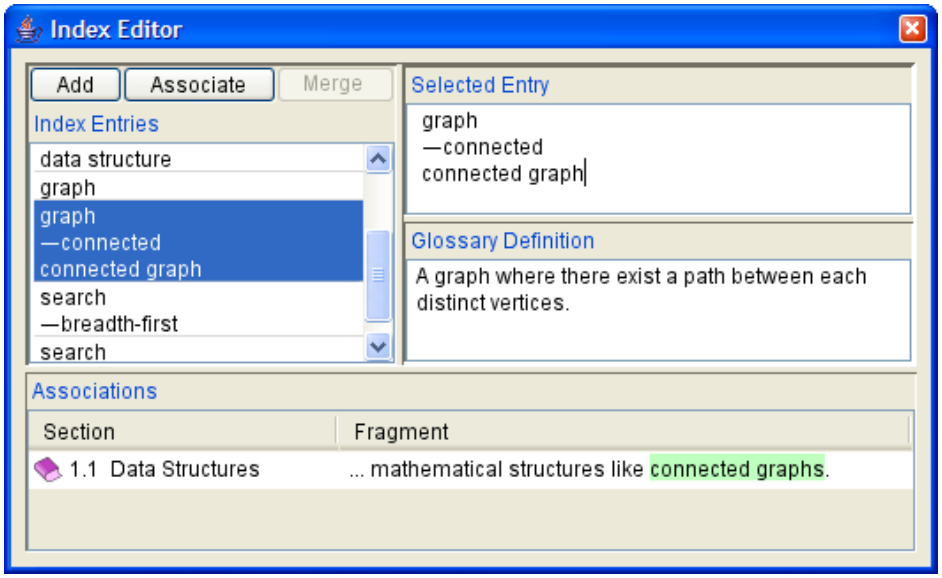


Figure 6.2 Index Editor Window

ondary components are listed for the common primary component. In the index editor, however, all components must be explicitly specified. Also note that index entries may have several alternative names. The second entry in our example has two names: *graph—connected*, and *connected graph*. The entry thus appears twice in the generated index section. The final appearance of the index section in our example is shown in Figure 6.3.

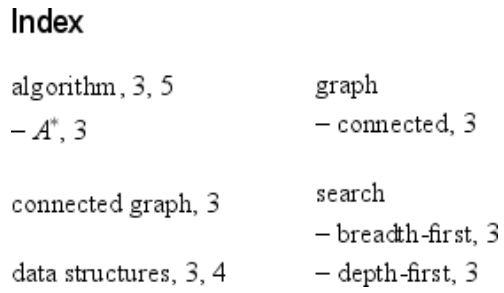


Figure 6.3 Final Appearance of the Index Section

To add new entry to the index, press the Add button, and update the entry text in the Selected Entry pane. To create alternative names, just put each name on a separate line. To create names with two or more components, put the secondary component below the primary one, and use the hyphen ('-') character to indent the secondary component. To define a third-level component, put it below the second-level component, and use two hyphens to indent it, etc. You may use as many levels as you need, but it is recommended not to use more than three levels.

Our example also shows that you may use not only plain text, but also formatting (italic, bold, etc.), inline equations, inline graphic, etc. In fact, each component of an index entry name is a full-featured paragraph.

When index entries are defined, we want to associate them with the objects of the document content. We may associate entries with existing objects, such as sections and paragraphs, but more frequently, we want to associate them with parts of paragraphs corresponding to phrases. To associate an index entry with document element, select the index entry in the list, and place the cursor in the document editor area to the desired element, or select a desired phrase. Then press the Associate button. If some text is selected, it is associated with the selected index entry. If no text is selected, a menu pops

up, offering various options of which element may be associated with the entry. After selecting particular object from the menu, the association is established.

It is also possible to define index entries the other way round: first to locate a phrase in the document, and then to define an index entry, associated with that phrase. This can be done using the IDX toolbar button from the formatting group of buttons (under the **B** button). For example, after selecting the "data structures" phrase in the editor and pressing the IDX, an Index Editor pops up as shown in Figure 6.4. We see that index entry identical to the selected phrase is added to the index, and is associated with the phrase occurrence in the document content. If the phrase is not appropriate for the index entry name, we may easily modify the name of the index entry, but the association is preserved.

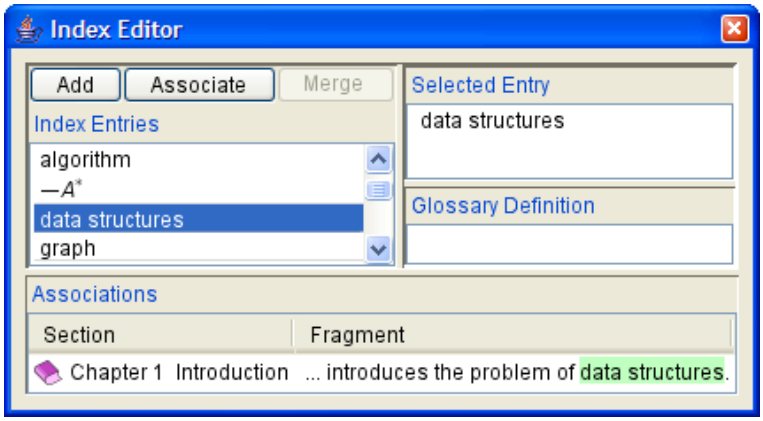


Figure 6.4 Creating Index Entry for a Phrase

If you later decide to merge two or more index entries into a single index entry, you may select them in the Entries pane (use arrows with `Shift` key, or click entries while holding the `Ctrl` key), and then press the Merge button. The list of occurrences associated with original index entries is merged into a single list, as well as the names of the selected index entries.

You may cancel association of index entry with document element, by selecting the association in the Associations pane, and pressing the `Delete` key. You may also delete the whole index entry and all its associations, by selecting the entry in the list and pressing the `Delete` key.

6.2 Creating Bibliographical References

For scientific documents, you often want to attach a list of bibliographical references to the end of the document, and refer to the list from within the document text. This process is separated into several independent tasks:

1. Create an inter-document database of bibliographical references.
2. Create a list of references to be associated with concrete document, to be printed in the Bibliography section.
3. Put references to the list created in step 2. into the document text.

6.2.1 Managing the Database of Bibliographical References

To add bibliographical references to your document, you must first prepare the database of such references. To manage the database, select the Bibliography... command from the popup menu, and press Database button. A Bibliography Database window shows up, see Figure 6.5.

The top part of the window displays a hierarchical list of bibliographical references. You may organize the references into groups. To create a new group, press the New Group button. To add

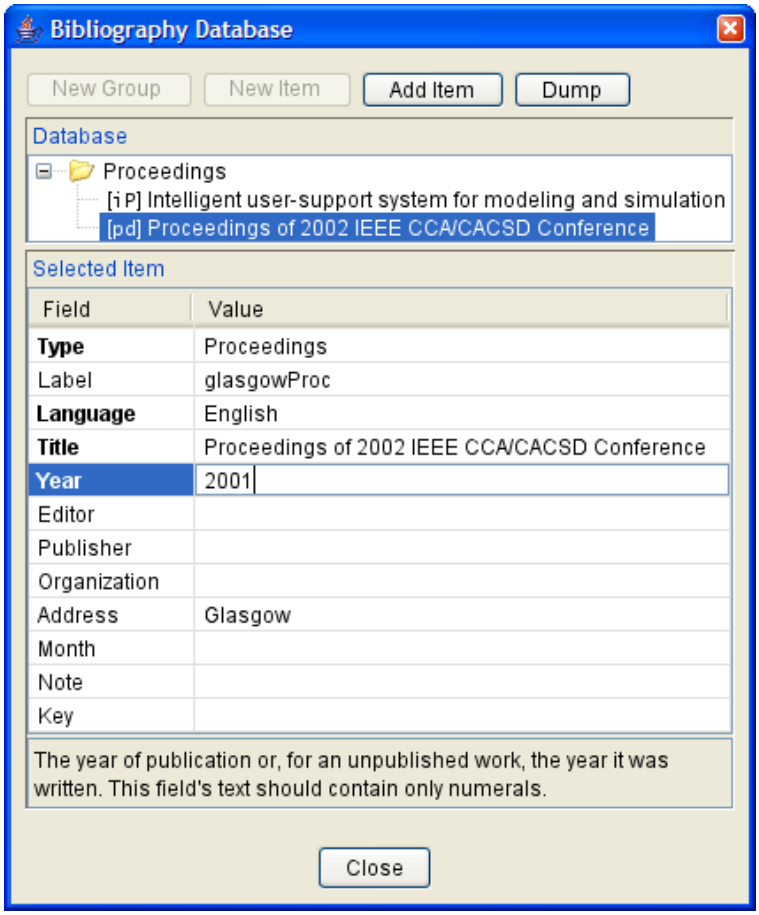


Figure 6.5 The Bibliography Database Window

new record to the selected group, press the New Item button. When you select a record in the list, the list in the bottom part of the window is populated with details of the selected record.

Firstly, you should select the type of the bibliographical reference. The possible types of bibliographical references are summarized in Table 6.1. When you select a type, a list below the type box is filled with fields corresponding to the selected type. Fields that are printed in bold are mandatory; if they are omitted, the field name is printed in red color to draw your attention. Fields that are not printed in bold are optional and may be omitted. List of all fields is summarized in table Table 6.2. The bold items correspond to mandatory fields.

Each record must have explicitly specified language. The language of the record should always correspond to the language of the publication. If you need to use a record in language different from the publication language, you may translate some fields to the required language. For instance, if you want to cite a Czech book from an English publication, you should create the bibliographical record of the book in the Czech language, and then translate some fields (e.g. the title) to English. The framework then uses the English title if you reference the record from an English context.

The Author and Editor fields may have multiple values, if the publication has more than one author or editor. If you want to provide more than one author or editor, you may either click the Author or Editor field and drag down, or just press comma (‘,’) while editing the field. Extra fields may be deleted by selecting appropriate row(s) in the table, and pressing `Delete`. You should not put more than one name to a single field!

Table 6.1 Types of Bibliographical References

Type	Description
[iP] In Proceedings	An article in the proceedings of a conference.
[pd] Proceedings	The proceedings of a conference.
[bk] Book	A book with an explicit publisher.
[iB] In Book	A part of a book, which may be a chapter and/or a range of pages.
[mt] Master's Thesis	A Master's thesis.
[pt] PhD Thesis	A PhD thesis.
[iC] In Collection	A part of a book with its own title.
[mn] Manual	Technical documentation.
[tr] Technical Report	A report published by a school or other institution, usually numbered within a series.
[bl] Booklet	A work that is printed and bound, but without a named publisher or sponsoring institution.
[jn] Journal	A journal or magazine series, without particular issue.
[ji] Journal Issue	A journal or magazine issue.
[at] Article	An article from a journal or magazine.
[up] Unpublished	A document with an author and title, but not formally published.
[ms] Misc	Use this type when nothing else seems appropriate.
[us] Unstructured	A citation with no structure, defined by a markup text. Please do not use. Provided for compatibility with existing unstructured citations.

Information about single reference may be distributed into more than one record. For example, information about an article published in conference proceedings is distributed into two records: In Proceedings, describing details of the article (title, authors, etc.), and Proceedings, describing the proceedings itself (editors, publisher, year of publication). In BibTeX, this information is mixed into a single record. If you create a new In Proceedings record, for instance, the window provides rows for each value of the record, the last row being value named ‘In’. If you click the value, two options show up. New..., which creates new Proceedings record and associates it with In Proceedings record, and Browse..., which lets you browse for existing Proceedings records. When you choose any of the two options, the fields from the selected record are appended to the list, see Figure 6.6.

Table 6.2 Fields of Bibliographical Records

Name	Description	Applies To
Address	Publisher's address. For major publishing houses, just the city is given. For small publishers, you can help the reader by giving the complete address.	Proceedings, Book, In Book, Master's Thesis, PhD Thesis, Manual, Technical Report, Booklet, Unpublished
Author	The name of the author.	**In Proceedings**, **Book**, **In Book**, **Master's Thesis**, **PhD Thesis**, **In Collection**, Manual, **Technical Report**, Booklet, **Article**, **Unpublished**, Misc
Chapter	A chapter number.	**In Book**
Edition	The edition of a book - for example, "second".	Book, In Book, Manual
Editor	Name of editor. If there are also "author" field(s), then the "editor" field gives the editor of the book or collection in which the reference appears.	Proceedings, **Book**
How Published	How something strange has been published.	Booklet, Misc
Institution	The institution that published the work.	**Technical Report**
Key	Used for alphabetizing and creating a label when the "author" and "editor" fields are missing. This field should not be confused with the Label field.	*all*
Language	Language of the publication.	***all***
Month	The month in which the work was published or, for an unpublished work, in which it was written.	Proceedings, Book, In Book, Master's Thesis, PhD Thesis, Manual, Technical Report, Booklet, Journal Issue, Misc
Note	Any additional information that can help the reader.	*all*
Number	The number of a journal, magazine, or technical report. An issue of a journal or magazine is usually identified by its volume and number; the organization that issues a technical report usually gives it a number.	Technical Report, Journal Issue
Organization	The organization sponsoring a conference.	Proceedings, Manual
Pages	A page number or range of numbers such as "42--111"; you may also have several of these, separating them with commas: "7,41,73--97".	In Proceedings, **In Book**, In Collection, Article
Publisher	The publisher's name.	Proceedings, **Book**, **In Book**
School	The name of the school where a thesis was written.	**Master's Thesis**, **PhD Thesis**
Series	The name of a series or set of books. When citing an entire book, the the "title" field gives its title and an optional "series" field gives the name of a series in which the book is published.	Book, In Book
Text	Unstructured description of the citation.	**Unstructured**
Title	The work's title.	**In Proceedings**, **Proceedings**, **Book**, **In Book**, **Master's Thesis**, **PhD Thesis**, **In Collection**, **Manual**, **Technical Report**, **Booklet**, **Journal**, **Article**, **Unpublished**, Misc
Type	The type of a technical report - for example, "Research Note".	Technical Report
Volume	The volume of a journal or multivolume book work.	Book, In Book, Journal Issue
Year	The year of publication or, for an unpublished work, the year it was written. This field's text should contain only numerals.	**Proceedings**, **Book**, **In Book**, **Master's Thesis**, **PhD Thesis**, **In Collection**, Manual, **Technical Report**, Booklet, **Journal Issue**, Unpublished, Misc

Field	Value
Type	In Proceedings
Label	glasgow
Language	English
Author	Michal Ševčenko
Author	Heřman Mann
Title	Intelligent user-support system for modeling and simulation
Pages	
Note	
Key	
In	Proceedings
[pd] **Language**	English
[pd] **Title**	Proceedings of 2002 IEEE CCA/CACSD Conference
[pd] **Year**	2001
[pd] Editor	
[pd] Publisher	
[pd] Organization	
[pd] Address	Glasgow
[pd] Month	
[pd] Note	
[pd] Key	

Figure 6.6 Chaining Records of Bibliographical References

6.2.2 BibTeX Compatibility

As you might have noticed, the BookEditor bibliography management system is similar to BibTeX. BookEditor contains a facility for importing BibTeX (`.bib`) files into RichDoc bibliographical database. The facility automatically performs necessary transformations, such as splitting multi-value fields (Author and Editor) into separate fields, and splitting compound records (Article, In Collection, In Proceedings and In Book) into multiple associated records. Please note that this facility may not support all BibTeX features, namely string macros.

6.2.3 About the Bibliographical Database

The bibliographical database being edited by the Bibliography Database window is intended for personal use. It is stored in a file `~/ksmsa/bibliography/.database`, where `~` denotes the home directory, under Windows can be usually found at `c:\Documents and Settings\`*`username`*. If you want to use the database on several computers, you should copy the database file accordingly. A system of inter-personal management of bibliographical references is not yet implemented.

Please note that the binary database format may not be compatible across different versions of the RichDoc framework. If you want to upgrade the framework, you should first dump the database to XML file, and import it from that file after upgrading.

6.2.4 Referencing Bibliography from the Document

When the database is prepared, you may create a list of references to be associated with particular document. To insert references to a list of document references, select them in the Bibliography References window, and press the Add Item button. The Bibliography Database window closes, and the selected references are added to the list of document references, see Figure 6.7.

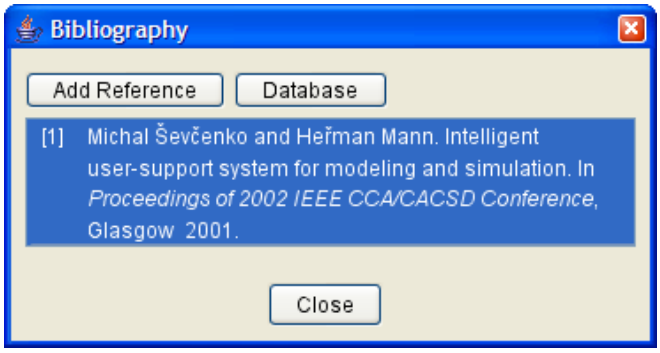


Figure 6.7 List of References Associated with a Document

To insert the reference, select it in the list, and press the Add Reference button, or just double-click the item in the list. A text containing the ordinal number of the reference, such as "[10]", is inserted into document. The text is permanently associated with the reference, and the number is updated automatically when necessary.

Chapter 7
Version Management and Localization

This chapter discusses how the RichDoc Framework supports version management of a document, in Section 7.1. Section 7.2 describes how the RichDoc Framework supports the management of localized versions of a master document.

7.1 Version Management

Under Construction

7.2 Managing of Localized Documents

Under Construction

Often you need to write a document, and provide localized versions of that document. It is also common that the master document is updated several times in its lifetime, and you would like to update the localized subversions accordingly. The RichDoc Framework's Localized Versions Management (LVM) greatly facilitates these tasks.

When you finish authoring of certain version of your document, and you want to create a localized version of it, use Tools → Create Localized Document command. A new document is created, which is initially a mere copy of the master document. When you open the newly created document, all paragraphs are displayed with blue background, indicating that they need translation. When you place text cursor into that paragraph, a localization window shows up. The localization window displays the difference between two master versions that needs to be accommodated into the localized document, see Figure 7.1*a*.

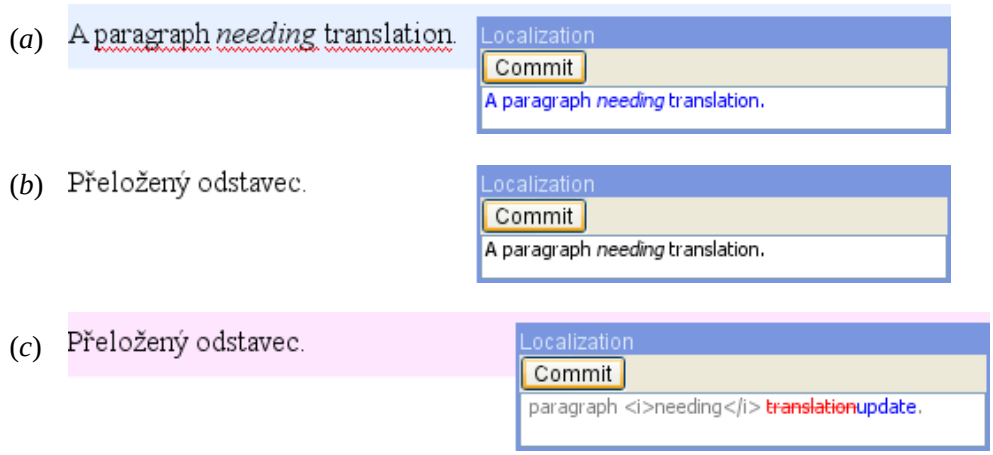


Figure 7.1 States of Localization Paragraphs: (*a*) Created, (*b*) Up-to-date, (*c*) Modified

Initially, the whole master paragraph is displayed in blue color, indicating that new material has been added and needs to be translated. When you finish your translation, press the Commit button to confirm that the translation corresponds to the current version of the master document. The correspondence is indicated by normal white background of the paragraph and normal black text color in localization window, see Figure 7.1*b*. When you later modify the master document and update the localized document, all paragraphs in the localized document needing revision are displayed with purple

background. If you put the text cursor into that paragraph, the localization window shows the change that has been done in the master since last translation, see Figure 7.1*c*. When you finish the revision of the translation, press the Commit button to confirm that the paragraph corresponds to the current version of the master again.

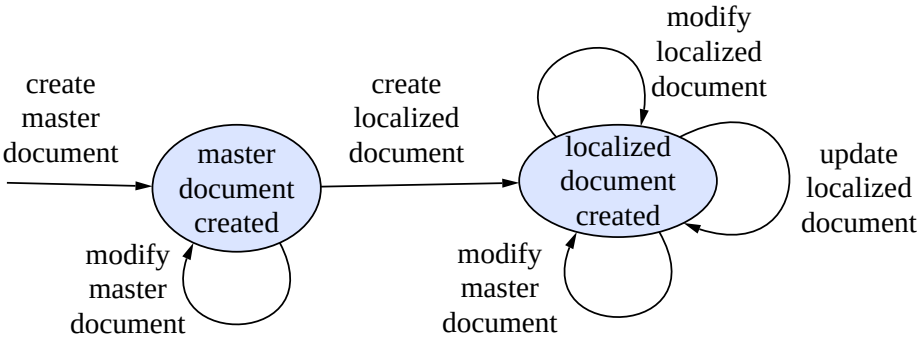


Figure 7.2 Localized Document Life Cycle

The life cycle of the documents is summarized in Figure 7.2. First, you must create the master document, and develop it for some time. When you create the localized document, you have two independent documents that can be modified independently, perhaps by different persons. When you finish some version of the master document, you may update the localized document by invoking the Create Localized Slave again, and then revise the localized document by modifying localized paragraphs that have been modified in the master document. Of course, you may create and manage many localized documents for different languages.

Note that the Editor supports only the master-slave scenario. That is, if you want to amend a group of localized documents, you should first amend the master version, and then update the localized versions accordingly. You cannot, for example, develop two language versions of a single document in parallel, for example write chapters 1–5 in English and 6–10 in Czech, and later on finish the missing chapters in the other language. You must also decide at the very beginning which language would play the master role.

Note that you may use the Create Localized Document command only once to generate an initial version of the localized document, and then you may change its structure considerably w.r.t the master document. However, if you want to use the automated synchronization system, note that the master and the slave documents must have identical structure, and may only differ in the paragraph texts. It is not possible, for example, to have a section with two paragraphs in the master document, which corresponds to three-paragraph section in the translated document. If you make any structural changes in the translated document, they will get lost upon next synchronization with the master document. This implies that if you delete any material from the master document, the corresponding translated material is deleted from the localized document as well. It is therefore recommended to backup your localized document before it is updated with the master.

7.2.1 Localizing the Index

If your document contains an index, you need to translate it as well. Translating index is quite similar to translating normal document content. If you open the index editor in a localized document, index entries are marked with the same colors as in the document, see Figure 7.3.

Up-to-date index entries are displayed with normal, white background. Entries needing translation have blue background, and entries needing revision have purple background. When you translate or revise the index entry, press the Commit button to confirm the translation. Note that line in the document, strict one-to-one correspondence between master and localization index entries is

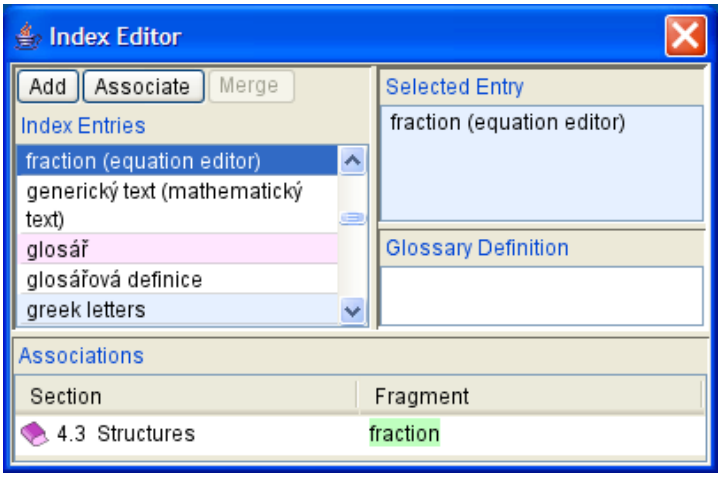


Figure 7.3 Index Editor Displaying Localized Index

required. You may add or remove index entries to the localized document, but such changes get lost when you update the slave with the master document again.

Note that when localized representation of a paragraph containing index association is added to a localized document, the associations are reconstructed in the localizable copy, and are automatically redirected to the localized index. When you translate the paragraph, you may do it in such a way that the associations are preserved. You are however free to add or remove associations, that is, index associations do not count to the requirement on structural equivalence of the master and the localized document. This also implies, that if you add new index association to a paragraph that has been localized already, that association is not reconstructed upon next localization update, and must be reconstructed manually.

Chapter 8
Building Online Courses

The BookEditor includes a module for preparing online courses compatible with the ADL™ SCORM® standard. The process of preparation and deployment is described in this chapter.

8.1 About SCORM

SCORM is an abbreviation of Sharable Content Object Reference Model. According to its authors, it is "a collection of standards and specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reusability of Web-based learning content." It enables to deliver web-based courses in standard form, so that they can be easily imported to various conforming Learning Management Systems (LMS).

The top-level structure of a course is called an *organization*. It is a hierarchical system of *activities*, which may include one or more *content objects*, see example in Figure 8.1.

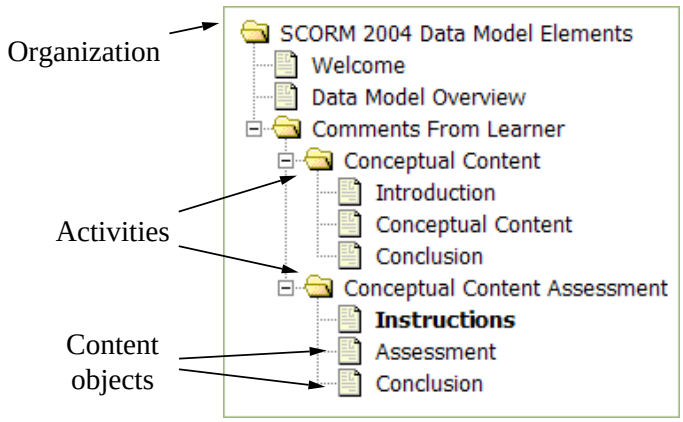


Figure 8.1 SCORM Course Structure

Content object is the actual media that is delivered to the learner. It is basically a web page or series of web pages. The learning management system provides facilities for navigation between content objects, or its individual pages.

The BookEditor supports creating documents conforming to the SCORM structure, and can export them directly to the SCORM format. However, note that not all SCORM features are supported by the BookEditor yet, namely

- Editing of metadata about the course.
- Changing navigation strategy from simple back-and-forth flow navigation.
- Communication with the LMS, such as in self-assessment tests.
- Alternative organization structures for single course.

8.2 Creating SCORM Course Document

To create new SCORM document, simply use the File→New... command from the pulldown menu, and select the SCORM Course document class. Then fill the document path and title as usual. Then

use the global document pane to create new activities, content objects, and pages. Activities are denoted with the icon. Activities may contain either pages (), representing single-page content objects, or multipage content objects (). The multipage objects then may contain pages. Nodes denoted with and icons are mere containers of other nodes, and thus have no associated documents. Page nodes denoted with icon have associated documents, but may not contain child nodes.

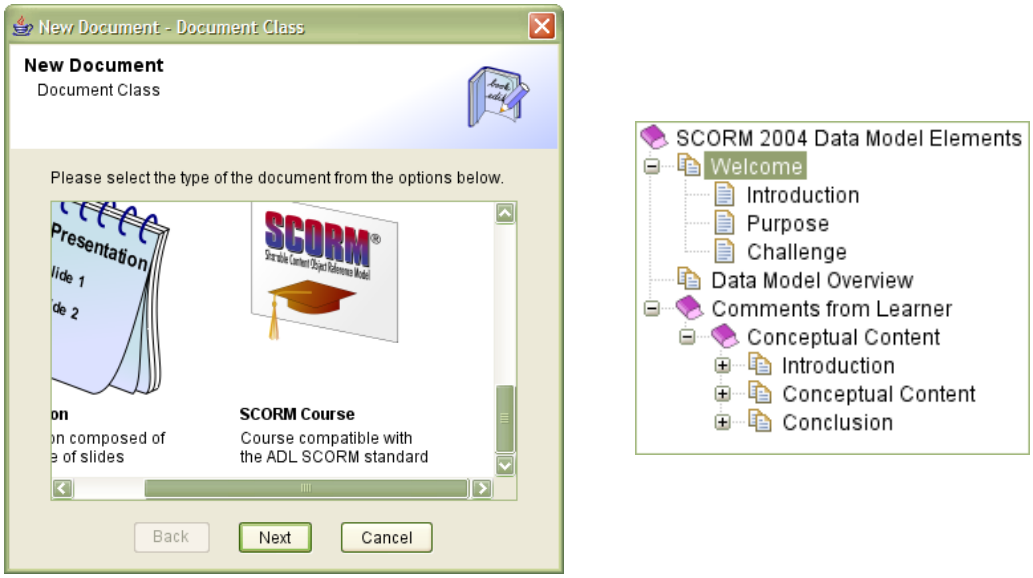


Figure 8.2 Creating and Structuring a Course Document

Chapter 9
Language Support

This chapter discusses how the RichDoc framework supports various languages. You may easily type text in any language that is supported by your local operating system, that is, your system must have installed appropriate fonts and input facilities. However, there is more than just typing text. This chapter tells you what you need to know to prepare non-English documents.

9.1 Setting the Language

As mentioned in Section 1.5, you select the primary language of a document when you are creating it. The language affects certain actions of the Editor, such as hyphenation, spell-checking, and text search. It also affects the language of text fragments automatically inserted to the document, such as prefixes of section titles. That is, when you change the document language from English to Czech, “Chapter 5” automatically turns to “Kapitola 5”, “Table of Contents” to “Obsah”, etc. Algorithms that perform text sorting are also affected: for instance, if rules for the Czech language are in operation, the text ‘ch’ is treated as single character during sorting, that appears between ‘h’ and ‘i’. The word ‘chata’ is thus considered after the word ‘hrad’ in alphabetical order, unlike English sorting rules.

The current version of the RichDoc Framework supports only two languages: English and Czech. If you need a support for another language, you may request it from the authors, or you may try to create required resources yourself, see Section 18.1. If the support for some language is not available, it is still good idea to set the document's language field to the correct value, just as information of potential readers or bookkeeping programs that may receive your document, or simply because the framework should not try to apply English-specific rules, such as morphology.

Chapter 10
Finding and Replacing Text

10.1 Finding Text

Like other document-management systems, the RichDoc framework contains a built-in facility for finding and replacing text. To find or replace text, use the Find / Replace command from the popup menu, or just press `Ctrl-F`. A Find / Replace window appears, see Figure 10.1.

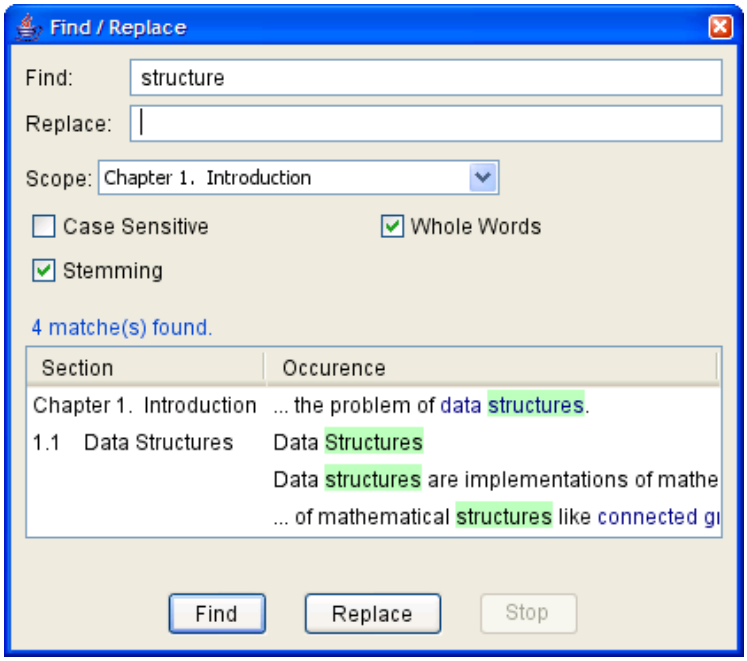


Figure 10.1 The Find / Replace Window

The two fields at the top of the window allow you to select the text to find, and eventually the replacement text. Using the Scope field, you may select whether only the active part being edited should be searched or modified, or whether the operation should be performed on the whole document.

The Case Sensitive option determines whether the capitalization matters during the search. If checked, only words that exactly match to the search phrase are found, if unchecked, words that differ only in capitalization are also selected. The Whole Words option determines whether the search phrase may be matched only with whole word (separated by spaces or punctuation), or it may be matched with a part of a word. For instance, the search phrase ‘text’ is matched with the word ‘textual’ only if the Whole Words option is unchecked. The Stemming option, when checked, attempts to match words that are morphological inflections of the search phrase. For instance, if you are looking for ‘cat on the mat’, the phrases ‘cats on the mat’, ‘cat on the mats’, etc. are also matched, if the stemming option is enabled. Note that the stemming is only available if the Whole Words option is also enabled.

When search options are specified, you may press the Find button. After a while, the Results list is populated with the search results. The results are organized into two columns: the first column displays the name of section where the search phrase has been found, and the second column displays a

fragment near the occurrence of the search phrase, with the search phrase highlighted. You may double-click a row in the list, to navigate to the corresponding place in the document.

10.2 Replacing Text

To replace the occurrences of the search phrase, specify the replacement text in the Replace field, and press the Replace button instead of the Find button. The rules for matching the text are the same as described in Section 10.1, but additionally, all matched texts are replaced with the specified replacement. The Results list this time summarizes the replacements that have been made, see example in Figure 10.2.

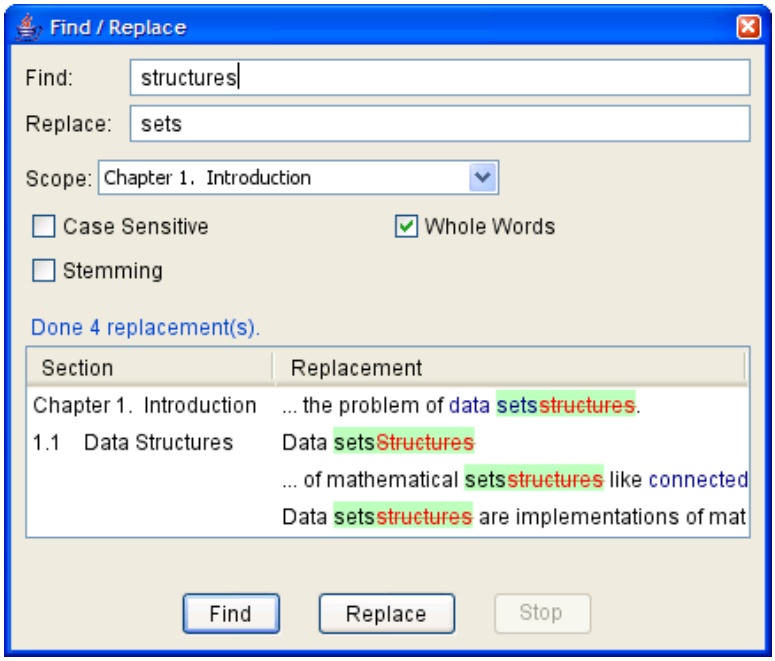


Figure 10.2 Replaced Text

As you can see in the figure, the results of the replacement operation look similar to those of the find operation. Each row shows the text that has been matched and deleted, displayed in red color and stroked-through, as well as the text that has been inserted.

Chapter 11
Importing, Exporting and Deploying Documents

The RichDoc framework can import and export documents from/to several formats. It also contains a component for presenting documents on the Web using the Apache Tomcat technology. This component may be readily used as a stand-alone web application, or may be easily integrated into an existing Tomcat application.

In particular, the RichDoc framework includes quality components for importing LaTeX (see Section 11.7) and exporting HTML (Section 11.6). This enables to use the RichDoc as a LaTeX to HTML converter, using the RichDoc format as an intermediate format.

11.1 Interface to Import/Export modules

All import/export modules have uniform interface for setting properties of the conversion process. Generally, there are two options of invoking a conversion: either from the command line, or from the user interface of the BookEditor.

Each module has different set of parameters that affect its operation. You can define a named sets of these parameters, called *profiles*. For example, if you frequently export certain document with particular settings, you can save these settings under appropriate name, and then recall them by that name whenever needed. If you invoke the conversion process from the BookEditor, such as using the command Export → LaTeX, a visual profile management tool shows up, see Figure 11.1.

Figure 11.1 Profile Management Window

At the top part of the window, there are controls for profile management. You can select active profile in the drop-down list, create new profile, save modified profile, or delete profile. Below, there are details of the selected profile. The profile consists of list of options for which you can specify custom values. The values that are in italic (text) or are blended (check boxes) indicate default values. The red 'A' letter indicates that the value is computed automatically from other value(s). For instance, the Main File value is associated with the Input Path value – it is the file title of the Input Path. If you change some value, the Save button becomes available, allowing you to save the modified settings. After pressing the OK button, the conversion process starts.

Alternatively, you may invoke the conversion from the command line using the command

`richDocIo -mode` *mode* [`-profile` *profile*] [`-interactive`] *profile_options*

where the *mode* may be any of values specified in Table 11.1. The list of options is then documented separately for each mode.

Table 11.1 Export / Import Modes

Mode	Section	Mode	Section
`exportHtml`	Section 11.2	`importHtml`	Section 11.6
`exportLatex`	Section 11.3	`importLatex`	Section 11.7
`exportPdf`	Section 11.4	`importDocbook`	Section 11.8

11.2 Exporting HTML

This module converts RichDoc documents into the Hypertext Markup Language format. The module translates the logical structure of the RichDoc document into appropriate HTML markup, including lists, tables, hyperlinks etc. Embedded figures and formulas are automatically converted into inline images linked from generated HTML files. The document may be split into separate files at specified level. Optionally navigation may be added to the generated files. A complete list of options follows.

General

Input Path (`-inputPath`)

Input path to the source RichDoc document.

Output Path (`-outputPath`)

Output path to the desired HTML output. It may be either a directory or a ZIP file.

Character Encoding (`-characterEncoding`)

Desired character encoding of the generated HTML files.

Split Level (`-splitLevel`)

Specifies at which level sections should get split into separate HTML files. Zero value specifies the level of chapters, value 1 corresponds to the level of sections, and so on.

Condensed Code (`-condensedCode`)

Specifies whether the generated HTML should omit any whitespace to reduce size.

Style Sheet

Style Sheet Path (`-styleSheetPath`)

Path to the style sheet file. The file is put into the generated directory or ZIP file, and generated HTML files are linked to the style sheet. If not specified, default style sheet is used.

Style Sheet Local Path (`-styleSheetLocalPath`)

Using this option you may specify different local path to the generated stylesheet.

Navigation

Create Top Navigation (`-createTopNavigation`)
Add a banner to the top of each generated HTML file with links to previous, next and up sections.

Create Bottom Navigation (`-createBottomNavigation`)
Add a banner to the bottom of each generated HTML file with links to previous, next and up sections.

Create List Of Child Links (`-createListOfChildLinks`)
Add links to immediate child sections to the end of each HTML pages.

Frame Set

Create Table Of Contents Frame (`-createTableOfContentsFrame`)
Create a series of HTML files representing expandable Table of Contents for the document. The generated `index.html` file would correspond to a HTML frameset containing the Table of Contents on the left and the main document body on the right.

contentFrameName (`-contentFrameName`)
Name of HTML frame containing the document body.

Decoration

Bottom Line (`-bottomLine`)
Specify text that should be added to the end of each generated HTML page, such as authorship or copyright information.

animation

imageMagickConvert (`-imageMagickConvert`)

Microsoft Html Help

Create Chm File (`-createChmFile`)
Compile generated HTML files into Microsoft HTML Help format.

Chm Output Path (`-chmOutputPath`)

Hhw Title Page (`-hhwTitlePage`)

Hhw Executable (`-hhwExecutable`)

11.3 Exporting LaTeX

Sometimes you may want to export RichDoc document into the LaTeX typesetting format. For instance, you have prepared an article using the BookEditor, and you want to send it to a publisher that requires special visual style of articles, and provides LaTeX-compatible style file for that reason. You may tune BookEditor's style sheets to match the visual appearance of your document to the required style, but the easiest way is just to export the document to LaTeX, add the publisher's LaTeX style, and use LaTeX to generate the final form.

This export module is still under construction, all of the problems are not solved yet, but in general, most features of the RichDoc framework can be easily converted to LaTeX. A list of issues that deserve particular attention is listed below.

Support for non-English languages

The language of the document is used to generate the reference to the LaTeX `babel` package. This ensures that titles of appropriate language are used, and that correct hyphenation patterns are activated. Regarding non-English characters, two options are possible. For accented characters (such as Č or ü), it is possible to convert them to latex macros, such as `\v{C}` and `\:{u}`. The second option is to save the document in some eight-bit encoding (such as iso-8859-1 for Western European languages, or iso-8859-2 for Central European Languages). The exported document then uses the `inputenc` package to specify the encoding.

Support for equations

Most equations should be exported to LaTeX without major problems, but some of the more exotic features may not be supported.

Support for 2D drawing

The embedded pictures are converted to the Encapsulated Postscript format, and if you have installed the GhostScript package, they may be optionally converted to the PDF format (you'll need this if you want to use PDFLaTeX to process your LaTeX document.) There is also a problem with non-English characters, because PostScript format does not have any uniform support for handling non-English characters. In this case, the export module converts words containing non-English characters to curves. This approach usually yields acceptable results.

Support for Tables

Converting complex tables to LaTeX may cause major problems, as the LaTeX table layout algorithm has quite limited capabilities. It cannot, for example, automatically determine appropriate width of columns containing long paragraphs. In fact, if you want to have a word-wrapped paragraph in a table, you must set the width of the table column to some fixed value.

Support for Index and Bibliography

Index and bibliography is automatically converted to LaTeX conventions. Moreover, the finishing phase, if enabled, automatically calls `bibtex` and `mkindex` programs to make index and bibliography correct. LaTeX-style glossary is not yet supported.

A complete list of options follows.

General

Input Path (`-inputPath`)

Input path to the source RichDoc document.

Output Path (`-outputPath`)

Output path to the desired LaTeX output. It may be either a directory or a ZIP file.

Main File (`-mainFile`)

Name of the file corresponding to the LaTeX document root, i.e. file to be processed with LaTeX.

Character Encoding (`-characterEncoding`)

Specifies character encoding to be used for non-ASCII characters. If "LaTeX" encoding is selected, non-English characters are escaped using common TeX/LaTeX convention, e.g. ü (u with umlaut) is escaped as `\:{u}`. If other encoding is selected, characters are encoded into single bytes, and an appropriate command is added to the document preamble.

Local Path (`-localPath`)

Specifies a path offset from the root file to other files. This path is used to generate inclusion commands, such as `\include` or `\includegraphics`.

Process Embedded Images (`-processDrawings`)

Specifies whether Encapsulated Postscript files should be generated for embedded 2D pictures. This option may be useful for temporarily disabling picture generation, if a large document need to be converted several times and picture generation slows down the conversion too much.

Process Inline Bitmaps (`-processInlineBitmaps`)

Specifies whether Encapsulated Postscript files should be generated for embedded inline bitmaps, see also Process Embedded Images option.

Generate PDF For Embedded Images (`-generatePdfForEmbedded`)

Generates embedded images in the PDF format besides the Encapsulated Postscript format. This is needed if you want to process the generated LaTeX document with PDFLaTeX to generate a PDF file.

Options

Font Size (`-fontSize`)

Specifies the font size of the target document.

Sloppy (`-sloppy`)

Specifies whether LaTeX `\sloppy` mode should be turned on. This mode ensures that LaTeX breaks paragraphs even if the breaking creates very large spaces between words. When disabled, problematic word is not put on the next line, but rather exceeds the printable area.

Max Width (`-maxWidth`)

Specifies the maximum width of tables and figures, in pts (1pt = 0.35mm or 1/72"). If a figure or table exceeds the maximum width, it is automatically scaled down to fit into the width.

Max Height (`-maxHeight`)

Specifies the maximum height of tables and figures, in pts (1pt = 0.35mm or 1/72"). If a figure or table exceeds the maximum height, it is automatically scaled down to fit into the height.

Finishing

Finishing Mode (`-finishingMode`)

Specifies whether the generated LaTeX document should be processed with LaTeX to generate a DVI file, with LaTeX + DVIPS to generate a PostScript file, or PDFLaTeX to generate a PDF file.

Finish Dir (`-finishDir`)

Specifies the directory where the LaTeX processor is invoked. Changing the directory may be useful if you want to use main file different from the automatically generated main file.

Finish Main File (`-finishMainFile`)

Specifies the directory for which the LaTeX processor is invoked.

11.4 Exporting PDF

This module can be used to convert a RichDoc document into a PDF format. The result is similar as if the document is printed to a regular printer. Optionally, hyperlinks in the document are converted to PDF hyperlinks, and/or an interactive table of contents (called bookmarks in PDF terminology) is added to the PDF document.

General

Input Path (`-inputPath`)

Input path to the source RichDoc document.

Output Path (`-outputPath`)

Output path to the desired PDF output.

Options

orientation (`-orientation`)

paperSize (`-paperSize`)

Hyperlinks (`-hyperlinks`)

Whether RichDoc hyperlinks should be converted to PDF hyperlinks.

Bookmarks (`-bookmarks`)

Whether PDF bookmarks, i.e. interactive table of contents, should be added to the PDF document.

Incremental (`-incremental`)

11.5 Exporting SCORM

This module converts RichDoc document into a file conforming to the ADL SCORM standard.

11.6 Importing HTML

This module converts HTML documents into a RichDoc document.

General

Input Path (`-inputPath`)

Specifies the path to the HTML files to be imported. The path may be either a directory containing HTML files, a ZIP file containing HTML files, or output from the KSMSA Web Crawler.

Output Path (`-outputPath`)

Specifies the path to the RichDoc document to be generated.

Language (`-language`)

Specifies the primary language of the generated RichDoc document.

Character Encoding (`-characterEncoding`)

Specifies the character encoding of the input HTML documents, if they contain encoded non-ASCII characters. Note that known HTML entities corresponding to non-ASCII characters (such as `&uuml;`) are automatically converted to single UNICODE characters. If encoding is specified in the Web Crawler database (i.e. HTML document was downloaded from the web and the web server returned encoding for it in the HTTP response header), that value is used instead. Note that encoding value from the `http-equiv` header tag is not used.

Document Class (`-documentClass`)

Specifies the desired document class of the target document.

File Filter

Include File (`-includeFile`)

Specifies a list of HTML files that should be imported. You may use wildcards, such as `*.html` for all files with html extension in the main directory, or `**/*.html` for all HTML files in all directories. If this list is empty, all files are included.

Exclude File (`-excludeFile`)

Specifies list of files that should not be imported. You may use wildcards.

Content Filter

Content Start (`-contentStart`)

Specifies a regular expression defining a position in a file where the conversion should start, such as `<body>`. This is useful to omit e.g. navigation code or banners. If not specified, or the start sequence is not found in the file, conversion starts from the beginning of the file. Otherwise, the conversion starts from the first occurrence of the start sequence.

Include Content Start (`-includeContentStart`)

Specifies whether the start sequence should be converted.

Content End (`-contentEnd`)

Specifies a regular expression defining a position in a file where the conversion should end, such as `</body>`. If not specified, or the end sequence is not found in the file, conversion continues to the end of the file. Otherwise, the conversion ends at the last occurrence of the end sequence.

Include Content End (`-includeContentEnd`)

Specifies whether the end sequence should be converted.

Exclude Fragment (`-excludeFragment`)

Specifies a regular expression of fragments that should be excluded from the conversion process.

Replace Fragment (`-replaceFragment`)

Specifies a regular expression-replacement pairs that should be used to pre-process the input HTML file. All occurrences of the regular expressions are replaced with the corresponding replacement.

Ignore Tags (`-ignoreTags`)

List of tag names (without < and > delimiters) that should be ignored. Note that only the tag delimiters are ignored, the content of the tag is processed normally. If you want to ignore entire tag, use Exclude Fragment option with regular expression `<tag[^>]>.*</tag>`.

Output Filtered (`-outputFiltered`)

Specifies a path to a ZIP file that will contain HTML files that were actually used for conversion, when filtering and replacements were applied. Since any problems found during conversion are reported w.r.t. the filtered content, the filtered files may be useful to

Content Modification

Number Sections (`-numberSections`)

Whether the sections generated from the `<h*>` tags should be numbered by the RichDoc framework.

Remove Original Section Numbers (`-removeOriginalSectionNumbers`)

Whether original section numbers from titles should be removed. This option should be checked if you also checked the Number Sections option.

Detect Unnumbered Sections (`-detectUnnumberedSections`)

This option marks sections that didn't contain number in original HTML source as unnumbered. Otherwise, they are automatically numbered unless you disabled the Number Sections option.

Misc

Print Font Size (`-printFontSize`)

Specifies the font size used when the generated RichDoc document is printed. If not specified the default font size is used.

11.7 Importing LaTeX

Documentation not yet available.

General

Input Path (`-inputPath`)

Specifies the path to the main LaTeX file to import.

Output Path (`-outputPath`)

Specifies the path to the RichDoc document to be generated.

Language (`-language`)

Specifies the primary language of the generated RichDoc document.

Character Encoding (`-characterEncoding`)

Specifies the character encoding of the input LaTeX document, if it contains encoded non-ASCII characters. Note that LaTex-escaped non-ASCII characters (such as `\:{u}`) are automatically converted to single UNICODE characters.

11.8 Importing DocBook

This module imports documents obeying the DocBook [1] XML markup, see http://www.docbook.org.

General

Input Path (`-inputPath`)

Specifies the path to the XML files with DocBook markup to be imported. The path may be either a directory containing XML files, a ZIP file containing HTML files, or output from the KSMSA Web Crawler.

Main File (`-mainFile`)

The main XML file corresponding to the root of the DocBook document.

Output Path (`-outputPath`)

Specifies the path to the RichDoc document to be generated.

Language (`-language`)

Specifies the primary language of the generated RichDoc document.

Character Encoding (`-characterEncoding`)

Specifies the character encoding of the input HTML documents, if they contain encoded non-ASCII characters.

Document Class (`-documentClass`)

Specifies the document class of the generated RichDoc document.

Create TOC (`-createToc`)

Whether Table of Contents should be added to the generated RichDoc document.

Create Short TOC (`-createShortToc`)

Whether short Table of Contents (chapters only) should be added to the generated RichDoc document.

File Filter

Include File (`-includeFile`)

Specifies a list of files or wildcards to be imported.

Exclude File (`-excludeFile`)

Specifies a list of files or wildcards to be excluded from the conversion process.

Misc

Print Font Size (`-printFontSize`)

Specifies the font size used when the generated RichDoc document is printed. If not specified the default font size is used.

11.9 Deploying Documents

Documentation not yet available.

Chapter 12
Printing

Chapter 13
The ScratchPad Application

So far, we discussed the primary application of the RichDoc framework, the BookEditor. The RichDoc framework also contains the ScratchPad application, intended for writing and managing personal notes. The user interface of the ScratchPad is quite similar to that of BookEditor, see Figure 13.1.

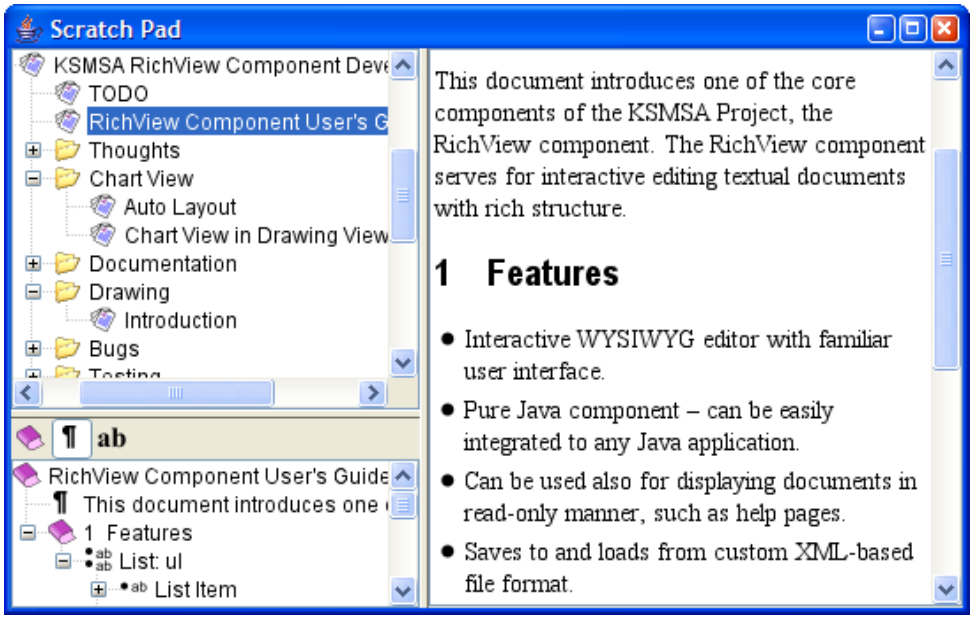


Figure 13.1 The ScratchPad Application

As you can see in the figure, the user interface is the same, except the upper-left pane, which does not display the structure of single document, but the structure of your personal notes.

13.1 Managing the System of Notes

The Notes Pane shows a hierarchical list of your notes. The items denoted by are groups, that contain other groups, or notes. Notes are denoted by the icon. The selected note may be denoted by the icon to indicate that it has been modified and needs saving.

If a group is denoted by the icon, it means that the group itself has an associated note. A group may also be denoted by the icon, which indicates that when it is selected, its children are collected to displayed in the editor pane as a read-only document.

Chapter 14
Troubleshooting

The RichDoc framework is an experimental project that is under rapid development. This means that the framework may contain bugs, many features are not finished, and some features and file formats may be changed without notice. If you still decide to use it on a production level, you should be very careful, save and backup your files regularly, and be prepared to face frequent problems. Namely, the framework has the following limitations:

- There is no undo/redo support. What you do to the document cannot be easily reversed. You should thus save your document frequently, so that you can revert to previously saved version if your document gets damaged, either due to your mistake, or due to the failure of the framework. If your document gets damaged during saving, you may attempt to recover it from emergency backup file, as described in Section 14.2.

14.1 Platform-specific problems

Although the RichDoc framework is implemented using technology supporting all major operating systems, there are some known problems with certain platforms. This section discusses some of them.

- On certain systems, it is difficult to set up input facilities to support national (non-ASCII) characters. For this purpose, the BookEditor application contains a user-level feature that maps keys on a keyboard to other, non-English letters. With this feature, you may type text in non-English language as if the underlying system supported it. The character map is currently provided only for the Czech language. You can switch between English and Czech keyboard by clicking the language button in the bottom-right corner of the main frame.
- The framework relies on the underlying window system to send notification when the main application window is deactivated. For instance, the main toolbar should be hidden upon deactivation. However, some systems fail to send such notification. For instance, if you use the cygwin X-server under MS Windows to display a RichDoc application running on UNIX system, deactivating the RichDoc application displayed in X-windows frame, by activating other window that itself is not an X-windows, does not cause the deactivated window to be notified. This causes the main toolbar window to remain visible, obscuring the newly activated window. This problem can be fixed only by configuring the RichDoc framework to display the main toolbar as a *lightweight* window. In lightweight mode, the toolbar is not a real window, but just a rectangular area painted into the host window, namely the RichDoc main frame. This solves the problem, but prevents the main toolbar to be dragged off the main frame of the RichDoc application.
- Some systems, namely cygwin X-server running under MS Windows, may not support drag & drop operations. There is again a possibility of enabling a *lightweight* drag & drop, which handles mouse events and simulates drag & drop operation on application level. This effectively enables the drag & drop, albeit with limited functionality.

14.2 File Backup and Recovery

Each time you save a document, the book editor creates an incremental backup of the file before it is rewritten with the new version. The word *incremental* refers to the fact that only files

that are going to be modified are backed up. This keeps the size of backup files small. The backup files are ZIP files containing original content of modified ZIP files. The emergency file has the path *$TEMP*/ksmsa/richDoc/backup/*$FILE*-*$ID*/*$DATE*.zip, where *$TEMP* is the path to your system's temporary directory (usually c:\Documents and Settings*$USERNAME*\Local Settings\Temp under Windows), *$FILE* is the short name of the file being saved, *$ID* is the global identifier of the file main section, and *$DATE* is the date of backup.

Note that this feature is intended for emergency purposes only, you should not rely on it as a kind of version management system. A better, more user friendly version management system is under construction.

Chapter 15
Acknowledgments

The RichDoc framework is an open-source project, which draws resources and libraries from other open-source systems. In this chapter, we would like to thank to all contributors for their willing to share their resources. Table 15.1 summarizes projects and resources that have been adopted.

Table 15.1 Resources used in the RichDoc Framework

Consortium	Project	Comment & Copyright
Apache Software Foundation	Apache XML Graphics	PDF generation library. Copyright (C) 1999-2003 The Apache Software Foundation. This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
	Apache Ant	Packaging and compressing utilities (TAR, bzip2).
TeX Users Group	TeX	Math fonts
		English hyphenation patterns (c) Frank Liang.
		Czech hyphenation patterns (c) Pavel Ševeček, Lingea s.r.o.
		Babel Package, Copyright 1993–2005 Johannes L. Braams & Contributors, under the LaTeX Project Public License.
ispell contributors	ispell	English spell-checking database (c) by Geoff Kuenning and other unpaid contributors.
		Czech spell-checking database (c) 2001 by Petr Kolář. Contributors to the Czech dictionary: Tomáš Čermák, Petr Prenghy, Hanuš Adler, Petr Kolář.

Part II
Expert's Guide

Chapter 16
Introduction to the Data Model

Chapter 17
The RichDoc Print Format

The RichDoc contains a facility for printing documents into intermediate format, that can be used later for print preview or sending to a real printer. The intermediate files are useful to avoid repetitive print layouts, which may be very lengthy for big documents.

The RichDoc Print format has similar role like PostScript and PDF formats, and also has similar architecture. There are several reasons why we decided to create a new format and not to reuse existing standard formats. The main reason is that there is no cross-platform, patent free, Java™ compatible library for writing and reading PostScript, PDF or similar standard format. Implementing such library would be difficult, as the formats are very complex, and many of their features could not be even utilized by our framework. Another reason is that these formats do not have good support for Unicode, while Java™ Printing APIs fully support Unicode.

Moreover, our format has one more extra feature: it supports *incremental printing*, which allows modification of existing print file by reprinting only those pages that have been modified since last printing. This feature is useful for keeping print files of large documents up-to-date, if the documents are often modified.

The RichDoc Print format is merely a serialization of commands issued through the interface of the `java.awt.Graphics2D` class. That is, the "printer" component implements the `Graphics2D` interface by serializing the commands and storing them to a binary file. The "playback" component reads the serialized commands, and sends them to supplied object compatible with `Graphics2D`, which may be either real printer or a print-preview component.

17.1 The Overall Structure of a Print File

The print file is actually a ZIP file, consisting of page definition files, embedded bitmaps, document source title, and an index to support the incremental printing function. Each page definition file obeys a format described in Section 17.2.

17.2 The Page Description Format

This section defines the format of a single file within the overall print file, which is a definition of objects on a single page. The page-definition format is binary, i.e. it is a stream of bytes. Therefore, we need to define data types that are used to define object properties, and how they are serialized into streams of bytes. The data types are defined in Section 17.2.1.

Table 17.1 defines the top-level structure of the page definition file. First, the description of the printing media is written, see Section 17.2.2. Then follows a list of printing instructions, as they were recorded by the printer component. Each instruction starts with a code byte, which defines the type of the instruction, see Section 17.2.3. The sequence of instructions is terminated by a `END_OF_FILE` value is encountered.

17.2.1 Data Types

The data types are listed in Table 17.2. All integer data types are signed using the common, i.e. "two's complement", encoding of negative numbers (e.g. unsigned 0xFF represents −1 byte value, 0xFE −2 value, etc.) All integers are stored in *big endian*, that is, more significant values are stored first.

Table 17.1 Overall Page Definition Structure

Field	Description
`pageFormat`	description of the printing media, see Table 17.8
`byte`	#1 instruction code
instruction	#1 instruction data, see Table 17.9
`byte`	#2 instruction code, etc., until `END_OF_FILE` byte encountered

Real numbers are encoded by first converting them to integers using `java.lang.Double.doubleToLongBits()` or `java.lang.Float.floatToIntBits()`, and then encoding them in big endian as integers. This encoding corresponds to the IEEE 754 standard for encoding numbers.

The string value is encoded the same way as if saved using `java.io.RandomAccessFile.writeUTF()`. That is, the length of the string is first encoded as a 16-bit integer, and then the string characters are appended, each encoded using the UTF-8 encoding.

Table 17.2 Data Types

Name	Description
`byte`	8-bit signed integer
`double`	64-bit real number
`float`	32-bit real number
`short`	16-bit signed integer
`int`	32-bit signed integer
`long`	64-bit signed integer
`boolean`	8-bit boolean
`string`	UTF-8 encoded string, with leading `short` for the string length
`shape`	serialization of `java.awt.Shape`, see Table 17.3
`stroke`	serialization of `java.awt.BasicStroke`, see Table 17.4
`transform`	serialization of `java.awt.geom.AffineTransform`, see Table 17.5

Besides primitive data types, there are also more complex types needed to encode some parameters of the `Graphics2D` interface. The serialization of `java.awt.Shape` is described in Table 17.3.

Table 17.4 Serialization of `java.awt.BasicStroke`

Field	Description
`float`	line-width
`byte`	end-cap: 0 – butt, 1 – round, 2 – square
`byte`	join-type: 0 – miter, 1 – round, 2 – bevel
`float`	miter-limit
`short`	dash length
`float`	dash #1
`float`	dash #2 etc.
`float`	dash phase (only if dash length > 0)

Table 17.3 Serialization of `java.awt.Shape`

Field	Description	
`byte`	winding rule, 0 – even-odd, 1 – non-zero	
`byte`	type of #1 segment, 0 – move-to, 1 – line-to, 2 – quad-to, 3 – cubic-to, 4 – close	
data for segment #1		
`byte`	type of segment #2, etc, until -1 is encountered	
move-to, line-to	`float`	x endpoint coordinate
	`float`	y endpoint coordinate
quad-to	`float`	x control point coordinate
	`float`	y control point coordinate
	`float`	x endpoint coordinate
	`float`	y endpoint coordinate
cubic-to	`float`	x control point #1 coordinate
	`float`	y control point #1 coordinate
	`float`	x control point #2 coordinate
	`float`	y control point #2 coordinate
	`float`	x endpoint coordinate
	`float`	y endpoint coordinate

Table 17.5 Serialization of `java.awt.geom.AffineTransform`

Field	Description
`double`	the scale-x value of the transform matrix
`double`	the shear-y value of the transform matrix
`double`	the shear-x value of the transform matrix
`double`	the scale-y value of the transform matrix
`double`	translate-x value of the transform matrix
`double`	translate-y value of the transform matrix

Table 17.6 Serialization of `java.awt.Color`

Field	Description
`byte`	the alpha value (0 = transparent, 255 = opaque)
`byte`	the red value
`byte`	the green value
`byte`	the blue value

17.2.2 Printing Media Definition

The printing media definition section of the file defines the size and orientation of the paper to which the material is printed, see Table 17.8. First, the orientation of the paper is written, then its total size,

Table 17.7 Serialization of `java.awt.Font`

Field	Description
`string`	font name
`byte`	font style (0 – plain, 1 – bold, 2 – italic, 3 – bold italic)
`float`	font size

imageable size (total size without margins), and imageable area offset from the total area. See Figure 17.1 for the meaning of fields in Table 17.8. Note that the fields describing the paper size refer to the portrait orientation of the paper. For instance, the A4 paper will always have the width of 210mm and the height of 297mm, regardless the paper orientation.

Table 17.8 `pageFormat` Field Definition, see Figure 17.1

Field	Description
`byte`	paper orientation: 0 – landscape (Windows), 1 – portrait, 2 – reverse (Macintosh) landscape
`double`	paper width
`double`	paper height
`double`	imageable x
`double`	imageable y
`double`	imageable width
`double`	imageable height

17.2.3 Printing Instructions

We call a single call to the `Graphics2D` interface a *printing instruction*. To encode the printing instruction, we must encode its type, and all its parameters. All supported printing instructions and their parameters are summarized in Table 17.9. Since the parameters closely correspond to the parameters of `Graphics2D` methods, we comment them very sparsely. See documentation of `Graphics2D` for more information.

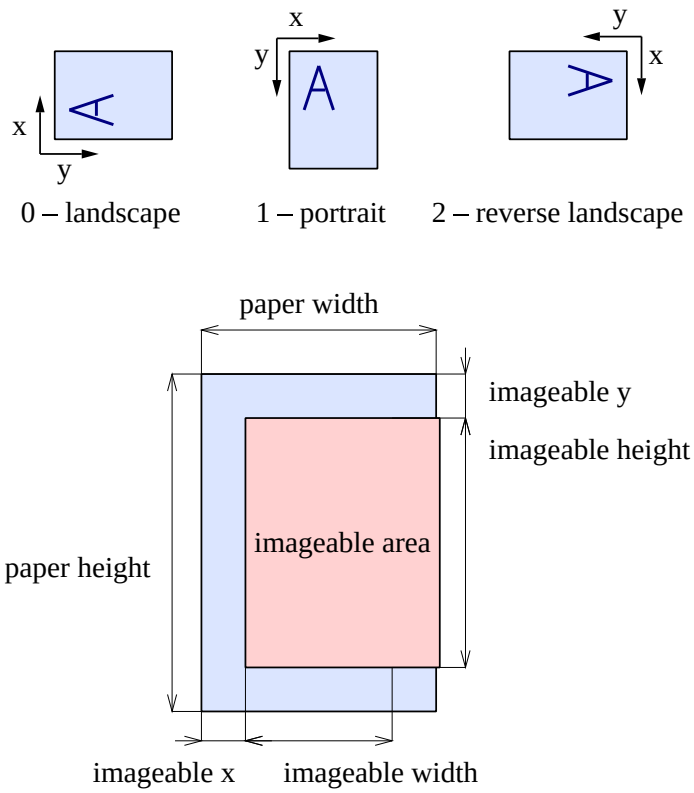


Figure 17.1 Meaning of `pageFormat` Fields

17.3 Notes on Printing to the RichDoc Print Format

In this section, we would like to mention some limitations and other technical notes regarding using the RichDoc Print interface from a Java™ application.

First, note that some features of the `java.awt.Graphics2D` may not be implemented. For example, painting using general `Paint` component is not supported. Also note that like Java Serialization format, this format is fragile, and is subject to change without notice. You should use it only for intermediate storage of print files, not for long-term storage. No support is provided for maintaining compatibility of various versions of this format. If you detect that a print file has different format version than you expect, you should consider the file invalid, and should not attempt to parse it.

Be aware that recorded printing commands may be played back in a different context that they were captured. It is therefore illegal to use any commands that set the global state of the `java.awt.Graphics2D` class, such as by calling `setTransform()` or `setClip()`. You should use their relative equivalents, i.e. `transform()` and `clip()`. It is, however, legal to save the state of the class, e.g. by calling `getTransorm`, and restore it later, e.g. by `setTransform()`. The printing component automatically converts these commands to `SAVE_TRANSFORM` and `RESTORE_TRANSFORM` instructions, respectively.

Table 17.9 `instruction` Fields

Instruction Code	Field	Description
0 = DRAW_STRING_INT	draw string on integer coordinates	
	string	the string
	int	the x coordinate
	int	the y coordinate
1 = DRAW_STRING_FLOAT	draw string on real coordinates	
	string	the string
	float	the x coordinate
	float	the y coordinate
2 = FILL_SHAPE	fills shape	
	shape	the shape to fill
3 = DRAW_SHAPE	draws shape	
	shape	the shape to draw
4 = SET_STROKE	sets the stroke	
	stroke	the stroke to set
5 = TRANSFORM	appends transform to current transform	
	transform	transform to append
6 = SAVE_TRANSFORM	saves current transform under given ID	
	short	transform ID
7 = RESTORE_TRANSFORM	restore previously saved transform	
	short	transform ID
8 = CLIP	reduces current clip area by given shape	
	shape	the shape to clip to
9 = SAVE_CLIP	saves current clip under given ID	
	short	transform ID
10 = RESTORE_CLIP	restore previously saved clip	
	short	transform ID
11 = RESET_CLIP	resets the clip to the state before playback has started	
12 = SET_COLOR	sets current color	
	color	the color to set
13 = SET_FONT	sets the current font	
	font	the font to set
14 = SET_FONT_VARIANT	sets the variant of the current font (to save space, if the font name is the same)	
	byte	font style
	float	font size
15 = DRAW_TRANSFORMED_IMAGE	draws image with transformation	
	string	the image file name
	transform	the transform to apply before drawing
16 = DRAW_IMAGE	draws image	
	string	the image file name
17 = SAVE_SHAPE	fills given shape, and saves it under given name	
	string	shape name
	double	shape x-location
	double	shape y-location
	boolean	x-mirrored
	shape	shape
18 = USE_SHAPE	fills previously saved shape	
	string	shape name
	double	shape x-location
	double	shape y-location
	boolean	x-mirrored
19 = SHAPE_SCALE	writes the scale to be used for SAVE_SHAPE and USE_SHAPE	
	double	scale
20 = END_OF_FILE	end of file mark	

Chapter 18
Contributing to the RichDoc Framework

The RichDoc framework is a public-domain project, which is provided for free to its users in the hope it will be useful. The users are encouraged to contribute to the project in any way, by reporting bugs, submitting suggestions, adding code or adding localized resources.

18.1 Contributing Localized Resources

So far, the framework fully supports the English language, and partially supports the Czech language (unfortunately the only two languages the author can speak).

18.1.1 Localizing User Interface

The RichDoc framework is delivered by means of a series of JAR (Java Archive) files, that contain the executable code, as well as localizable resources for the user interface of the framework.

The most important part, that must be localized at a minimum to reasonably support a language, is in the package `org.ksmsa.richView.model`. It contains names of text fragments automatically inserted to documents, such as “Table of Contents”, “Chapter”, “Section” etc.

18.1.2 Creating Language Packs

A *Language Pack* is a bundle of resources that is external to the executable JAR files of the framework. It contains the following resources:

Hyphenation Patterns
: Hyphenation patterns tell the RichDoc framework how to correctly hyphenate words. The format the RichDoc framework is using is similar to hyphenation files of the TeX system, and thus can be simply imported from TeX, as hyphenation files within the TeX system are usually in the Public Domain.

Spell-checking Database
: Spell-checking database allows the RichDoc framework to report spelling errors. The databases are adopted from the *ispell* project.

Stemming Rules Database
: Stemming rules define how given language inflects words. It is used by the framework during indexing and search, to improve the recall of the search system. The search system can then find words even if they appear in the document in their inflected form. The database also contains the *stop list*, i.e. list of words that are not significant for the search, which includes articles, conjunctions, auxiliary verbs, etc. Database may also contain list of exceptions, i.e. list of words and their inflections that do not obey the rules.

Chapter 19
Styles

This chapter describes in more detail the style languages that are used to customize the RichDoc framework.

19.1 Visual Styles

Visual style language defines the rules that affect the process of visualization of a RichDoc document. It is quite similar to HTML Cascading Style Sheet system. The visual styles are organized into a cascade of style files. The anatomy of a style file is shown in Figure 19.1.

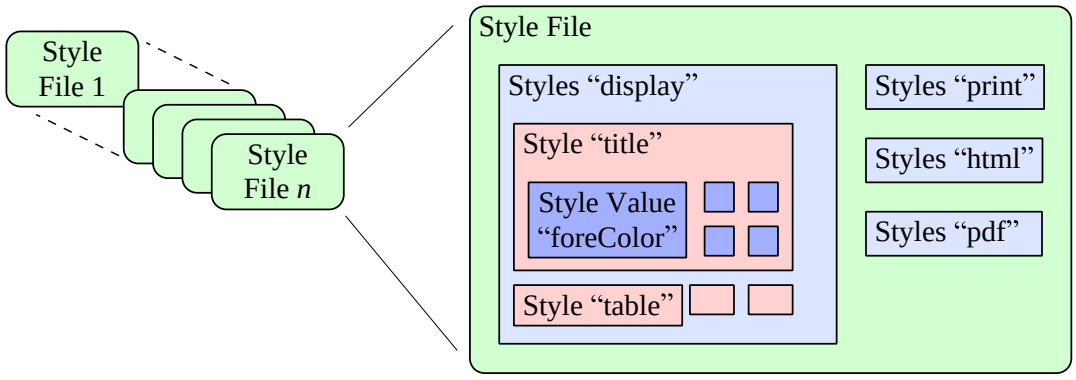


Figure 19.1 Anatomy of Style File

Each style file contains visual rules, organized into structures of several levels:

Styles Level
The Styles structure groups rules applicable for different context of rendering, such as display, printing, exporting to PDF, etc. You may also create intermediate Styles structures containing general rules and inherit these from more specific Styles structures.

Style Level
The Style structure groups rules to be applicable to one element of the document, such as paragraph, table, word etc. Style structures have names that are used to match the structures with the object, see Section 19.1.1. Like with styles structures, Style structures may inherit data from other Style structures.

Style Value Level
The Style Value represents single visual property to be applied to an object, such as foreground color, font size etc.

19.1.1 Matching and Inheriting Style Structures

The ultimate goal of the style mechanism is to assign to each document element, such as paragraph or table, a set of style values, such as foreground color or font size. When this assignment is done, we can use the style values to render the document element. Although we may do the assignment explic-

itly for each element, like in traditional word processors, we do it more systematically using external matching mechanism, see Figure 19.2.

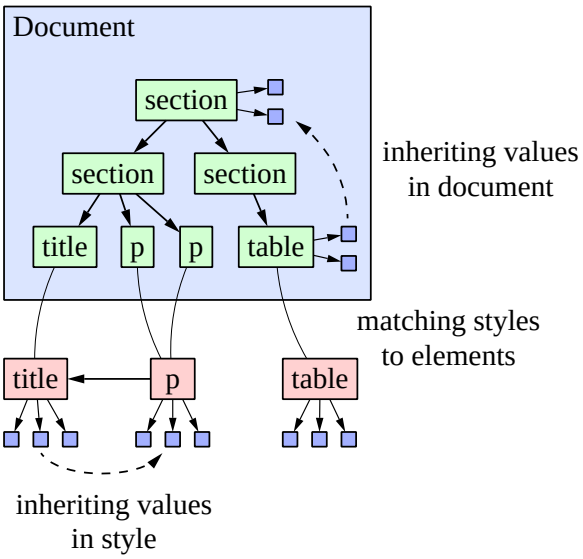


Figure 19.2 Matching Elements to Styles

First, for each document element, we find the best matching Style structure. Since Style structures are actually containers of style values, this assignment effectively assigns the document element a group of style values. If we need a style value for particular element, we first ask its associated Style structure. If the value is not found, we attempt to *inherit* it from elsewhere.

If the Style structure has a parent Style structure (such as the "title" style in our example is derived from the "p" style), we first try to inherit the value from here. This kind of inheritance we call *inheritance in style*. If this fails, we attempt to inherit the value from the containing element (such as the "table" element in our example inherits from its containing "section" elements). This process we call *inheritance in document*.

Note that not all style values may be inherited in document. For instance, the margins of elements are not inherited, because that would add to the margin for each new level of containment in the document. The colors, on the other hand, are inherited in document: if we say that a section has specific foreground color, all its children inherit that color unless they override it.

Finally, we need to explain how we match Style structures to document elements. Each document has a tag name, and optional class name. Tag names of various types of elements are described in Section 17.2. They roughly correspond to HTML tag names. From the tag name and class name, we construct *element name*, which is either `tagName` or `tagName#class`. The *element path* is dot-separated list of element names of all parent elements of given element from the document root towards the element, including the final element. For instance, the element path of the table in Figure 19.2 is `section.section.table`. A *partial element path* is element path with some parts omitted. We may omit any number of dot-separated items from the beginning of the path, and/or any number of class qualifiers. For instance, the partial element paths of the path `section#appendix.p#note` are: `p`, `p#note`, `section.p`, `section.p#note`, and `section#appendix.p`. The element path or any of element partial paths are said to *match* the element. We say that p_1 is *stronger match* than p_2 if (a) p_1 is longer than p_2 in terms of the number of the dot-separated elements, or (b), if the paths have the same length, but p_1 has class qualifier sooner than p_2. The partial paths of our example path are listed in order of their strength, from the weakest one to the strongest one.

After these complicated definitions, we come to a simple conclusion: we match document element with that style whose name has best match with the element's path. For instance, we have style

s_1 with name `section#appendix.p` and s_2 with name `section.p#note`. They both match our example element, but we match it with s_1, because it has stronger match. That is, we ignore the `note` class of the paragraph in favor of obeying the `appendix` class of the containing section.

19.2 Document Styles

Document style language defines various document classes (for books, articles, etc), and for each class defines rules how document material is named and numbered.

19.3 Bibliography Styles

Bibliography style language defines types of bibliographical references (see Table 6.1), and for each reference, how the linear textual representation, that is displayed in the Bibliography section of a document, is constructed from the database fields.

Bibliography style is defined in a XML file `style.xml`. The file has structure shown in Figure 19.3. The root element is `styles`, having `style` element for each type of publication. The `type` attribute of the `style` element corresponds to one of the publication type from Table 6.1. Its value is the English name of the publication type in Java convention, e.g. "`technicalReport`".

```
<styles>
   <style type="type">markup </style>
   ⋮
   <style type="type">markup </style>
</styles>
```

Figure 19.3 Structure of a Bibliography Style File

The `markup` contains the actual XML markup that generates the citation text. It may contain ordinary paragraph markup (see Section 17.2), plus the following special elements:

`<dbField [b="text"] [a="text"] [from="record"] />`

element that inserts one field from the record being rendered. *`dbField`* is the name of the field to be inserted, i.e. one of fields listed in Table 6.2. Again, the field name is formed from the field-'s English name in Java style, e.g. `<howPublished />`. The `b` attribute may be used to insert text (e.g. punctuation) before the field text, if the field text is not empty. Likewise, the value of the `a` attribute is conditionally inserted after the field text. The `from` attribute may be used for qualification of record associated with the record being rendered. For instance, if we render In Proceedings record, the `title` element refers to the in proceedings title. If we want to refer to the title of the containing proceedings, we should use `<title from="pd" />`.

If the field contains multiple values (such as the `author` field), all values are inserted with appropriate punctuation used to separate the values.

`<name type="type" />`

Inserts the localized name of the publication type. For instance, `<name type="phdThesis" />` inserts the phrase "PhD Thesis" in an appropriate language.

`<s value="key" />`

Inserts the localized string of any kind. The *`key`* identifies the string to be added. For instance, `<s value="and" />` inserts the conjunction "and" in an appropriate language.

For instance, the `inProceedings` type of publication is defined by markup "`<author />. <title />. <s value="inProceedings" /> <i><title from="pd" /></i>, <address from="pd" /> <month from="pd" /> <year from="pd" />.`". When interpreted, it may be translated into text "Michal Ševčenko and Heřman Mann. Intelligent user-support system for modeling and simulation. In *Proceedings of 2002 IEEE CCA/CACSD Conference*, Glasgow 2001.".

Index

Bibliography

[1] *DocBook.org*. http://www.docbook.org/.