

# The evolution and use of a personal L<sup>A</sup>T<sub>E</sub>X metapackage

Esben Sloth Andersen

esa@business.aau.dk

Version: 2007-10-04, 07:07

This document defines and describes `esameta.sty` and its variant `esametaps.sty`. They are personal L<sup>A</sup>T<sub>E</sub>X packages that largely consist of calls of other packages. These ‘metapackages’ are selected to help the development of the author’s complex book and paper projects—but some of his decisions might be of more general relevance. The use of the L<sup>A</sup>T<sub>E</sub>X system of ‘literate programming’ and the testing the compatibility of new commands and packages with the previously included parts of the metapackage should also be noticed.

## Contents

	5.2	Quotations, etc. . . . .	24
	5.3	List structures . . . . .	24
	5.4	Tables of contents, and reference issues . . . . .	25
<b>1</b>	<b>2</b>	<b>Introduction</b>	
<b>2</b>	<b>4</b>	<b>The esameta packages and their documentation</b>	
2.1	5	‘Literate programming’ . . . . .	
2.2	7	Languages and metalanguages . . . . .	
2.3	8	Beginnings of <code>esameta.sty</code> . . . . .	
2.4	8	Typesetting of verbatim code, logos, and dates . . . . .	
<b>3</b>	<b>10</b>	<b>Page layout, fonts, and section titles</b>	
3.1	10	The layout of pages and paragraphs . . . . .	
3.2	13	Fonts . . . . .	
3.3	14	Section titles, numbering, and headers . . . . .	
<b>4</b>	<b>17</b>	<b>Specifying and typesetting languages</b>	
4.1	17	Checking spelling . . . . .	
4.2	17	Multi-lingual support . . . . .	
4.3	18	The table of translations . . . . .	
4.4	19	Mathematics . . . . .	
<b>5</b>	<b>21</b>	<b>Text structures</b>	
5.1	21	Notes . . . . .	
		6	<b>Floats and images</b> <b>28</b>
		6.1	Text boxes . . . . . 28
		6.2	Tables . . . . . 29
		6.3	Graphics and figures . . . . . 32
		<b>7</b>	<b>Referencing and cross referencing</b> <b>33</b>
		7.1	Indexing . . . . . 33
		7.2	Bibliographic citations and references . . . . . 34
		7.3	Cross-references . . . . . 35
		7.4	Hyperref for pdf files . . . . . 36
		7.5	The postponed definitions of textbox . . . . . 39
		<b>8</b>	<b>Using esameta.sty</b> <b>39</b>
		8.1	The system of L <sup>A</sup> T <sub>E</sub> X-related files 39
		8.2	Setting up of documents . . . . . 40
		8.3	Selecting output formatting . . . . . 42
		8.4	Exploiting error messages and warnings . . . . . 44
		<b>9</b>	<b>Discussion</b> <b>45</b>
			<b>Bibliography</b> <b>46</b>

## 1. Introduction

If we produce a complex book or a large set of notes and papers, we need an advanced system for document development and typesetting. If we include a lot of mathematics and graphics,  $\LaTeX$  is a good choice. Even outside the core areas for  $\LaTeX$  use—mathematics, computer science, physics, theoretical microeconomics, etc.—it can be of major importance.  $\LaTeX$  emphasises document structure and the logical mark up of text in contrast to a WYSIWYG system consisting of, for instance, MS Word, Equation Editor, and EndNote. In the latter system, layout details tend to pester document development, the consistent change of equations and figures is very difficult, and systematic cross-referencing is not really a part of the game. The initial ease of applying the what-you-see-is-what-you-get principle turns to increasingly difficult-to-solve problems.

The author experienced these problems when writing the book *Evolutionary Economics* (Andersen, 1994) with the Word–EqEditor–EndNote system—and decided to turn to  $\LaTeX$ . After a period during which the WYSIWYG system again came to dominate, the development of two large book projects motivates an update of the applied  $\LaTeX$  system. Andersen (2006a) gives a general account for this system, and some of its  $\TeX$ nic details are presented in the present article. The provided solutions can be used by a large number of programs on practically any computer system. However, the author presently uses a  $\LaTeX$  system on MS Windows computers. The presently selected editor programs, WinEdt<sup>1</sup> (see Figure 2 on page 4) and  $\TeX$ nicCenter, provide interfaces to the Mik $\TeX$  distribution of  $\LaTeX$  and to the bibliographical database program Jabref. Thus, WinEdt and  $\TeX$ nicCenter serve as entries to the world of  $\LaTeX$ . The documented  $\LaTeX$  package is *not* such an entry point. On the contrary, it is a *personal* tool for its author.<sup>2</sup> However, many of the solutions as well as the documentation system might be of relevance for fairly advanced  $\LaTeX$  users. The solutions and their documentation also give an impression of the power of the  $\LaTeX$  system to any interested reader.

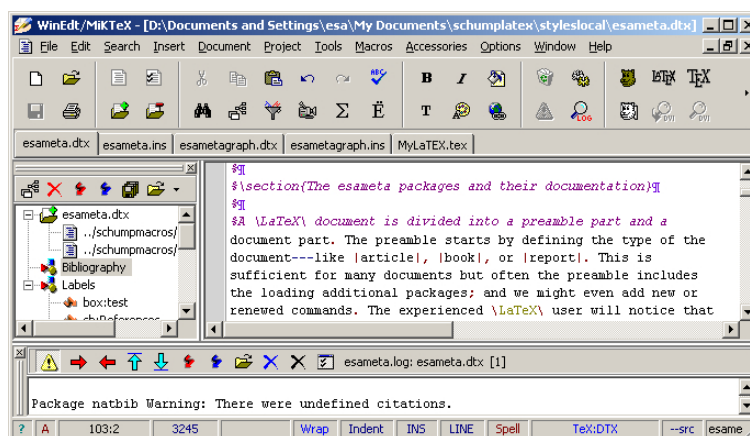
It should be noted that author's first turn to the  $\LaTeX$  system (see Andersen, 1997) was not entirely successful—although several manuscripts and papers were produced (see e.g. Andersen, 2001a; 2001b). One of the problems is that the skills obtained during a  $\LaTeX$ -paper project are often half-forgotten when they are needed for the next project. Furthermore, the next project might need  $\LaTeX$  packages that are not compatible with those used previously. Therefore, we need a 'test bed' for checking the functioning and compatibility of new additions of packages. Finally, the  $\LaTeX$  classes and packages provided by universities (e.g. for the production of PhD theses) and publishers (e.g. Springer and Kluwer) tended to undermine the development of the personalised  $\LaTeX$  code in the document preambles and in the auxiliary personal package that was originally called `esa.sty`.

In retrospect, the problem was that the author only slowly developed an understanding of the concept of a personal  $\LaTeX$  metapackage. When finally grasping and implementing this concept, he followed a tradition that dates back to Donald Knuth's invention of the  $\TeX$  system as a personal tool for the high-quality typesetting of his multi-volume book

---

<sup>1</sup>The WinEdt program is an alternative that in several respects is better. It is widely used in the  $\LaTeX$  community (Kopka and Daly, 2004, 15).

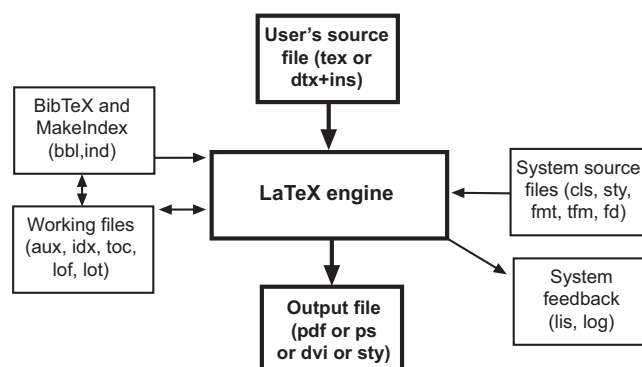
<sup>2</sup>Normal  $\LaTeX$  packages have users that require that the basic features of the packages are kept unchanged—and that they function in a large number of environments. This is *not* the case for a personal metapackage. Thus, the `esameta` package and its documentation is never 'frozen' to a larger degree than suggested by the difficulties of changing the author's stock of  $\LaTeX$  documents. Furthermore, the package and the present document will be corrected and changed without notice—except for the version identifier. The present version is 2007-10-04, 07:07.

Figure 1: The WinEdt editor program for interfacing with  $\text{\LaTeX}$ 

*The Art of Computer Programming* (Knuth, 1997). Today, a tailor-suited metapackage can still be used as a toolbox for the quick and consistent production of documents—and it is much easier to develop thanks to the efforts of the community of  $\text{\LaTeX}$  users and programmers. The costs of developing a basic version of a tailor-made metapackage are still considerable. So are the costs of transforming the basic  $\text{\LaTeX}$  versions of our documents to the in-house  $\text{\LaTeX}$  styles of some publishers—or to MS Word if the publisher does not accept  $\text{\LaTeX}$ . The present author’s experience, however, is that these high costs are rewarded by very significant productivity gains and by the ability to handle levels of complexity that were unthinkable before developing the metapackage. The constantly developing metapackage also go hand-in-hand with the incremental development of our  $\text{\LaTeX}$  skills. An additional advantage is that the metapackage allows us to standardise appearance of manuscripts and working papers that are made available at our websites.

The metapackage must change to survive as a tool for our projects. This process of change has some similarity with an evolutionary process—more or less like Stroustrup (1994) described it for the C++ programming language. The code of the metapackage is *replicated* for new projects; these projects also *innovate* the code by additions and deletions; and the projects serve as an ever-changing *selection environment* that checks the fitness of variants of the metapackage. Even though we at any point of time only see the code of the metapackage, the design of this code is largely the outcome of such an ‘evolutionary’ process. As we shall see below, this process is heavily influenced by the double nature of the output produced with the help of the metapackage: it produces on-line pdf documents with hyperlinks and well as printer-friendly pdf files. One of the consequences is that the important  $\text{\LaTeX}$  packages that exploits the PostScript format cannot be used directly.

The present article documents the  $\text{\LaTeX}$  packages and macros whose dominant ‘selection environment’ is the needs related to the development of two books on the economist Joseph A. Schumpeter and his evolutionary theory (Andersen, ?; 2006c)—as well as the related papers. Some features similar to those of the present metapackage are present in packages and ‘templates’ provided by publishers and universities. However, since major layout decisions will be taken later, the basic principle has been to use the  $\text{\LaTeX}$  the standard document classes as much as possible. The ones used for the book projects are report (for working documents and on-line documents) and book (for serious comment-

Figure 2: File types in the  $\text{\LaTeX}$  system

ing of prints)—as well as *article*. However, the Schumpeter books (and the related papers) are pretty complex, so many additions have been needed. These additions are largely covered by [Mittelbach and Goossens \(2004\)](#)—but some basic issues have required reference to the  $\text{\TeX}$  documentation ([Knuth, 1990](#)). Furthermore, the complexity has been increased because a choice had to be made with respect to the file type of the output (see [Figure 2](#)). The main choice fell on *pdf* files that serve communication of the Internet, but this means that the facilities provided by systems based on other types of file (*ps* and *dvi*) are not directly applicable. To standardise the access to the important (graphics) tools that use these formats, the present document produces two versions of the metapackage: the PDF-oriented *esameta.sty* and the PostScript-oriented *esametaps.sty*.

It is beyond the scope of the present article to discuss the software and  $\text{\LaTeX}$  distributions needed for the personal metapackage (see [Andersen, 2006a](#)). However, they include a text editor that supports the production of syntactically correct documents and that serves as an interface to the programs and files of the  $\text{\LaTeX}$  system. For the author's MS Windows computers,  $\text{\TeX}$ nicCenter has been chosen such an editor (see [Figure 1](#) on the previous page). It can interface with different  $\text{\LaTeX}$  distributions, of which  $\text{\MikTeX}$  was chosen. The production of *pdf* output by using  $\text{\TeX}$ nicCenter with the option `LaTeX => PDF`. Thus an efficient  $\text{\LaTeX}$  editor program does much more than replace MS Word and Equation Editor. However, it does not replace EndNote's handling of the maintenance of bibliographic databases. For this purpose the JabRef program has been selected. However, an editor program like  $\text{\TeX}$ nicCenter serves to ease the integration of this database into our documents. It also supports the decomposition of documents into a file database. The efficient organisation of such a text database is crucial. Actually, some knowledge about the theory of (relational) databases ([Date, 2004](#)) is very useful for the development of the file system and other data structures relating to the development of complex documents.

## 2. The esameta packages and their documentation

A  $\text{\LaTeX}$  document is divided into a preamble part and a document part. The preamble starts by defining the type of the document—like *article*, *book*, or *report*. This is sufficient for many documents but often the preamble includes the loading additional packages; and we might even add new or renewed commands. The experienced  $\text{\LaTeX}$  user will notice that preambles can often be reused for different documents. Therefore,

the different types can be placed in ordinary  $\LaTeX$  files (with extension `.tex`) but they can also be placed in package files (with extension `.sty`). The difference is that within package files the `@` character is interpreted as part of commands while the `@` character in `tex` files is simply a symbol for making email addresses. Thus we can paste our preamble into the package file called `esa.sty` and load it after we have defined the type of document. The loading is made by the command `\usepackage{esa}`. This package will normally be multiplied into `esa1.sty`, `esa2.sty`, ... to adapt to different types of document. However, we may also try use  $\LaTeX$  programming (with `if-then` commands) to let the `esameta.sty` package adapt itself to different situations. Finally, we may remark that our package needs to be documented. This can be done by comments inserted into the `esameta.sty` file. However, this is a very primitive form of documentation. The alternative is to turn to ‘literate programming’—of which the present article is a rather strange example. In any case, it should be noticed that the article is constrained by the fact that the file that produces the article is also used for the production of the  $\LaTeX$  package that gives the commands needed for the production of the article!<sup>3</sup> Let us consider this strange loop.

### 2.1. ‘Literate programming’

Extensive facilities for the writers of professional packages has been developed by the  $\LaTeX$  community. The basic principle of this ‘literate programming’ (Knuth, 1992) is that the code and its documentation should be integrated. The level of integration obtained by making comments within the code file is not acceptable since hardly anyone reads this code and since the writers of code, therefore, tend to ignore the necessary updates of their comments. Instead, documentation and code should be printed together as real articles. Furthermore, means should be provided to produce the actual program code as separate files.

It is especially Frank Mittelbach (see Mittelbach and Goossens, 2004, Ch. 14) who has developed the  $\LaTeX$  system for literate programming. According to this system packages have been developed in integrated documents that produce two types of output. Furthermore, the system provides extensive facilities for the production of ‘change histories’ and automatic indexing. One of the purposes of present article is to demonstrate that this system is also applicable and helpful for writers of personal metapackages. A minor difference is that the tools for recording the package history and indexing the code is of limited use (and they have been excluded from the present article). The major difference is the free style of the presentation of `esameta`.<sup>4</sup> In contrast to standard documentation documents, we include a large number of examples from different areas of application. These examples serve a double purpose. First, they make it easy to return to a package that has largely been forgotten since the last time it was used. Second, the production of the present highly complex document quickly reveals whether the addition of new  $\LaTeX$  features has caused problems for those features already in use.

The  $\LaTeX$  documentation system consists of an integrated file from which both the package itself and the package documentation are produced. One output is the  $\LaTeX$  package (the `sty` file) and the other output is the documentation (e.g., in `pdf` format). It

---

<sup>3</sup>Exceptions are the production of game matrices that is incompatible with the `array` package—as well as all the packages that depend on the production of PostScript output. The latter applies the `esameta.sty` version of the present package.

<sup>4</sup>The free style, however, is constrained by the fact that the sequence in which code is defined has to take into account the side effects of the commands. For instance, `hyperref` modifies the commands of previously loaded packages.

**The file esameta.dtx:**

```

% \iffalse
%<*driver>
\documentclass[a4paper]{ltxdoc}
\usepackage{esameta}
\begin{document}
  \DocInput{esameta.dtx}
\end{document}
%</driver>
% \fi
%\title{A personal metapackage}
%\maketitle
%\section{Introduction}\label{sec:intro}
%This is the documentation of the metapackage called |esameta|.
%It starts by defining itself.\DescribeMacro{\ProvidesPackage}
%  \begin{macrocode}
\ProvidesPackage{esameta}
%  \end{macrocode}
%\section{Tasks}\label{sec:tasks}
%This package will be developed gradually as needs emerge
%in relation to the writing of articles and books.

```

**The file esameta.ins:**

```

\input docstrip.tex
\askforoverwritefalse
\generate{\file{esameta.sty}\from{esameta.dtx}{esameta}}
\end

```

Figure 3: Rudimentary definition and documentation of the esameta package

is important to consider these three files (see Figure 3).

dtx files	<p><b>Integrated file:</b> The integrated document is defined in a dtx file called <code>esameta.dtx</code> (see Figure 3). The different parts of this file is distinguished by whether or not paragraphs start with the comment mark <code>%</code>. Documentation is placed in the <code>%</code>-marked paragraphs. Package code has no <code>%</code> but it is enclosed by the <code>macrocode</code> environment that has to be preceded by a <code>%</code> and four spaces (<code>%_</code>). Thus it is marked in the following way:</p> <pre> %  \begin{macrocode} Commands ... %  \end{macrocode} </pre>
<p>doc.sty ltxdoc.cls pdf files</p>	<p><b>Documentation file:</b> To produce the package documentation (<code>esameta.pdf</code>) we run <math>\LaTeX</math> and <math>\BibTeX</math> directly on the dtx file with the document class <code>ltxdoc</code> by means of the <code>doc</code> package. The result includes both documentation text and the verbatim printing of the package code. However, the <code>doc</code> package starts reading the commands placed within the strange <code>driver</code> environment (see Figure 3). When <code>doc</code> executes the command <code>\DocInput</code>, it processes the other parts of the dtx file.</p>
<p>docstrip.sty ins files sty files</p>	<p><b>Package file:</b> To produce the package itself, we run the <code>docstrip</code> package via a helping</p>

file called `esameta.ins`.<sup>5</sup> The program is short as demonstrated in the lower part of Figure 3 on the previous page. Its function is to read the pure code lines and write them into the file `esameta`. The figure shows a simplified version. The applied docstrip driver is designed to handle conditional code in order to two different versions of `esameta` (`esameta.sty` and `esametaps.sty`).

‘Recursion’ in `esameta.dtx`: As we have seen, the present documentation of `esameta` is one of the outputs of such a system of ‘literate programming’ (Knuth, 1992). The other part only consists of the numbered-line code of the present document. The special feature for the present documentation of `esameta.sty` is that this complex documentation is produced by means of `esameta.sty`. This feature might be called recursion—but in practice it is iteration: At a given point of time  $t$ , we save `esameta.dtx`. Then we at time  $t + 1$  run `esameta.ins` to produce `esameta.sty` (and `esametaps.sty`). Finally, we at time  $t + 2$  run `esameta.dtx` through the  $\LaTeX$  engine (or, more specifically, through `pdfTeX`) to produce `esameta.pdf`. With an efficient  $\LaTeX$  system, the whole procedure only takes a few seconds. This is extremely important since difficult-to-detect errors and mistakes will occur both in the documentation part and the package part of the integrated `dtx` document (see Section 8.4 on page 44).

## 2.2. Languages and metalanguages

As soon as we drop WYSIWYG text processing, we need to distinguish between the input of coded (or ‘marked-up’) text and the output document. Consider the following example:

The present version of the documen-	1 \noindent The present version of the
tation of <code>esameta.sty</code> was printed on	2 documentation of <code>\texttt{esameta.sty}</code>
2007-10-04. Its documentation at time $t$	3 was printed on <code>\printdate</code> . Its
is not always equal to its documentation	4 documentation at time $t$ is not always
at $t + 1$ .	5 equal to its documentation at $t + 1$ .

In the left column we have the coded text. The paragraph had been changed to become non-indented and `esameta.sty` is typeset with typewriter font. Furthermore, the date 2007-10-04 has been added and mathematical typesetting is included.

This description of the code used in a  $\LaTeX$  document distinguishes between input and output by formatting the former in typewriter font. However, we also need to describe the usage of  $\LaTeX$  commands and the definitions of these commands. For instance, `\texttt` is a command with an ‘argument’. The usage of this command can be generally described by `\texttt{<text>}`, where *text* is the text that we want to be typeset with typewriter font and the angled parentheses are omitted. This metalinguistic notation will be used in the present article. However, the command sometimes includes an optional argument so we use the notation `\commandname{<argument>}[<option>]`.

Another problem concerns the description of the definition of commands like `\printdate`. This definition is made by means of the `\newcommand` which has the following structure `\newcommand{<command name>}{<commands to be applied>}`.<sup>6</sup> The definition is

<sup>5</sup>When `esameta.ins` is run in a program like `TEXnicCenter`, an ‘emergency stop’ occurs, but the correct `esameta` is nevertheless produced.

<sup>6</sup>Actually, we do not have to use the curly braces around the *command name*—so some computer scientists call it ‘syntactic sugar’—but it at least serves pedagogical purposes. However, the notation is not always followed in the following, especially because we modify code from various sources.

```
\newcommand{\printdate}
  {\number\year-\two@digits\month-\two@digits\day}
```

The definition of `\printdate` illustrates a basic problem in the description of  $\LaTeX$  code. The problem is that Leslie Lamport (Lamport, 1994) built the  $\LaTeX$  system on top of Donald Knuth's  $\TeX$  system (Knuth, 1990). In other words, Lamport built  $\LaTeX$  as a high-level language that is defined by means of the low-level  $\TeX$  language. The commands available within the  $\LaTeX$  language are normally sufficient for the purposes of the `esameta` package. However, it is sometimes convenient or even necessary to use commands from the low-level  $\TeX$  system. This is the case in the definition of `\printdate`. Here all commands are made in  $\TeX$  code!

### 2.3. The beginnings of `esameta.sty` and `esametaps.sty`

When one of the `esameta` packages are loaded by  $\LaTeX$ , it starts by defining its own identity. The distinction is made by providing `docstrip` conditional code. Here is the part of the code that is used for the production of `esameta.sty`:

```
1 \*esameta&!esametaps)
2 \ProvidesPackage{esameta}
3 \</esameta&!esametaps)
```

The  $\LaTeX$  code is simply `\ProvidesPackage{<package name>}`. However, this code is placed within an enclosing environment that causes conditional inclusion of the code in the style files (Mittelbach and Goossens, 2004, 819–20). The beginning of this environment is actually written as `%<*esameta&esametaps>!`. It is a command to the `docstrip` program to place the code in `esameta.sty` *and not* (`&!`) in `esametaps.sty`. Similarly, we start the `esametaps.sty` file by

```
4 \*esametaps&!esameta)
5 \ProvidesPackage{esametaps}
6 \</esametaps&!esameta)
```

After identifying `esameta.sty`, we call a few basic tools that are generally needed: (1) facilities for simple programming of if–then decisions and while loops are provided by the `ifthen.sty` package, (2) the `calc` package provides simple calculations for programming purposes, and (3) the `etex` package allows us to use the radical increased of resources provided by the  $e\TeX$  program that is included in modern  $\LaTeX$  implementations (see Mittelbach and Goossens (2004, 871–7, and e.g. 907).<sup>7</sup> The use of the `ifthen` package, including the definition of the needed (Boolean) variables, is illustrated below (see e.g. Section 4.3 on page 18 and the paragraph on partial documents on page 43).<sup>8</sup> Here we find commands like `\ifthenelse{\boolean{PrintChapter}}{\<cmd if true>}{\<cmd if false>}`, where the *commands if true* are executed if `PrintChapter` evaluates to `true`.

```
7 \usepackage{ifthen}
8 \usepackage{calc}
9 \usepackage{etex}
```

### 2.4. Typesetting of verbatim code, logos, and dates

**Verbatim text:** The present document illustrates the need for presenting programming structures with special tools. They are provided by packages that allow verbatim type-

<sup>7</sup>Several of the packages loaded by `esameta.sty` make heavy use of  $\LaTeX$  ‘counters’. Therefore, we need to use `etex`’s expansion of the number of available counters from 256 to 32,768.

<sup>8</sup>New Boolean variables are initially set to `false`. However, we set this value explicitly to make clear what is happening.

setting of such constructs (Mittelbach and Goossens, 2004, 151–75). However, the `verbatim` command is modified by the `ltxdoc.cls` used for producing the present document. Therefore, we have to provide the ordinary `verbatim` features by the specialised command `\esaverbatim` that is included in the beginning of the ‘real’ documents:

```
\esaverbatim
verbatim.sty 10 \newcommand{\esaverbatim}{%
              11 \usepackage{verbatim,alltt,upquote,moreverb}}
              12 %\usepackage{program} %Incompatible!
```

In examples of the functioning of  $\LaTeX$  commands, the code is typically placed close to its result. The code is presented ‘verbatim’ in typewriter font while the output is presented as it is. This can most easily (and ‘fancily’) be done in two columns—as in Goossens et al. (1994) and Mittelbach and Goossens (2004). Here the code is placed in the right column and the result of executing the code is placed in the left column. The packages that produce these two-column examples of the use of  $\LaTeX$  commands are `fancyvrb.sty` and `fvr-b-ex.sty`. However, we need special options for the `esametaps` package so we only include `fancyvrb` and `fvr-b-ex` in `esameta.sty`. For general use we define a couple of commands (`\exante` and `\expost`) that standardise the appearance of the example boxes.

```
fancyvrb.sty fancyvrb and fvr-b-ex. However, we need special options for the esametaps package
fvr-b-ex.sty so we only include fancyvrb and fvr-b-ex in esameta.sty. For general use we define
\exante      a couple of commands (\exante and \expost) that standardise the appearance of the
\expost      example boxes.

              13 \esametaps
              14 \usepackage{fancyvrb}
              15 \usepackage{fvr-b-ex}
              16 \esametaps
              17 \newcommand{\exante}{\vspace{8pt}\noindent%
              18   \begin{minipage}{\textwidth}\small\hrule\vspace{3pt}}
              19 \newcommand{\expost}{\vspace{2pt}\hrule\end{minipage}\vspace{8pt}}
```

The relevant feature of the `fvr-b-ex` package is the `SideBySideExample` environment. This environment prints the `verbatim` command in the right column and its formatted result in the left column. The additional commands serves to create spacing and rulers. A simple application is to explain how  $\LaTeX$  codes and formats basic mathematics. The code is

```
\exante
\begin{SideBySideExample}
  [xrightmargin=0.5\textwidth,numbers=left]
  $s_i=x_i/x$ ...
\end{SideBySideExample}
\expost
```

The result is the following example box:

---

$s_i = x_i/x$	1 <code>\$s_i=x_i/x\$\\</code>
$\bar{z} = \sum_{i=1}^n s_i z_i$	2 <code>\$\bar{z}=\sum_{i=1}^n s_i z_i\$</code>

---

`\Metafont` **Logos:** When discussing the features of `esameta.sty`, it is helpful to apply the tradition of formatting basic programs in special ways. Logos like  $\TeX$  and  $\LaTeX$  have been defined in the loaded packages. We also need the definition of other  $\LaTeX$ -oriented logos like `METAFONT`, `AMS- $\LaTeX$` , and `BIB $\TeX$`  (as well as expressions like  $37^\circ\text{C}$ ). These features are programmed by means of low-level  $\TeX$  code:

```
20 \font\logo=logo10 scaled\magstephalf
21 \newcommand{\Metafont}{\logo METAFONT\null}
22 \providecommand{\BibTeX}{\rmfamily B\kern-.05em%
```

```

23 \textsc{i\kern-.025em b}\kern-.08em%
24 T\kern-.1667em\lower.7ex\hbox{E}\kern-.125emX}}
25 \newcommand{\AMSLaTeX}{\leavevmode\hbox{\mathcal A\kern-.2em%
26 \lower.376ex\hbox{\mathcal M$\kern-.2em\mathcal S$-\LaTeX}}
27 \newcommand{\degg}[1]{\mbox{\raisebox{3pt}{\mathcal S}\hspace{-.5pt}#1}}

```

`\today` Printing date and time:  $\LaTeX$  defines the command `\today`, which in the `babel` package (see Section 4.2 on page 17)—with the `british` option—states that the present document was printed on 4th October 2007. However, we would also like the more technical and standardised 2007-10-04. This is produced with `\printdate` (with low-level coding taken from the Swedish part of the `babel` package). To identify the printed versions of our documents precisely, we also need the time of printing. However, the  $\TeX$  command `\time` gives the minutes since midnight. To produce the more readable printing time (like 07:07 for the present document), we use a few higher-level calculations (with `calc`) to define `\printtime` (Mittelbach and Goossens, 2004, 873).

```

28 \newcommand{\printdate}{%
29 \number\year-\two@digits\month-\two@digits\day}
30 \newcounter{hours}\newcounter{minutes}
31 \newcommand{\printtime}{%
32 \setcounter{hours}{\time/60}%
33 \setcounter{minutes}{\time-\value{hours}*60}%
34 \ifthenelse{\value{hours}<10}{0}{}\thehours:%
35 \ifthenelse{\value{minutes}<10}{0}{}\theminutes}

```

### 3. Page layout, fonts, and section titles

As already mentioned, we want as few style changes as possible. However, it is helpful to redefine the space for text (the ‘body’ of the page) and other entities. To check out issues on the layout of pages, paragraphs, etc., we load the `layouts` package developed by Wilson (2004) (see also Mittelbach and Goossens, 2004, 199–205). This package has produced the present document’s figures that display layouts. Since the package is very specialised, it might seem a luxury to load it generally, but we shall take the chance.<sup>9</sup>

```

layouts.sty
36 \usepackage{layouts}

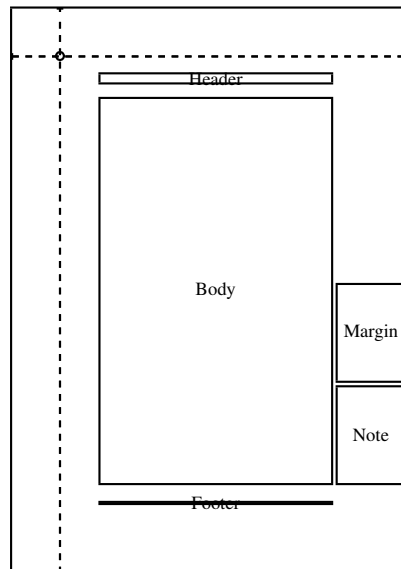
```

#### 3.1. The layout of pages and paragraphs

Page layout: The understanding of the design of  $\LaTeX$  pages helps to make the revisions needed for the special features of `esameta.sty`. This revision, however, requires the reading of special measurement units. When defining layout, the main measure is ‘points’ (pt)—but we also use millimetres and inches. For instance, the  $\LaTeX$  systems version of the height of A4 paper is 296.9965mm or 11.6952in or 845.04684pt (produced by commands like `\printinunitsof{mm}\prntlen{\paperheight}`). Given an understanding of these measures, we can inspect the chosen page layout depicted by Figure 4.

The figure demonstrates that the page body is placed near the middle of the paper page. This is especially suitable when using the `report` document class. The major interference with standard layouts, however, is that ample space is given for marginal

<sup>9</sup>Since the `layouts` package uses many  $\TeX$  counters, this luxury would be dangerous if we had not loaded the `etex` package that radically increases the maximum number of counters.



Lengths are to the nearest pt.

<code>page height = 845pt</code>	<code>page width = 598pt</code>
<code>\hoffset = 0pt</code>	<code>\voffset = 0pt</code>
<code>\oddsidemargin = 61pt</code>	<code>\topmargin = 27pt</code>
<code>\headheight = 12pt</code>	<code>\headsep = 25pt</code>
<code>\textheight = 578pt</code>	<code>\textwidth = 347pt</code>
<code>\footskip = 30pt</code>	<code>\marginparsep = 10pt</code>
<code>\marginparpush = 10pt</code>	<code>\columnsep = 30pt</code>
<code>\columnseprule = 0.0pt</code>	

Figure 4: The standard page layout of esameta.sty

notes (see Section 5.1 on page 21). The main reason is that this gives space for large editorial notes. The large spacing between these notes helps quick search for individual notes.

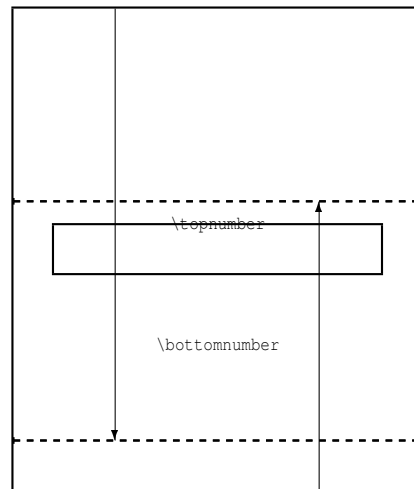
It is not all documents that need interference with standard page layouts. For instance, such interference is not warranted for the present document. Therefore, the layout described by Figure 4 will only be loaded if the command `\esapagelayout` is placed in the preamble of the document. This command is defined in the following way:

`\esapagelayout`

```

37 \newcommand{\esapagelayout}{
38   \setlength\textwidth{4.8in}
39   \setlength\textheight{8in}
40   \setlength\oddsidemargin{0.85in}
41   \setlength\evensidemargin{0.85in}
42   \setlength\marginparwidth{100pt}
43   \setlength\marginparsep{10pt}
44   \setlength\marginparpush{10pt}
45   \setlength\topmargin{27pt}
46   \setlength\headheight{12pt}
47   \setlength\headsep{25pt}
48   \setlength\footskip{30pt}
49   \setlength\columnsep{30pt}

```



```

\topnumber = 2          \topfraction = 0.899
\bottomnumber = 2       \bottomfraction = 0.600
\totalnumber = 4        \textfraction = 0.100

```

Figure 5: Float page layout for decreasing likelihood of float-only pages

```
50 \setlength\columnseprule{0pt}}
```

The most important issue for many complex documents is to increase the limited space the  $\LaTeX$  standard has given for ‘floats’ (figures, tables, and text boxes; see Section 6 on page 28). Otherwise, these items would often be placed on separate pages and even at the end of the document (Mittelbach and Goossens, 2004, 193–9, 283–8). This behaviour might be manipulated when producing the final manuscript—but it is disturbing during the development of the manuscript. The solution is to change the standard allocation of space for the floating items. The renewed allocation is depicted by Figure 5. This figure is somewhat confusing. The basic idea is that floats can be placed in three ways: where they occur [h], at the top of the page body [t], and at the bottom of the page body [b]. Since we normally prefer the top placement, we allow that only 10% of the page is used for ordinary text (the box in the middle of the figure). This means that 90% might be used for floats. These 90% may be used at the top of the text body while 60% may be used for bottom floats. We also allow 2 floats to be placed at the top and 2 at the bottom. This behaviour is obtained in the following way:

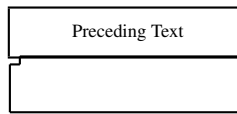
```
\floatpagefraction
```

```

51 \renewcommand{\floatpagefraction}{.9}
52 \renewcommand{\topfraction}{.9}
53 \renewcommand{\bottomfraction}{.6}
54 \setcounter{totalnumber}{4}
55 \setcounter{topnumber}{2}
56 \setcounter{bottomnumber}{2}
57 \renewcommand{\textfraction}{.1}

```

Paragraph layout: Yet another design task concerns paragraphs. Basically, we ignore this issue. However, it might later be relevant to manipulate the parameters. They are—for the present document—depicted by Figure 6.



Lengths are to the nearest pt.  
`\parindent = 15pt`      `\parskip = 0pt`  
`\baselineskip = 12pt`    `\linewidth = 355pt`

Figure 6: Paragraph parameters

When filling a page with text, a concrete issue is the splitting of paragraphs into lines with standardised spacing between words. If  $\text{\LaTeX}$  cannot obtain its very high level of perfection, it produces paragraphs in which some lines extend beyond the right text margin. The idea is that the user of the system should fix the problems. This, however, is a bad idea during the development of the books. Therefore, the document-specific command `\sloppy` resets the spacing requirements to a lower level of ‘perfection’.

`\sloppy`  
`multicol.sty`

The hyphenation difficulties becomes especially serious in multicolumn text—like the following two-column example:

We add the ability to print the text in multiple columns—like in the present paragraph. This feature is provided by the `multicol` package that defines the environment `multicols` (Mittelbach and Goossens, 2004, 184–9).

```
58 \usepackage{multicol}
```

### 3.2. Fonts

Computer Modern

**Basic fonts:** When the output version of a  $\text{\LaTeX}$  document is produced, METAFONT automatically supplies the different fonts in any needed scale. For this purpose font definitions are needed—both for the default Roman font and for specialised fonts. The standard  $\text{\TeX}$  font is Computer Modern that provides a good integration of ordinary text with mathematics. As long as we stay with this font, an adequate combination of the Roman default, the sans serif font, and the ‘typewriter’ font is provided.

We can also use more well-known fonts—like Times and Palatino—if we load them in both a text version and a mathematical version (see Mittelbach and Goossens, 2004, 513–23).

`mathptmx.sty`  
`mathpazo.sty`  
`\esatimes`  
`\esapalatino`

The Times combination can be loaded by the `mathptmx` package while `mathpazo` provides a version of Palatino. For the large Schumpeter book for quick printing, Times are better. The command `\esatimes` is used to switch on the combination with Times, and `\esapalatino` is used to switch on the combination with Palatino.

In both cases, we need to specify the auxiliary fonts. Here we make the same choices for both Times and Palatino:

`helvet.sty`

- Helvetica is used as the sans serif font (by loading `helvet` with a scaling option). We mainly use it for the typesetting of section titles and headers (see Section 3.3 on the next page).
- Courier is used as the ‘typewriter font’ (by loading the `courier` package). In combination with Times and Helvetica, Courier is loaded by the `pslatex` package. This package provides a

```
courier.sty
```

condensed version Courier (a 85% horizontal scaling). This hack is not really advisable—but it seems warranted during the development of the manuscripts because it gives room for URLs and long lines of code.

Combinations of Times/Palatino, Helvetica, and Courier are provided by the commands `\esatimes` and `\esapalatino`:

```
59 \newcommand{\esatimes}{\usepackage{mathptmx}%
60 \usepackage[scaled=.90]{helvet}\usepackage{courier}}
61 \newcommand{\esapalatino}{\usepackage{mathpazo}%
62 \usepackage[scaled=.95]{helvet}\usepackage{courier}}
```

The choice between Computer Modern, `\pslatex`, `\esatimes`, and `\esapalatino` is discussed in Section 8.3 on page 42.

**Multiple languages:** Irrespective of which font is used, the basic encoding is important for books that combine English with German, French, and Danish. One of the problems is obvious:

Normally we in $\LaTeX$ write	<code>_</code> Normally we in <code>\LaTeX\</code> write <code>\_</code>
<code>ä, â, æ, ø, å.</code>	<code>_</code> <code>"a, ^a, \ae, \o, \aa.</code>

```
fontenc.sty
```

The T1 and latin1 encodings serve all the Western European languages—so that e.g. ä, â, æ, ø, and å do not need special symbols. The T1 encoding (the ‘Cork encoding’) also means that characters can be produced by calling their (decimal, octal, or hexadecimal) number by `\symbol{⟨integer⟩}`. For instance, an explicit space like `_` is defined by `\symbol{32}` and an exclamation mark `!` is defined by `\symbol{33}`; see font tables in Kopka and Daly (2004, 370) and Mittelbach and Goossens (2004, 449). However, most of these symbols are also provided as commands with meaningful names. For instance, the backslash `\` is reserved for commands but it can be written by `\textbackslash` or by `\symbol{92}`.

```
inputenc.sty
```

```
latin1
```

```
\symbol
```

```
63 \usepackage[T1]{fontenc}
64 \usepackage[latin1]{inputenc}
```

```
yfonts.sty
```

Furthermore, we occasionally need old German fonts like in Schumpeter: „Gustav von Schmoller und die Probleme von heute“, and therefore we also call the `yfonts` package. This package produces the old-fashioned typesetting of Schumpeter’s ‘Gustav von Schmoller und die Probleme von heute’ by the commands `\swabfamily\fraklines Schumpeter: `Gu\stava ...` (Haralambous, 1991; Mittelbach and Goossens, 2004, 394–6).

```
65 \usepackage{yfonts}
```

```
soul.sty
```

```
\ul
```

**Underlined text:** The underlining of text is not really a font matter, and it is not encouraged in  $\LaTeX$ . However, it is useful for indicating entries to the Index (see Section 7.1 on page 33). Therefore, we load the `soul` package (Mittelbach and Goossens, 2004, 88–92). It provides the command `\ul{⟨text⟩}` that produces the underlining of text.

```
66 \usepackage{soul}
```

### 3.3. Section titles, numbering, and headers

**$\LaTeX$  sectioning:** The  $\LaTeX$  system gives strong support to a sectioning system that includes parts, chapters, sections, subsections, and paragraphs (Mittelbach and Goossens,

2004, 22–45). For instance, the command `\chapter{<Title>}` does not only produce a numbered title but also an entry to the Table of Contents. Furthermore, the counter `chapter` is incremented so that the next chapter gets the correct number and so that the sections of the chapter is automatically prefixed with the chapter number (which also can be printed directly by the command `\thechapter`). A somewhat similar approach is used for the titles and numbers of figures and tables.

Section titles: Among the consequences of a sectioning command is the use of a pre-defined layout for the section title (also called heading). Each section level, like part, chapter, and section, has its own format but it is influenced by general parameters. The  $\LaTeX$  standard is to use Roman font but it might be argued that it is easier to recognise (small versions of) titles if they are set in a sans serif font. In any case, it is relatively easy to switch between the two design strategies. We simply load the `titlesec` package (Mittelbach and Goossens, 2004, 36–45) with an appropriate option. The option is `sf` (sans serif font)—which when we use the font sets based on Times and Palatino means Helvetica. Thereby the formatting of nearly all titles change. However, we need manually to deal with the titles of book parts (see Section 5.4 on page 25) as well as the front page (see Section 8.3 on page 42).

`titlesec.sty`

```
67 \usepackage[sf]{titlesec}
```

The package `titlesec` supports all aspects of the formatting of titles as well as the inclusion of additional code (e.g. for the table of contents). We shall largely ignore these fancy possibilities but we, nevertheless, add punctuation after the section number. Other possibilities are exploited in Section 5.4 on page 25.

`\titlelabel`

```
68 \titlelabel{\thetitle.\enspace}
```

Sectioning without numbering: The ‘starring’ of sectioning commands has been the standard way of avoiding that the different kinds of document parts obtain numbering and are added to the Table of Contents. An example is `\chapter*{Preface}`. However, this command is problematic (1) by combining into one two different concepts (non-numbering and non-listing); (2) by being incompatible with some of the uses of the `titlesec` package; and (3) by making it impossible for the `hyperref` package (Section 7.4 on page 36) to create hyperlink targets that are placed correctly.

`secnumdepth`

The alternative is to control numbering by means of the `secnumdepth` counter and the entries to the table of contents by the `tocdepth` counter (Mittelbach and Goossens, 2004, 24, 50). If we set the `secnumdepth` counter to `-2` in the preamble of a document, then all parts (level `-1`), chapters (level `0`), sections (level `1`), etc. will be typeset without a number.

`starredsections`

To make localised omissions of title numbering (including the numbering of entries into the table of contents), we define the environment `starredsections`. If we, for example, enclose some of the chapter-level sections in the master file in this environment, then they will not include numbered titles. Afterwards, the depth of the numbering of sections is restored to `esasecdepth`.

`\esasecdepth`

```
69 \newcommand{\esasecdepth}{2}
70 \newenvironment{starredsections}
71   {\setcounter{secnumdepth}{-2}\renewcommand\sectionmark[0]{} }
72   {\setcounter{secnumdepth}{\esasecdepth}}
```

Using this environment on the above example, we obtain the non-numbered version of the Preface by the non-starred command `\chapter{Preface}`. This produces not only the non-numbered title but also wrong headers and wrong entries to the table of contents.

`\markboth` They are corrected by the command `\markboth`. We also add a label for cross-reference (see Section 7.3 on page 35).

```

\begin{starredsections}
  \chapter{Preface}
  \markboth{Preface}{Preface}
  \label{ch:preface}
  ... The contents of the preface ...
\end{starredsections}

```

This strategy only partially works for the chapter-like sections at the end of large documents. Thus, we have to solve additional problems with the References (see Section 7.2 on page 34). In other cases it have not been practically possible to exclude starred sectioning commands. For instance, the Index (see Section 7.1 on page 33) is still defined in this way—and starring is used relatively freely for Parts and Sections.

**Headers:** Let us now turn to the text placed above the page body—the page headers (see Figure 4 on page 11). The headers for the non-numbered Preface (and similar chapter-like parts of the document) were treated in the preceding paragraph. Now we turn to the general problem about the rather ugly-looking L<sup>A</sup>T<sub>E</sub>X standard for headers. Instead of sticking to this standard, we use ‘fancy headers’ provided by the `fancyhdr` package (Mittelbach and Goossens, 2004, 224–32):

`fancyhdr.sty`

```

73 \usepackage{fancyhdr}
74 \pagestyle{fancy}

```

We use his package to redefine the standard headers so that adequately formatted information on the book, the chapter, or the section is placed at the one end of the header and the current page at the other end. We first empty the pre-existing definitions, make a couple of overall definitions, and use the general L<sup>A</sup>T<sub>E</sub>X command to change the standard way chapters are presented in headers.

```

75 \fancyhead{}
76 \fancyfoot{}
77 \fancyheadoffset{0pt}
78 \renewcommand{\footrulewidth}{0pt}
79 \@ifundefined{chapter}
80   {\renewcommand{\sectionmark}[1]{\markboth{\thesection.\enspace#1}{}}}
81   {\renewcommand{\chaptermark}[1]{\markboth{\thechapter.\enspace#1}{}}}

```

`\fancyhead`

Then we apply new definitions by means of the command `\fancyhead`. The optional arguments to this command for a book (two-side printing) is RO (right on odd pages), RE (right on even pages), LO (left on odd pages), and LE (left on even pages). RE and LE are ignored for report and article (and gives a warning). To put into these slots we have `\thepage`, `\leftmark`, and `\rightmark`. The contents of these marks depend on the applied class. In book and report `\leftmark` contains the chapter information while `\rightmark` concerns the section information.

We define RO and LE as a formatted version of the page number. RE is set to a formatted version of the L<sup>A</sup>T<sub>E</sub>X command `\leftmark` that in book contains chapter information. The difficult issue is LO, where we have to distinguish between three cases. We have an article if no chapter is undefined. We have a book if our new Boolean variable `PrintBook` is true (but we can also use the command to print section information in the headers of a report). For article and report, we reuse the definition of RE. For books, LO is defined as a formatted version of `\rightmark`.

```

82 \fancyhead[RO,LE]{\sffamily\bfseries\thepage}
83 \fancyhead[RE]{\sffamily\nouppercase{\leftmark}}
84 \newboolean{PrintBook}
85 \setboolean{PrintBook}{false}
86 \fancyhead[LO]{%
87   {\@ifundefined{chapter}%
88    {\sffamily\nouppercase{\leftmark}}%
89    {\ifthenelse{\boolean{PrintBook}}%
90     {\sffamily\nouppercase{\rightmark. Rev\printdate}}%
91     {\sffamily\nouppercase{\leftmark. Rev\printdate}}}}

```

The thickness of the ruler beneath the header is determined by the command `\headrulewidth`. Generally, we want a ruler, but we have to take care of the problem of text boxes, figures and tables being placed just below the ruler. Since this looks strange, we remove the ruler by means of the command `\iftopfloat`. We also redefine `\cleardoublepage` to avoid headers on empty book pages. Within the center environment we might write a message like ‘Empty page’.

```

92 \renewcommand{\headrulewidth}{%
93   \iftopfloat{0pt}{\iffloatpage{0pt}{0.4pt}}
94 \renewcommand{\cleardoublepage}{\clearpage
95   \if@twoside \ifodd\c@page\else
96   \hbox{\vspace*\fill}
97   \begin{center} \end{center}
98   \vspace{\fill}\thispagestyle{empty}\newpage
99   \if@twocolumn\hbox{\newpage}\fi\fi}

```

To distinguish between the two books, we slightly alter the headers of the large book. This is done command `\esalargebook` (described on page 43), where we omit the ruler beneath the header text through

```
\renewcommand{\headrulewidth}{0pt}
```

## 4. Specifying and typesetting languages

### 4.1. Checking spelling

Spell checkers in many languages are included in most editor programs (including  $\TeX$ nicCenter). The checking of several languages included in the same document is difficult, however. Here the automatic language-detection facilities provided by MS Word are important. For this reason, relevant parts of the document are occasionally pasted into a Word document. Here the grammar is also checked, and the counting of words and characters is provided.

### 4.2. Multi-lingual support

Hyphenation and other language issues in a multi-lingual environment are taken care of by the complex `babel` package (Mittelbach and Goossens, 2004, Ch. 9).

```
100 \usepackage[danish,greek,french,german,english,british]{babel}
```

The last of the options to this package (here: `british`) is the dominant one. It is therefore `\today` has the format 4th October 2007. However, the dominant option can be overruled with respect to some language issue. If we e.g. need German hyphenation, we enclose the text in the `german` environment:

```
otherlanguage
```

---

```

1 In the entrepreneur
2 \begin{otherlanguage}{german}
In the entrepreneur 'der wesentliche
3 \der wesentliche Unterschied der
Unterschied der dynamischen Betrach-
4 dynamischen Betrachtungsweise
tungsweise gegenüber der den hedon-
5 gegenüber der den hedonischen
nischen Gleichgewichtsmenschen vor-
6 Gleichgewichtsmenschen
aussetzenden Statik liegt.' (Schumpeter,
7 voraussetzenden Statik liegt.'
1908)
8 \end{otherlanguage} (Schumpeter, 1908)

```

---

The hyphenation facilities provided by `babel` are impractical and not always sufficient. An additional device to resolve hyphenation problems in  $\LaTeX$  is provided by the

`\hyphenation`

`\hyphenation` command:

```

101 \hyphenation{Ent-wick-lung na-tio-nal-ö-ko-no-mie Shi-o-noya%
102 Spiet-hoff Schum-pe-ter Ba-werk Le-de-rer macro-eco-nomics%
103 micro-eco-nomics manu-script the-o-re-tis-chen Czer-no-vitz}

```

The fine-tuning of hyphenation is a part of  $\LaTeX$ 's quest for the production of 'perfect' documents (see e.g. Section 3.1 on page 10). During the development of our documents we accept much imperfection and apply the `\sloppy` command in the preamble of the documents.

### 4.3. The table of translations

The mix of languages is mainly due a need for including the originals of the many translations that are found in the Schumpeter books. They are brought in the Appendix that compares translations with the original quotations. This appendix is produced by means of the

`glossary.sty`

`MakeIndex` program and the `glossary` package (on the latter, see Talbot (2006)). This package uses the `longtable` environment (see Mittelbach and Goossens, 2004, 259–64) to produce glossaries and the only change that has been needed is to reset both columns to `p{0.5\textwidth}`.

```

104 \usepackage[hyper=true,header=plain]{../styleslocal/esaglossary}

```

The production of the entries for the translations table is made by the command

`\trans`

`\trans`, which is used as follows:

```

\trans{10412}{Oppenheimer}{hero epos}{Heldengedicht}

```

This command has four arguments: (1) the unique identifier of the translation;<sup>10</sup> (2) a short description of the source; (3) the translation; and (4) the original version. The `\trans` command puts the translation in the main text and add both the translation and the original to the Appendix 'Translations' (by first 'storing' the information and calling it):

```

105 \newcommand{\trans}[4]{#3 [T:#1]}
106 \storegloentry{#1}{name={\footnotesize#4 [#2]},%
107 description={\footnotesize#3 [T#1]}, sort={#1},%
108 format=hyperm}\usegloentry{#1}}

```

Through this command, the translations are sorted according to the unique identifier—thereby reflecting their order of appearance in the book. The alternative is to sort records by their source description. To choose this alternative, we have to set the global Boolean variable `glossortworks` to `true` (otherwise it is `false`); and within the Appendix 'Translations' we add commands of the format

`glossortworks`

---

<sup>10</sup>`MakeIndex` sorts the entries according to an integer identifier—and to create a sufficient name space, we start with an arbitrary digit, '1', followed by two digits for the chapter/appendix identifier and two digits for the identification of the translation within the chapter/appendix.

```

\ifthenelse{\boolean{glossortworks}}{\text if true}{\text if false}! The reset-
\esaglossortworks ting of the variable glossortworks is done by the command \esaglossortworks. The
major function of this command is to redefine \trans if it is called in the preamble of a
document.11
109 \newboolean{glossortworks}
110 \setboolean{glossortworks}{false}
111 \newcommand{\esaglossortworks}{%
112   \setboolean{glossortworks}{true}%
113   \renewcommand{\trans}[4]{##3 [T##1]%
114     \storegloentry{##1}{name={\footnotesize##4 [##2]},%
115     description={\footnotesize##3 [T##1]},%
116     sort={##2}, format=hyperrrm}%
117   \usegloentry{##1}}

```

Unfortunately, *MakeIndex* only accepts records of a maximum size of 1024 characters. To adapt to this restriction on the combined size of the translation and its original (including spaces and overheads), several large quotations needs to be split into two. Furthermore, *MakeIndex* gives a special meaning to the characters !, ", @, and l. One possibility for producing these characters is to use the applied font encoding (T1), where they are defined by `\symbol{33}`, `\symbol{34}`, `\symbol{64}`, and `\symbol{124}`.

The translations are activated by `\makeglossary` (after `TEXnicCenter` has been customised to work with the `glossary` package<sup>12</sup>).

```

118 \makeglossary
\glossaryname The production of the output (sorted after works) also requires that we put the following
\entryname commands at an appropriate place in the document:
\descriptionname \renewcommand{\glossaryname}{Translations}
\printglossary \renewcommand{\entryname}{Original}
\renewcommand{\descriptionname}{Translation}
\renewcommand{\glossarypreamble}{Introductory text.\par}
\printglossary

```

#### 4.4. Mathematics

Another language is that of mathematics. Although it is rather sparsely used except in a couple of chapters in the Schumpeter books, some of the rich facilities provided by any  $\LaTeX$  are necessary. We even need some of the extended mathematics features of the  $\mathcal{A}MS\text{-}\mathcal{L}aTeX$  packages (from the American Mathematical Society). One of the provided environments is `gather`. It functions better with the `hyperref` package (see Section 7.4 on page 36) than the standard `equation` environment. Therefore, we simply redefine `equation` to become a synonym of `gather`.

```

amsmath.sty
amssymb.sty
equation := gather
119 \usepackage{amsmath,amssymb,latexsym}
120 \let\equation\gather
121 \let\endequation\endgather

```

<sup>11</sup>Except for a single argument, the redefined command is identical to the original. Since the ‘inner’ redefinition takes place within the ‘outer’ definition of a command, the appearance differ: the double hash sign `##` has to be used to indicate the arguments of the inner definition.

<sup>12</sup>In `TEXnicCenter`’s Build/Define Output Profiles, we make a new output profile (`LaTeX => PDF (trans/glos)`) that is identical to everything in `LaTeX => PDF` except that the Command Line Arguments to *MakeIndex* is changed to `-s "bm".ist -t "bm".glg -o "bm".gls "bm".glo`. To generate correct hyperlinks in the translations table and in the index, we need to run both `LaTeX => PDF (trans/glos)` and `LaTeX => PDF` several times.

The facilities of  $\text{\LaTeX}$  and the  $\text{\AMS-L\LaTeX}$  packages are covered by Mittelbach and Goossens (2004, Ch. 8) and by Grätzer (1996). The use of some of them is sketched by Andersen (2006a).

As illustrated below, an equation can be typeset in several ways. If we enclose the equation in dollar signs ( $\dots$ ), it will be printed in-line in compressed form; if we enclose the equation in double dollar signs ( $\dots$ ), it will be displayed on its own line; if we enclose it in an `equation` environment, it will be displayed and numbered—like Equation (1).

$\dots$   
 $\dots$   
`equation`

The calculation is

$Q_t = \sum_{j=1}^n A_{jt} K_{jt}$ , where ...

$$Q_t = \sum_{j=1}^n A_{jt} K_{jt}$$

$$Q_t = \sum_{j=1}^n A_{jt} K_{jt} \quad (1)$$

```
1 The calculation is \ $Q_t = \sum_{j=1}^n
2 A_{jt}K_{jt}$, where \dots
3 $$Q_t = \sum_{j=1}^n A_{jt}K_{jt}$$
4 \begin{equation}
5 Q_t = \sum_{j=1}^n A_{jt}K_{jt}
6 \label{eq:Qt}
7 \end{equation}
```

The above examples demonstrate that the  $\text{\LaTeX}$  system makes a large number of decisions for us. These decisions include (1) centring of displayed equations with equation number to the right and an addition vertical space before and after the equation line, (2) condensed form of in-line equations, (3) printing of variables in italics with adequate horizontal space between them, (4) the use of a smaller font size for subscripts and superscripts, (5) etc. However, the typesetting of complex equations requires additional environments—like `align`, `cases`, and `multline`.<sup>13</sup> They are illustrated by the following equations and pseudo-equations:

`align`  
`cases`  
`multline`

$$r^2 = s^2 + t^2$$

$$x = \frac{y+z}{\sqrt{s+2u}}$$

```
1 \begin{align*}
2 r^2 &= s^2 + t^2 \\
3 x &= \frac{y+z}{\sqrt{s+2u}}
4 \end{align*}
```

$$f(x) = \begin{cases} -x^2, & \text{if } x < 0; \\ \alpha + x, & \text{if } 0 \leq x \leq 1; \\ x^2, & \text{otherwise.} \end{cases}$$

```
1 \begin{equation*}
2 f(x) =
3 \begin{cases}
4 -x^2, & \&\text{\text{if } $x < 0$;} \\
5 \alpha + x, & \&\text{\text{if } $0 \leq x \leq 1$;} \\
6 x^2, & \&\text{\text{otherwise.}}
7 \end{cases}
8 \end{equation*}
```

Theoretical Economics =  
‘Statics’ + ‘Dynamics’

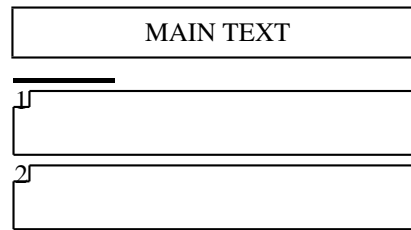
```
1 \begin{multline*}
2 \text{\text{Theoretical Economics}} = \\
3 \text{\text{'Statics'}} + \text{\text{'Dynamics'}}
4 \end{multline*}
```

`definition`  
`theorem`

We might also make use of the numbered environments `definition` and `theorem`.

`\newtheorem{definition}{Definition}`

<sup>13</sup>The starred versions of these environments exclude numbering.



Lengths are to the nearest pt.  
`\footins = 9pt`            `\footnoteseq = 7pt`  
`\baselineskip = 10pt`      `note separation = 40pt`  
`rule thickness = 0.4pt`  
`rule length = 0.25 times the \textwidth`

Figure 7: The footnote layout of `esameta.sty`

```
123 \newtheorem{theorem}{Theorem}
```

The use of these environments is illustrated by Definition 1 and Theorem 1.

<b>Definition 1</b> <i>An evolutionarily stable strategy is ...</i>	<pre>1 \begin{definition}\label{def:test} 2 An evolutionarily stable strategy is \ldots 3 \end{definition} 4 \begin{theorem}\label{thm:test}\raggedright 5 An evolutionarily stable strategy exists in 6 a two-dimensional pay-off matrix. 7 \end{theorem}</pre>
---	--

## 5. Text structures

### 5.1. Notes

Footnotes: The typesetting of footnotes is well supported by the L<sup>A</sup>T<sub>E</sub>X system (Mittelbach and Goossens, 2004, 110–25). Thus, each footnote can contain cross-references, bibliographic references, and several paragraphs. If provided with a label, the footnote can be referenced from elsewhere by number and/or by page (see Section 7.3 on page 35). If footnotes contain long words, we need to include the standard command `\raggedright`. The following example illustrates some of the functioning of footnotes (although it really shows a footnote in a minipage environment).

`\raggedright`

Schumpeter's *Wesen*.<sup>a</sup>

<sup>a</sup> *Das Wesen und der Hauptinhalt der theoretischen Nationalökonomie.*

This book served as the foundations for much of Schumpeter's teaching.

```
1 Schumpeter's \emph{Wesen}.\footnote{
2 \raggedright \emph{Das Wesen und der Hauptinhalt
3 der theoretischen Nationalökonomie}.\quad
4 This book served as the foundations for much of
5 Schumpeter's teaching.\label{fn:test}}
```

The placement of footnotes on the printed page is determined by several parameters. Their names and values are depicted by Figure 7.

The large Schumpeter book contains a large number of footnotes while they are ultimately excluded from the smaller book. Since text is copied from the large to the smaller book, the latter will include many `\footnote` commands. The default behaviour is to number them by Arabic numerals. However, the command `\romanfootnotes` switches to Roman numerals, especially to allow for a combination of footnotes and endnotes. The `\footnote` commands cannot be removed from the smaller book since the two books share a few files (like ‘Schumpeter’s Works’) that are provided with footnotes for the large book. Therefore, the smaller book needs a command for switching off the footnotes. This `\hidefootnotes` command might be called a quick hack—but it works according to its purpose!

`\romanfootnotes``\hidefootnotes`

```
124 \newcommand{\romanfootnotes}
125   {\renewcommand{\thefootnote}{\roman{footnote}}}
126 \newcommand{\hidefootnotes}{\renewcommand{\footnote}[1]{}}
```

**Endnotes:** During the development of the manuscripts, footnotes are better than endnotes. These footnotes will most likely stay in the larger Schumpeter book while they will be converted into endnotes in the smaller book. During the development of the books we need different combinations of footnotes and endnotes.

Endnotes are not included in the standard L<sup>A</sup>T<sub>E</sub>X styles. Instead they are produced with the `endnotes` package (see Mittelbach and Goossens, 2004, 125–26) that produces a list of endnotes that corresponds to Figure 7 on the previous page except that the ‘Main Text’ should be reinterpreted as the section title ‘Notes’ and that no ruler is added. When this package is loaded, we may use `\endnote{<text>}` in addition to `\footnote{<text>}`. As a standard, both types of note are numbered by Arabic numerals. The command `\romanendnotes` switches to Roman numbering of the endnotes. The command `\footnoteasendnote` transforms all footnotes to endnotes so that they are combined with the notes defined by the `\endnote` command.

`endnotes.sty``\romanfootnotes``\footnoteasendnote`

```
127 \usepackage{endnotes}
128 \newcommand{\romanendnotes}
129   {\renewcommand{\theendnote}{\roman{endnote}}}
130 \newcommand{\footnoteasendnote}{\renewcommand{\footnote}{\endnote}}
```

`\enoteheading``\enotesize``\enoteformat``\makeenmark`

The `endnote` package is not as developed and well-integrated as many other L<sup>A</sup>T<sub>E</sub>X packages—so care is needed.<sup>14</sup> We need to redefine a few endnote-related commands to fit the formatting styles of the Schumpeter books. First, we redefine `\enoteheading` to produce the heading of the section of endnotes for articles and endnotes placed at the end of chapters (if the endnotes ultimately are placed at the end of the book, we need to develop this command). Second, `\enotesize` is changed to small font. Third, we change `\enoteformat` so that e.g. the size of the paragraph indent corresponds to that used elsewhere in the book. However, the major change is made in the last three lines of that command. Apart from changing the layout of the endnote list we also define a hyper-link target for each endnote.<sup>15</sup> Fourth, the reference to this target is defined in `\makeenmark` that produces the endnote mark within the text as a superscripted numeral that in on-screen documents functions as a hyperlink.<sup>16</sup> Unfortunately, the hyperlink does

<sup>14</sup>The documentation is actually found inside the `endnote.sty` file!

<sup>15</sup>As in the case of hyperlinks, we need a unique label. In articles, this is the endnote number. In books and reports, this number is prefixed by the chapter number. We still have the problem of endnotes in unnumbered chapters!

<sup>16</sup>Hyperlink issues are discussed in Section 7.4 on page 36. The present solution is not perfect since following the hyperlink leads to the second line of the endnote. There are no link back, so we need to use the facilities of e.g. Adobe Acrobat Reader.

not bring us to the hypertarget but to the line immediately below that target!<sup>17</sup>

```

131 \renewcommand{\enoteheading}
132   {\section*{\notesname}\mbox{}\par\vskip-\baselineskip}
133 \renewcommand{\enotesize}{\small}
134 \renewcommand*{\enoteformat}
135   {\setlength{\rightskip}{0pt}
136    \setlength{\leftskip}{15pt}
137    \setlength{\parskip}{2pt}
138    \setlength{\parindent}{15pt}
139    \noindent\enotesize\leavevmode\hskip-20pt\makebox[20pt][l]%
140     {\hypertarget%
141      {\@ifundefined{chapter}{\theenmark}{\thechapter.\theenmark}}%
142      {\theenmark}}}
143 \renewcommand{\makeenmark}{\hyperlink%
144   {\@ifundefined{chapter}{\theenmark}{\thechapter.\theenmark}}%
145   {\textsuperscript{\theenmark}}}

```

Basically, we determine where to place the endnotes in the book by the command `\theendnotes`. They can be placed near the end of the document and sectioning of the list can be provided by the command `\addtoendnotes{<text and commands>}` placed within the chapter files. We initially place the endnotes after each chapter so we have to check whether the applied document class resets the endnote counter to zero after each chapter.<sup>18</sup>

We also have the problem of entering the Notes sections into the table of contents in a way that is recognised by the `hyperref` package. To solve and standardise the treatment of endnote printing, we embed the call of `\theendnotes` in a command called `\thechapendnotes`. This command does nothing unless we make the command `\setboolean{printendnotes}{true}`. This feature means that we can add the `\thechapendnotes` command to both books.

```

146 \newboolean{printendnotes}
147 \setboolean{printendnotes}{false}
148 \newcommand{\thechapendnotes}{%
149   \ifthenelse{\boolean{printendnotes}}
150   {\phantomsection
151    \addcontentsline{toc}{section}{\numberline{} \notesname}
152    \theendnotes
153    \setcounter{endnote}{0}}{}}

```

**Marginal notes:** Within `esameta.sty`, marginal notes are produced by the command `\esamarpar{<text>}`. Presently, they are used only for commenting. Marginal notes cannot be used to replace the indexing marking by underlining and the translation marking (by codes like T10208). One of the reasons is that marginal notes in footnotes are placed in a confusing way (often on the following page). Since the marginal notes are largely for the author, we provide the `\hideesamarpar` command for turning them off.

```

154 \newcommand{\esamarpar}[1]{%
155   \marginpar[\raggedleft\footnotesize #1]%
156   {\raggedright\footnotesize #1}}
157 \newcommand{\hideesamarpar}{\renewcommand{\esamarpar}[1]}{}}

```

<sup>17</sup>It probably takes too much time to remedy this problem—as demonstrated by the code of `hyperref.sty`.

<sup>18</sup>Another problem is to empty the external `ent` file. In the present version of the `endnotes` package this is done when the first endnote in the next chapter is written. If a chapter has no endnotes, it mysteriously ‘inherits’ the endnotes from the last chapter.

## 5.2. Quotations, etc.

The Schumpeter books also contain a lot of quotations. Therefore, they have a special environment. It simply changes the standard  $\LaTeX$  environment to small typeface. However, it is easy later to change the following definition:

```

esaquote
158 \newenvironment{esaquote}%
159   {\small\noindent\begin{quote}}
160   {\end{quote}}

```

Some of the quotations are translations. Therefore, any change of the definition of `esaquote` should be tested for compatibility with the `\trans` command (see Section 4.3 on page 18).

## 5.3. List structures

The manuscripts developed with `esameta.sty` make extensive use of list structures. Such structures are well supported by the  $\LaTeX$  system (Mittelbach and Goossens, 2004, 128–51). Actually, even the `quote` environment is defined as a list. However, it is not only quotations that needs a redefinition. Furthermore, even where the predefined list environments (`itemize`, `enumerate`, `description`) are appropriate, it seems warranted to give special names to the applied list environments. The reason is that each basic list type is really used for different purposes. If we give a special name to each of these purposes, then it is later easy to make fine tuning. Presently, such definitions have only been made for some of the logical list types.

The  $\LaTeX$  definition lists are not fully adequate for the purposes of the books. Therefore, we provide a version (`esadeflist`) that changes the label and use hardly any spacing between the items. Such a list has the following structure:

```

esadeflist
\begin{esadeflist}{\label} ... \end{esadeflist}.

```

The next line is typeset according to this definition (but the first label is defined as `\item[\texttt{label}]`):

`esadeflist`: Figure 8 depicts the parameters of an `esadeflist` list. The major tricks are to set the size of the label to zero and to give a negative intent to each item on the list.

Evaluation: The parameters look strange—but they work. However, it might be advisable to switch to e.g. the `paralist` package.

Definition: The environment `esadeflist` is defined in the following way:

```

161 \renewcommand{\descriptionlabel}[1]%
162   {\hspace{\labelsep}\textrm{#1:}}
163 \newenvironment{esadeflist}
164   {\begin{list}{}%
165    {\setlength\leftmargin{25pt}
166     \setlength\labelwidth{0pt}%
167     \setlength\parsep{3pt}%
168     \setlength\itemsep{0pt}%
169     \setlength\itemindent{-25pt}%
170     \let\makelabel\descriptionlabel}}
171   {\end{list}}

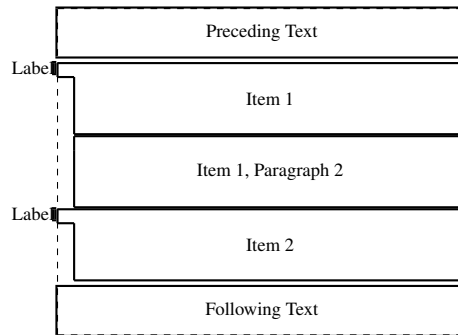
```

To typeset long lists (like the one in the Appendix ‘Dramatis Personae’) in a condensed version (like in the References), we use the following definition:

```

esalist
172 \newenvironment{esalist}
173   {\begin{list}{}%

```



Lengths are to the nearest pt.

```

\leftmargin = 25pt      \rightmargin = 0pt
\itemindent = -25pt    \labelwidth = 0pt
\labelsep = 5pt        \listparindent = 0pt
\topsep = 8pt          \parskip = 0pt
\partopsep = 2pt      \parsep = 3pt
\itemsep = 0pt

```

Figure 8: Layout of an esadeflist list

```

174 {\setlength\leftmargin{12pt}}%
175 \setlength\labelwidth{0pt}%
176 \setlength\parsep{3pt}%
177 \setlength\itemsep{0pt}%
178 \setlength\itemindent{-10pt}}
179 {\end{list}}

```

At the moment the notes of long tables are set as footnotes, but later they ought to follow the table. Here the following construct can be used:

```

esanotelist
180 \newenvironment{esanotelist}
181   {\begin{list}}{}%
182   {\footnotesize\setlength{\parsep}{3pt}}%
183   \setlength\itemsep{0pt}}
184 {\end{list}}

```

To develop further list structures, the `paralist` package (Mittelbach and Goossens, 2004, 132–38) is applied.

```

paralist.sty
185 \usepackage{paralist}

```

An example of its features of `paralist` is the `compactenum` environment, which can be enclosed in a quote environment:

- (1) Start with `\begin{quote}`
- (2) and `\begin{compactenum} [ (1) ];`
- (3) then write a sequence of `\item` followed by text;
- (4) end with `\end{compactenum}` and
- (5) `\end{quote}`

#### 5.4. Tables of contents, and reference issues

Entries to tables of contents: The production of tables of contents is normally functioning automatically. If not, we can simply add an entry by

```

\addcontentsline

```

```
\addcontentsline{<tabletype>}{<entrytype>}{<entrytext>}.
```

Here *tabletype* is the tables of contents (toc), figures (lof), tables (lot), or boxes (lob); *entrytype* determines its placement (e.g. part or subsection); and *entrytext* is the text to place in the table. Then the tables can be generated with the commands `\tableofcontents`, `\listoffigures`, and `\listoftables` (see also Section 6.1 on page 28).

However, these procedures do not work for the automatically generated parts of the document. Therefore, we have to redefine some of these commands. At the same time we add labels for cross-referencing (see Section 7.3 on page 35). Some of these redefinitions are made elsewhere in the document.<sup>19</sup> Presently, we only change the definition of the printing of the pages generated by the commands `\part{<partname>}` and `\part*{<partname>}`. These changes could have been made with the `titlesec` package but we instead copy the code from the `book` class definition and make a few changes. In addition to the making entries to the Table of Contents, the redefined commands also make labels that can be referred to. The reference labels for ordinary parts include their number (like `pt:II`) while the *partname* is used for starred parts (like `pt:Appendices`). Finally, we print the part title in Helvetica and distinguish between books/reports and articles (cf. Section 3.3 on page 14).

```
186 \newcommand{\esatitlefont}{\sf\bfseries}
187 \def\@part[#1]#2{%
188   \ifnum \c@secnumdepth >-2\relax\refstepcounter{part}%
189     \addcontentsline{toc}{part}{\protect\numberline{\thepart}#1}%
190     \else\addcontentsline{toc}{part}{#1}\fi
191   \markboth{}{}%
192   {\centering
193     \interlinepenalty \@M \normalfont
194     \ifnum \c@secnumdepth >-2\relax \@ifundefined{chapter}{\Large}{\huge}
195       \esatitlefont \partname\nobreakspace\thepart
196       \@ifundefined{chapter}{.\nobreakspace}{\par \vskip 20\p@}\fi
197       \@ifundefined{chapter}{\Large}{\Huge}
198       \esatitlefont #2\label{pt:\thepart}\par}%
199   \@endpart}
200 \def\@spart#1{%
201   \refstepcounter{part}\addcontentsline{toc}{part}{#1}%
202   \markboth{}{}%
203   {\centering
204     \interlinepenalty \@M \normalfont
205     \Huge \esatitlefont #1\label{pt:#1}\par}%
206   \@endpart}
```

This revision of the definitions from the `book` class is not elegant, but it serves to handle both the label issue and the Table of Contents. Simpler solutions can be used when the task is simply to add a troublesome document section to the table of contents (e.g. the method described in a paragraph on page 15 or the `hyperref` command `\phantomsection` used for the Index). This is partly done by document-specific commands. In Section 7.1 on page 33, an example is given that can also be used for adding the lists of figures, tables, and boxes to the Table of Contents.

**Local tables of contents:** A special task is to provide chapters with their own tables of contents. The argument for the addition of such tables is mainly that the chapters for

<sup>19</sup>See Section 7.2 on page 34 and Section 7.1 on page 33.

the books are distributed independently. The tool is the `titletoc` package (Mittelbach and Goossens, 2004, 58–66), which is closely related to the `titlesec` package (see Section 3.3 on page 14). Thus, we add mini-tables of contents by manipulating the definition of the chapter titles through the command `\titleformat`. At the same time we add minor changes to the standard formatting of chapter titles. However, our concrete use of the command `\titleformat` has serious side effects that *have* to be remembered. As emphasised in Section 3.3 on page 14, this requires that we use the ‘starred’ versions of the chapter title commands (including those implicit in e.g. `\tableofcontents` and `\bibliography`) with care.

```
titletoc.sty
\titleformat
```

```
207 \usepackage{titletoc}
208 \newboolean{esachaptoctoc}
209 \setboolean{esachaptoctoc}{false}
210 \titleformat{\chapter}{display}
211   {\sf\huge\bfseries}{\chaptertitlename\ \thechapter}{20pt}%
212   {\Huge\raggedright}[\ifthenelse{\boolean{esachaptoctoc}}%
213   {\vspace*{4pc}\normalfont\normalsize\startcontents
214   \printcontents{1}{1}{\setcounter{tocdepth}{3}}{}}]
```

The package `shorttoc.sty` adds the facility of including a short table of contents before the long table.

```
215 \usepackage[tight]{shorttoc} % alternative option = loose
```

```
\listoftables
listofmixed
```

The list of boxes, figures, and tables: There are two reasons for integrating the lists of boxes, figures, and tables into a single list. First, the individual lists are very short when we want them for individual chapters. Second, the list of boxes apparently creates strange problems for the creation of chapter-specific tables of contents (see Section 6.1 on the following page). The integrated list is produced by `\listoftables` that is redefined by standard T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X commands (Mittelbach and Goossens, 2004, 45–54). The chosen solution is to integrate the other lists into the renamed list of tables by commands like `\ext@figure{lot}` (which means that the list of tables is extended with the entries to the list of figures<sup>20</sup>). Since the individual items of the integrated list will often have the same number, we identify them as Box, Figure, and Table by changing the way they are formatted by defining/redefining `\l@Box`, `\l@figure`, and `\l@table`. In our documents, we have to redefine `\listtablename` to something like ‘Boxes, Figures, and Tables’. We have to postpone the commands concerning text boxes until Section 7.5 on page 39—but they are close to the following definitions:

```
216 \renewcommand{\ext@figure}{lot}
217 \renewcommand{\l@figure}[2]{%
218   \@dottedtocline{1}{0em}{2.25em}{Figure~#1}{#2}}
219 \renewcommand{\l@table}[2]{%
220   \@dottedtocline{1}{0em}{2.6em}{Table~#1}{#2}}
```

It should be noticed that the redefinition of the command `\listoffigures` implies that it cannot be used for producing a list that only consists of figures. This side-effect might later be overcome by producing the integrated list by a command that might be called `\listofmixed`.

<sup>20</sup>Actually, non-standard items created by `wrapfigure` are not included. If we had integrated tables into the list of figures, essential items would have been lost.

## 6. Floats and images

In  $\LaTeX$  documents, figures, tables, and text boxes are floated to a convenient place, such as the top of the next page. This placement is controlled by parameters set by the commands `\floatpagefraction` and `\textfraction` (see Section 3.1 on page 10). When a page is nearly filled with a floating item, it is placed at the end of the document together with the following floats. A `\clearpage` command will solve this problem, but it will also cause a break in the text. Therefore, a temporary change of the parameters might be warranted. Additional possibilities are provided by the `\afterpage` command. This command is found in the `afterpage` package (Mittelbach and Goossens, 2004, 289). One of the features of floating items is their captions. They can be managed by the `caption` package (Mittelbach and Goossens, 2004, 308–17). We presently use the options of reducing the size of the caption font, reducing the margins, and separating the label:

`\afterpage`  
`afterpage.sty`  
  
`caption.sty`

```
221 \usepackage{afterpage} % dangerous package!
222 \usepackage[font=small,format=hang,margin=10pt]{caption}
```

### 6.1. Text boxes

The text boxes serve several purposes. First, they present Schumpeter’s main works in a condensed format. Second, they allow the illustration and documentation of various points in the main text. Third, they break the visual monotony of the main text.

The text boxes are produced by means of the author’s modification of the ‘ruled float’ of the  $\LaTeX$  `float` package (Mittelbach and Goossens, 2004, 291–5). We start by loading a local copy of `float.sty`<sup>21</sup> and making some modifications. For instance, we change the ruler position, format the float name with normal font, add punctuation, and do not number boxes by section in articles.

`float.sty`

```
223 \usepackage{./styleslocal/float} % a local copy of float
224 \renewcommand{\fs@ruled}{%
225   \def\fs@cfont{\rmfamily}\let\fs@capt\floatc@ruled
226   \def\fs@pre{\hrule\kern2pt}%
227   \def\fs@post{\kern2pt\hrule\relax}%
228   \def\fs@mid{\kern2pt\hrule\kern2pt}%
229   \let\fs@iftopcapt\iftrue}
230 \renewcommand{\float@listhead}[1]{%
231   \def\tempa{\chapter*}\@tempa{#1\@mkboth{#1}{#1}}%
232 \@ifundefined{chapter}
233   {\def\float@newx#1[#2]{\@ifundefined{c@#1}{\newcounter{#1}%
234     \expandafter\edef\csname the#1\endcsname{\noexpand\arabic{#1}}}}{}
235 \@ifundefined{chapter}}{}
236   {\renewcommand{\floatc@ruled}[2]{\@fs@cfont #1.} #2\par}}
```

The use of the `float` package requires caution in relation to the `hyperref` package (see Section 7.4 on page 36). The problem is that `hyperref` changes the commands of `float` and that we cannot use the modified commands before they have been made. Therefore, we have to postpone the real definition of them until the very end of `esameta.sty` (see Section 7.5 on page 39). However, it is convenient to present them in connection with the discussion of text boxes. We start by using the modified `float` package for creating `Box` as an ordinary float with a the name `Box`, the standard placement `top`, the list of boxes `lob`, and the numbering within chapter.

`Box`

<sup>21</sup>Strangely, the text boxes revert of double-line captions if `float.sty` is loaded from its normal position in the  $\LaTeX$  file system.

**Box 1** A text box

This is a text box. As described in the main text, such boxes are defined in a rather unusual way.<sup>a</sup>

---

<sup>a</sup>For instance, their contents should be enclosed in curly braces.

---

```
\floatstyle{ruled}
\newfloat{Box}{t}{lob}[chapter]
```

Then the `Box` environment is used in the definition of the `textbox` environment, where the caption and the contents are provided by arguments to the beginning of the environment. Furthermore, the contents of the text box is placed in a `minipage`. This `minipage` can include footnotes, and it only fills 95% of the text width.

`textbox`

```
\newenvironment{textbox}[2]%
{\begin{Box}[t]%
 \caption{#1}%
 \centering\vspace{6pt}\small%
 \begin{minipage}{0.95\textwidth}#2%
 \end{minipage}%
 \vspace{4pt}}%
{\end{Box}}
```

The `textbox` environment produces **Box 1** in the following way:

```
\begin{textbox}{A test box}
{This is a text box. As described in the main text, such boxes
are defined in a rather unusual way.\footnote{For instance,
their contents should be enclosed in curly braces.}
\label{box:test}}
\end{textbox}
```

The list of text boxes can be produced at an appropriate place in the beginning of the document make the command

```
\listof{Box}{List of Text Boxes}
```

However, this list interferes strangely with the inclusion of chapter-specific tables of contents. Therefore, we instead have chosen to produce an integrated list of boxes, figures, and tables (see the paragraph on the topic on page 27).

## 6.2. Tables

**Basic tables:** The tabular material adds an important dimension to the Schumpeter books. Some of this material is included in text boxes (see Section 6.1 on the previous page), but for the purpose of standardisation all tables are put in separate files. The tables are often produced by means of a combination of the `tabular` and `table` environments with the addition of the `booktabs` package (Mittelbach and Goossens, 2004, Ch. 5).

```
tabular
table
booktabs.sty 237 \usepackage{multirow}
multirow.sty 238 \usepackage{booktabs}
239 \newcommand{\luft}{\ \\[2pt]}
```

Row description	Column description	
	Column 1	Column 2
Row 1	Case 1	Case 2
Row 2	Case 3	Case 4

Table 1: A table template

In the passing we may note that `\luft` represents an idiosyncratic attempt to systematise the spacing in tables has been used in some documents, but the facilities provided by the `booktabs` package seems more appropriate. Using the packages and the standard  $\LaTeX$  environments `table` and `tabular`, we can produce Table 1. Table 1 is produced by means of the following code:

```

\begin{table}[t]
\caption{\centering}
\begin{tabular}{p{3.1cm}cc}
\toprule
\multirow{2}{3cm}{Row description}&
\multicolumn{2}{1}{Column description}\\
\cmidrule(r){2-3}
&Column~1 &Column~2\\
\midrule
Row~1 &Case~1 &Case~2\\[1mm]
Row~2 &Case~3 &Case~4\\
\bottomrule
\end{tabular}
\caption{A table template}
\label{tab:test}
\end{table}

```

We might need to rotate wide tables so that they exploit the height of the page. The rotation is produced by the commands provided by the `sidewaystable` environment provided by the `rotating` package. The tables are placed on their own pages, but as seen by the example just below, we may also rotate a table within the text by the environment `sideways`.

```

rotating.sty
sidewaystable
sideways

```

```
240 \usepackage{rotating}
```

Rows	Columns			
	Col 1	Col 2	Col 3	Col 4
Row 1	C 1	C 2	C 3	C 4
Row 2	C 1	C 2	C 3	C 4

`array.sty` The array problem: The `array` package (Mittelbach and Goossens, 2004, 243–51) is

used to obtain better control over the formatting of table columns. Since it cannot function together with another relevant package (*sgame*, see below), we do not include it in *esameta.sty* but load it in the beginning of the documents that use it by the command

```
\usepackage{array}
```

**Long tables:** This command is necessary in the main files for the Schumpeter books since they make extensive use of the *longtable* package and since *array* is useful for the formatting of this kind of table. *longtable* is used for the construction of tables that runs over multiple pages (Mittelbach and Goossens, 2004, 259–64).

*longtable.sty*

```
241 \usepackage{longtable}
242 \newcommand{\esatabtitle}{Insert the table title}
```

*longtable*

Let us see how the table of contents for *History of Economic Analysis* is constructed:

```
\begin{longtable}>{\small\raggedright}p{10.5cm}>{\small}r}
\caption{Detailed contents of \emph{History}}
\label{tab:hea}\tabularnewline
\toprule &\emph{Page}\tabularnewline \toprule
\endfirsthead
&\emph{Page}\tabularnewline[2pt]
\endhead
\endfoot \endlastfoot
\textbf{Part I. Introduction: Scope and Method} &1\tabularnewline*
1. Introduction and plan &3\tabularnewline*
\quad{\footnotesize 1. Plan of the book} &3\tabularnewline
...
\midrule
Editor's appendix &1185\tabularnewline
...
Subject index &1231\tabularnewline*
\bottomrule
\end{longtable}
```

Here the first line is rather complex since it (1) begins the table, (2) defines the first column to be formatted as paragraphs with unequal length of the lines, and (3) defines the second column (with page numbers) as being right aligned. The second line defines the table caption and the third line defines a label for cross-reference. Then follows the formatting of the top of the parts of the table placed on the first and the following pages. The contents of the table is defined line by line. Between the lines we place different types of ruler lines.

We do not want the long table to be broken between pages at arbitrary lines. To control breaking, we use the ‘starred’ `\tabularnewline*` (or `\tabularnewline*[2pt]`). This command implies that no page break takes place before the next table line. If a break takes place, then the current line will be at the top of the next page.

`\tabularnewline*`

**Payoff matrices:** The matrix of the payoffs of a theoretical game is a special type of table that is treated separately by Andersen (2006b), mainly because the *sgame* package for game theorists is incompatible with *array.sty* that is needed for most of the tables of the Schumpeter books. Therefore, `\usepackage{sgame}` is not automatically loaded by *esameta.sty*. Instead, game matrices have to be developed separately and imported as small pdf files. When doing so, we need to load *sgame* by the command `\esagames:`

*sgame.sty*

`\esagames`

```

243 \newcommand{\esagames}
244   {\@ifpackageloaded{array}
245    {\PackageError{esameta}%
246     {sgame incompatible with array and not loaded}%
247     {sgame incompatible with array and not loaded}}
248   {\usepackage{sgame}}}}

```

Since the present document does not use the `array` package, it is possible to demonstrate how the basic information about a game of strategy is introduced in  $\LaTeX$  documents. Here we can use the `game` environment and the related table-like coding:

		Player 2		
		<i>L</i>	<i>M</i>	<i>R</i>
Player 1	<i>T</i>	2, 2	2, 0	0, 3
	<i>B</i>	3, 0	0, 9	1, 1

```

1 \begin{game}{2}{3}[Player~1][Player~2]
2   &$L$ &$M$ &$R$\\
3 $T$ &$2,2$ &$2,0$ &$0,3$\\
4 $B$ &$3,0$ &$0,9$ &$1,1$
5 \end{game}

```

### 6.3. Graphics and figures

Graphics add important information to the books. It cannot be treated here for two reasons (see [Goossens et al., 1997](#)). First, it is a topic quite different from that of ordinary typesetting. Second, the major way of producing graphics is by means of the PostScript language, a language that is incompatible with the direct production of pdf output. For these reasons, graphics is treated by means of the `esametaps.sty` version of the package. This version should be loaded before the `esametagraph.sty` package (see Andersen, 2006b). Thereby it can load and revise the graphics packages that use the PostScript language. Each piece of graphics is placed in separate file and imported into the main documents. The production of the graphics files is covered by Andersen (2006b).

The graphics (and much else) are normally placed in the standard  $\LaTeX$  figure environment ([Mittelbach and Goossens, 2004](#), Ch. 6). The graphics is included in the figures by the `\includegraphics` command. We use the `graphicx` package ([Mittelbach and Goossens \(2004, 613–28\)](#)) to define the optional arguments to this command (see Andersen, 2006b). We also add the `color` package. For both these packages it is important to specify whether the applied driver is `pdftex` or `dvips` ([Kopka and Daly, 2004](#), 154)—but this is normally done automatically.

```

\includegraphics
graphicx.sty
color.sty

```

```

249 \usepackage{graphicx}
250 \usepackage{color}

```

The standard import commands do not produce text that wraps around figures. To place photos of Schumpeter in the introductory paragraphs to the historical chapters, we need either `floatflt` or `wrapfig` ([Mittelbach and Goossens, 2004](#), 298–306). Although only the first is used at the moment, both have been included in the package:

```

floatflt.sty
wrapfig.sty

```

```

251 \usepackage{floatflt}
252 \usepackage{wrapfig}
253 \newlength{\wrapsep}
254 \setlength{\wrapsep}{1mm}
255 \newlength{\saveintextsep}
256 \setlength{\saveintextsep}{\intextsep}

```

## 7. Referencing and cross-referencing

References in general can be classified in index references, bibliographic references, and cross-references between elements of the document. The `hyperref` package supports these references with hyperlinks.

### 7.1. Indexing

Although indexing (Mittelbach and Goossens, 2004, Ch. 11) is normally considered the last ToDo in a book project, it also serves book development. As we shall see below, several facilities are available. But *MakeIndex* provides the basic facilities—if we make the following commands in the preamble:

```
index.sty
\makeindex 257 \usepackage{index}
258 \makeindex
```

To generate systematic index entries, the both the names of persons and the code names of Schumpeter’s works have been defined as L<sup>A</sup>T<sub>E</sub>X commands. If we, for instance, write `\Samuelson, Paul A.` Samuelson is placed in the index:

```
\newcommand{\Samuelson}{\ul{Samuelson}\index{Samuelson, Paul A.}}
```

The indexing of Schumpeter’s works is achieved by a more complex type of command. For instance, `\Cycles` is defined in the following way:

```
\newcommand{\Cycles}{\textsl{\hyperlink{Cycles}{Cycles}}%
\index{Schumpeter, dworks@\textbf{Schumpeter, works by year}!&
S1939a@\emph{Business Cycles} [\Cycles, 1939]}
```

The first part of this command defines that the work is slanted (`\textsl{Cycles}`) and made into a hyperlink (`\textsl{\hyperlink{Cycles}{Cycles}}`). Then follows an indexing command with the structure `\index{<index entry>}`, where the *index entry* is split in a first level and a second level divided by an `!`. At each level we first define how the entries should be sorted and then the text of the entry (`{<sorting info>@{<entry text>}`). Given this structure of the command, it should be clear that the first level of the index entry is `\textbf{Schumpeter, works by year}` and that it should be placed in the index before ‘Schumpeter, Elizabeth Brody’ because of `dworks`. At the second level the ordering of the specific works is made by year (‘S1939a’).

The concrete in-text citations are illustrated by the following example (where the hyperlink to *Cycles* does not work in the present document!):

---

Samuelson and Schumpeter’s <i>Cycles</i> ; mathe-	1 \Samuelson\ and Schumpeter’s \Cycles;
tical economics. [mathematical economics]	2 mathematical economics.\ICmathecon\

---

All the definitions of works, concepts, and names (several hundred commands!) are contained in three files. However, since the `esameta` file is also used for other projects, these files are called by `esameta`. The underlining of `Samuelson` and the parenthetical `[mathematical economics]` solely serve the indexation process so it shall, of course, be removed from the final version of the books. The indications of the indexing of names and concepts can be removed by setting the Booleans `PrintIC` and `PrintIN` to `false`. Since the indexing of concepts is developed gradually, a facility for switching it off is included. This is done by setting the Boolean `PrincICentries` to `false`.

```
259 \input{../schumpmacros/worksindexcmd}
260 \newboolean{PrintIC}
```

```
worksindexcmd.tex
conceptsindexcmd.tex
worksindexcmd.tex
```

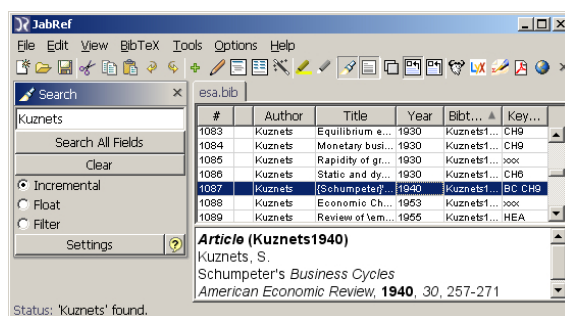


Figure 9: The JabRef program.

```

261 \setboolean{PrintIC}{true}
262 \input{../schumpmacros/conceptindexcmd}
263 \newboolean{PrintIN}
264 \setboolean{PrintIN}{true}
265 %\input{../schumpmacros/namesindexcmd} %called by esametax
266 \newboolean{PrintICentries}\setboolean{PrintICentries}{true}
267 \newcommand{\cindex}[1]%
268   {\ifthenelse{\boolean{PrintICentries}}{\index{#1}}{}}

```

Finally, we come to the difficult problems of adding the Index to the table of contents and to the cross-reference system. These problems are solved by two document-specific additions to the call of the index at the end of the document. The first addition is the command `\phantomsection`<sup>22</sup> followed by an `\addcontentsline`. Then we use the full version of the `\printindex` command with an optional prologue:

```

\phantomsection
\printindex[[]]

```

```
\printindex[indexname][prologue text],
```

where the *indexname* should be default if we refer to the ordinary index and the *prologue text* in an introductory text. It might be empty except for the label definition.

Here is an example of the document-specific additions to the `\printindex` command:

```

\cleardoublepage
\phantomsection
\addcontentsline{toc}{chapter}{\indexname}
\printindex[default][Some introductory text.\label{ch:Index}]

```

## 7.2. Bibliographic citations and references

The efficient handling of citations and references is a major advantage of a complete L<sup>A</sup>T<sub>E</sub>X system (Mittelbach and Goossens, 2004, Ch. 13). The exploitation of this advantage presupposes a bibliographic database in BIB<sub>T</sub>E<sub>X</sub> format (e.g., in JabRef; see Figure 9<sup>23</sup>). Then the task is to produce a bibliography by including the following commands at the end of the document:

```

esaschump.bst
esa.bib

```

```

\bibliographystyle{../biblio/esaschump}
\bibliography{../biblio/esa}%
\addcontentsline{toc}{chapter}{References}%
\label{sec:references}

```

<sup>22</sup>This command is provided by the newest version of the `hyperref` package, see Section 7.4 on page 36.

<sup>23</sup>Actually, much of the database was developed in EndNote, formatted for Refer, exported to a text file, and imported into JabRef.

These commands starts by loading a bibliographic format adapted for `natbib`. This format was generated by the  $\text{\TeX}$  program `makebst/custom-bib` that (when run through `docstrip` and loaded with `merlin.mbs`) asks the user a fairly large number of questions. By means of these answers, the program produces a customised bibliographic format (like the one used in the present document). This format applies the  $\text{\BIBTeX}$  style language (Mittelbach and Goossens, 2004, 805–12), and it has been slightly revised. A similar procedure can be used for modifying the bibliographic output for the specifications of any publisher. The next of the listed commands defines the file (`esa.bib`) with the bibliographic information for  $\text{\BIBTeX}$ . Finally, we add information for the table of contents and for cross-referencing.

A major tool for producing citations and references is `natbib` (Mittelbach and Goossens, 2004, 700–15). As mentioned by Andersen (2006a), `natbib` is the best package for author–year citations. In the books, the standard format is Andersen (2006a).

<code>natbib.sty</code>	<code>\citep</code>	(Mittelbach and Goossens, 2004, 7–15)	1 \citep[7--15]{Mittelbach2004}\
<code>\citet</code>		Andersen (2006a)	2 \citet{Andersen2006b}\
<code>\citeyear</code>		Andersen (2006a)	3 Andersen (\citeyear{Andersen2006b})

```
269 \usepackage{natbib}
```

The consequence of this package is not only that a citation is inserted (with the format defined by the five arguments of `\bibpunct`) but also that the reference is included in the bibliography at the end of the document.

```
270 \bibpunct{(){};}{a}{,}{,}{,}
271 \setlength{\bibsep}{0mm}
272 %\setlength{\bibhang}{0.6mm}
273 %\renewcommand{\bibindent}{0.0}
```

The bibliography is produced by the command `\bibliography{<database>}`, where *database* is the name of the local `bib` file. The standard command is redefined by `natbib` so that it is possible to include a preamble. We define a simple preamble that may be redefined in our documents. We also modify the `\bibsection` command from `\chapter*` to `\chapter` (see the paragraph on page 3.3 on page 15) so that it is treated correctly by the `hyperref` package (see Section 7.4 on the following page).

```
274 \newcommand{\bibpreamble}%
275   {\@ifundefined{chapter}%
276    {\label{sec:References}}{\label{ch:References}}}
277 \renewcommand{\bibsection}%
278   {\@ifundefined{chapter}%
279    {\section*{\bibname}\protect\markboth{\bibname}{\bibname}%
280     \addcontentsline{toc}{section}{\numberline{\bibname}}}%
281    {\chapter{\bibname}\protect\markboth{\bibname}{\bibname}}}
```

### 7.3. Cross-references

Cross-referencing is easy in  $\text{\LaTeX}$ . If *labelname* has not been used before, you define a label by `\label{<labelname>}` and refer to it by either `\ref{<labelname>}` or `\pageref{<labelname>}`. Thus you can insert the label-creating command `\label{test1}` here. Then `\pageref{test1}` gives ‘35’ and `\ref{test1}` gives ‘7.3’. The page reference is obvious while the result of the ‘ordinary’ reference shows up to be the identifier of the most recent  $\text{\LaTeX}$  entity (in the present case a subsection).<sup>24</sup>

<sup>24</sup>Automatically generated labels are discussed in Section 5.4 on page 25.

label names

The production and recognition of unique label names is not easy. Therefore, most of the labelnames have the structure  $\langle prefix \rangle : \langle name \rangle$ , like in `ssec:cross`. The applied prefixes are `pt:`, `ch:`, `sec:`, `ssec:`, `par:`, `fig:`, `tab:`, `box:`, `fn:`, and `eq:`. Even with these prefixes, the production of unique labelnames is difficult. Therefore, the name part of the labelname is often prefixed with a chapter identifier so that the standard labelname is  $\langle prefix \rangle : \langle chapter-id \rangle - \langle name \rangle$ . In the present document we could have used a  $\langle section-id \rangle$  by calling the section `sec:ref` and the present subsection something like `ssec:ref-cross`. Footnotes are fully recognised members of this reference system: if a label like `fn:languages-glossary` is placed in a footnote, we can refer to it by number (footnote 12) and/or by page (a footnote on page 12).

In a complex book it is useful often to refer both to an item number (e.g. a figure) and the page on which it is placed (e.g. to Section 6.3 on page 32 and footnote 12 on page 19). This could be obtained by the following command:<sup>25</sup>

```
\eref 282 \newcommand{\eref}[1]{\ref{#1} on page~\pageref{#1}}
```

\ref However, specialised packages provides much more control over cross-referencing:

\pageref

\vref

\nameref

Section 6.3

The section on page 32

Section 6.3 on page 32

The section on Graphics and figures

1 Section~\ref{sec:graphics} \\

2 The section on page~\pageref{sec:graphics} \\

3 Section~\vref{sec:graphics} \\

4 The section on \nameref{sec:graphics}

varioref.sty

The two first commands are L<sup>A</sup>T<sub>E</sub>X standards but other commands require the packages `varioref` and `nameref` (see Mittelbach and Goossens, 2004, 68–77):

nameref.sty

```
283 \usepackage{nameref,varioref}
```

showkeys.sty

During the development of the system of cross-references it is useful to visualise both labels and references in the documents. In the relevant documents, this feature is provided by loading the `showkeys` package (see Mittelbach and Goossens, 2004, 68):

```
\usepackage[notref,notcite]{showkeys}
```

Here we have disabled reference indicators—but if we exclude the options, these indicators will also be shown.

#### 7.4. Hyperref for pdf files

hyperref.sty

Hyperlinks: Since hyperlinks have a huge set of users that largely perform at least a preliminary inspection of documents on their computer screen, the precise functioning of the on-line document is worth quite some considerations. The attempts of improving the web document are rather well supported by `hyperref` package. The consequences of the use this package is especially obvious in the present document's Table of Contents (on page 1) as well as in the References (on page 46). The function of `hyperref` is to add hypertext links to all the L<sup>A</sup>T<sub>E</sub>X cross-referencing commands (including the table of contents, bibliographies, references to floats, etc.) and to allow the user to write ad hoc hyperlinks, including those to external documents and URLs. These and other possibilities are described by Goossens and Rahtz (1999, 35–67), Kopka and Daly (2004, 251–61), and at [www.tug.org/applications/hyperref/manual.html](http://www.tug.org/applications/hyperref/manual.html).

More specifically, the function of `hyperref` is to change the commands of other packages. Therefore, we load `hyperref` as the last of the major text-oriented packages. Furthermore, we need to consider whether our documents will converted directly into pdf by

<sup>25</sup>This command is still used in some of the documents, although `\vref` is smarter.

pdf $\TeX$  or whether a `dvi` file is an intermediate output—in the ordinary  $\TeX$  manner. All `hyperref` commands can be used by pdf $\TeX$  while many of them causes ordinary ‘drivers’ to stop during the processing of the document. Since we sometimes need these drivers for the processing of PostScript-oriented documents, the `esameta` package exists in two versions: `esameta.sty` and `esametaps.sty`.

The call of `hyperref` for ps-based solutions uses two options. The first is to use a `dvips` driver that starts by producing a `dvi` file that is converted to a `ps` file (which in turn can be converted to a `pdf` file). The second option is to include into the bibliography hyperlinks back to the places where the references were cited. More precisely, `hyperref` uses `natbib`’s transformation of the list of references into a kind of index of citations. It simply transform the listed page numbers into hyperlinks.

dvips  
pagebackref

```
284 \usepackage{esametaps}
285 \usepackage[dvips,pagebackref=true]{hyperref}
286 \end{esametaps}
```

docstrip command

The only novel thing in this call is that we now really use the `docstrip` environment that causes conditional inclusion of the code in the style files. As already mentioned (see page 8), the beginning of this environment is actually written as `\usepackage{esametaps}`!. It is a command to the `docstrip` program to place this code in `esametaps.sty` and not `(&!)` in `esameta.sty`.

pdftex  
pdfpagelabels

The call of `hyperref` for pdf-based solutions also includes `pagebackref` (which apparently cannot be set by a later command). Furthermore, the `pdftex` driver is used. Finally, the option `pdfpagelabels` causes page references to be formatted as in the document. This is necessary in books that first use Roman and then Arabic page numbers.

```
287 \usepackage{esametaps}
288 \usepackage[pdftex,pdfpagelabels,pagebackref=true]{hyperref}
289 \end{esametaps}
```

hycap.sty

With respect to floats, `hyperref` makes links to the caption. Since this is placed below figures and many tables, the jump does not necessarily display their contents. The `hycap` package is designed to resolve this problem by defining the command `\capstart` in the beginning of such environments. .

```
290 \usepackage{hycap}
```

\hypersetup

**Hypersetup:** `Hyperref` sets a large number of options, partly after recognising the overall output mode of the document. The options (largely Booleans) can either be reset in the package call or in set-up commands `\hypersetup{<list of options>}`. Even if options have been set, they can be overwritten in later set-up calls. We, however, need to recognise that only a subset of these options are applicable for all drivers. Let us start with some of the general options.

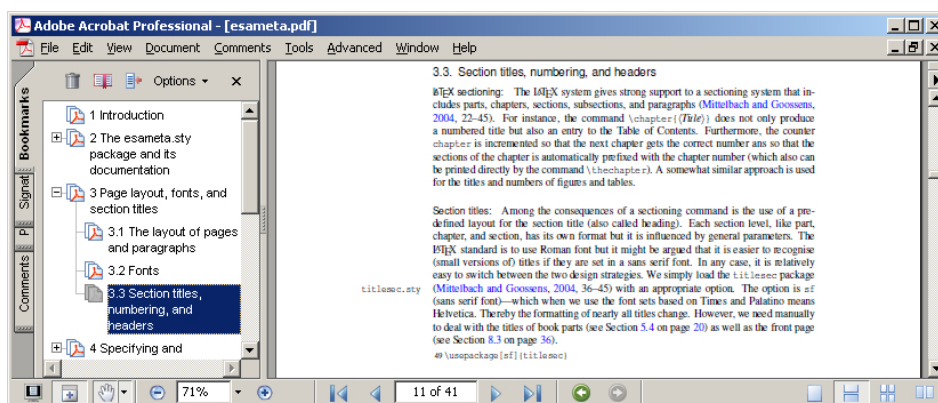
The standard set-up for hyperlinks is rather ugly-looking. For instance, `colorlinks=false` produces a frame around each hyperlink and the colouring is complex. Therefore, we use `\hypersetup` to add ordinarily coloured links and give them a uniform colour:

```
291 \hypersetup{colorlinks=true, linkcolor=blue, anchorcolor=blue,%
292 citecolor=blue, filecolor=blue,menucolor=blue, pagecolor=blue,%
293 urlcolor=blue}
```

url.sty

By the way, the correct breaking of URLs is a special problem that is partly taken care of by the `url` package:

```
294 \usepackage{url}
```

Figure 10: The `esameta.sty` output in Adobe Acrobat Reader

The setting of the `hyperref` options serve to create a very efficient system of referencing for on-line pdf documents. However, for the fine tuning of this system, a few additions were made in Section 6.1 on page 28, Section 7.1 on page 33, and Section 7.2 on page 34. Among the remaining problems are that `hyperref` does not immediately recognise the two paginations found in books (Roman numerals in the front matter and then Arabic numerals). This problem is resolved by setting the following option.

```
295 \hypersetup{plainpages=false}
```

A minor remaining page-reference problem arises from the unnumbered title page in the `report` class (created by the `\maketitle` command). In `hyperref`, this page is numbered by ‘1’, just like the following page. Therefore, we get the pdfTeX warning of ‘duplicate destinations’, and the hyperlinks that should point to the first page of the main matter of the book actually point to the title page.

The use of `\hypersetup` is not restricted to package files. On the contrary, we may set or reset most of them in the preamble of individual documents. For instance, the preamble for the printing of the document without hyperlinks would include `draft` (while `final`, somewhat confusingly, turns the hyperlinks on).

**Pdf viewer issues:** The production of `hyperref` documents includes considerations on the control over e.g. Acrobat Reader’s display the pdf file and the ‘document property’ information. This possibility is only available in documents that are produced with the pdfTeX program. Therefore, we exclude the definitions from the `esametaps.sty` version of the metapackage. Here are the presently applied options in `esameta.sty`:

```
296 (*esameta&!esametaps)
297 \hypersetup{%
298   pdfpagelayout=OneColumn, pdfstartview=FitH, pdfview=FitH,%
299   pdfpagemode=UseOutlines, bookmarksnumbered=true,%
300   pdfauthor={Esben Sloth Andersen}}
301 \end{esameta&!esametaps}
```

These options are largely selected for the readability and navigability of our on-line documents. Such a pdf file opens with the table of contents as ‘bookmarks’ in the navigation panel while the document is displayed in full width (see Figure 10). We can also navigate through the document by using the many internal hyperlinks that `hyperref` has created.

The options set by the *personal* metapackage also include the normally used name for the author. However, we can overwrite it in the preamble of the document. Here we also need to specify the title of the document for Acrobat Reader's 'Document Properties'.

```
\hypersetup{draft, pdftitle={Title of the Document}}
```

### 7.5. The postponed definitions of textbox

As already mentioned, `hyperref` redefines the commands of other packages. Among them are `float`. However, the `\newfloat` command must use `hyperref`'s redefined commands. Therefore, the already discussed definition of the `textbox` environment (see `textbox` Section 6.1 on page 28) is placed after the loading of `hyperref`:

```
302 \floatstyle{ruled}
303 \newfloat{Box}{t}{lob}[\@ifundefined{chapter}{section}{chapter}]
304 \newenvironment{textbox}[2]%
305   {\begin{Box}[t]%
306    \caption{#1}%
307    \centering\vspace{6pt}\small%
308    \begin{minipage}{0.95\textwidth}#2%
309    \end{minipage}%
310    \vspace{4pt}}%
311   {\end{Box}}
```

Another postponed issue is the integration of text boxes into the overall List of Boxes, Figures, and Tables (see the paragraph on the topic on page 27). Since the command `\ext@Box{lof}` is now defined, we use it. Furthermore, we define the way in which text boxes are recorded in the integrated list.

```
312 \renewcommand{\ext@Box}{lot}
313 \newcommand{\l@Box}[2]{%
314   \@dottedtocline{1}{0em}{3.15em}{Box~#1}{#2}}
```

## 8. Using esameta.sty

### 8.1. The system of L<sup>A</sup>T<sub>E</sub>X-related files

The efficient use of `esameta.sty` presupposes a well-designed system of files. The core of this file system is defined by the Comprehensive T<sub>E</sub>X Archive Network (CTAN) and by the applied L<sup>A</sup>T<sub>E</sub>X distribution (e.g. MikT<sub>E</sub>X). The T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X files are found in the directory called `/texmf/` and especially in `/texmf/tex/latex/` and `/texmf/doc/latex/`. The updating of these files should largely be done automatically (when we require a package that has not yet been downloaded). However, we also need to define a local system for L<sup>A</sup>T<sub>E</sub>X-related files. This system should be as simple as possible with respect to search for files, backup, cross-referencing, etc. Figure 11 on the following page describes the basics of such a system. This system provides the foundations for the use of relative links in `esameta.sty`. Thus, the command `\bibliography{../biblio/esa}` will refer to the same file in all L<sup>A</sup>T<sub>E</sub>X-based papers and books.<sup>26</sup>

Figure 11 has been developed by means of the `parallel` package and the related `\LR` command (Mittelbach and Goossens, 2004, 181–4). It has problems with the alignment

<sup>26</sup>There are smarter ways of securing common use of files—but the selected one has the advantage of visuality and simplicity.

---

/esalatem/	The local L <sup>A</sup> T <sub>E</sub> X and document files
/styleslocal/	Files relating to esameta.sty
/biblio/	BIB <sub>T</sub> E <sub>X</sub> related files
esa.bib	The BIB <sub>T</sub> E <sub>X</sub> database
/schumpmacros/	Commands for concrete indexing, etc.
namesindexcmd.tex	A set of L <sup>A</sup> T <sub>E</sub> X macros that produce entries to the name index
worksindexcmd.tex	A set of macros that format citations of Schumpeter's works and make index entries
/images/	Pdf versions of images and drawings as well as the files that define them
/figures/	Files that define figures; just as tables and boxes, they are collected in a common folder to ease standardisation and reuse
/tables/	Files that define tables
/boxes/	Files that define text boxes
/booklarge/	The file system for the large book
/booksmall/	The file system for the smaller book
/paper1/	Here is a paper
/paper2/	Here is another paper; each paper that uses (a modification of) esameta.sty is placed at the same level as the book to simplify access to common resources
/organisation/	Administrative issues relating to the books and the papers
/latexliterature/	A set of pdf and dvi files relating to esameta.sty

---

Figure 11: The local system of L<sup>A</sup>T<sub>E</sub>X files

of the texts in the left and right columns. The `parallel` package is actually designed for the two-column display of language translations so it might also be used for its real purpose. However, the table of translations for the Schumpeter books is designed in a different way (see Section 4.3 on page 18).

```

315 \usepackage{parallel}
316 \newcommand{\LR}[2]%
317   {\ParallelLText{\texttt{#1}}}%
318   \ParallelRText{#2}\ParallelPar}

```

## 8.2. Setting up of documents

To apply `esameta.sty`, we need to call it in the beginning of each of our L<sup>A</sup>T<sub>E</sub>X documents.<sup>27</sup> However, since some decisions have been postponed, we need to include them in the preamble. There are two major kinds of command:

- General commands that at least cover the two Schumpeter books (and most of the different documents that relates to them). These commands are placed in the `esametax.tex` file.
- Document-specific commands are placed in the preambles of the respective `tex` files.

The general decisions that could not be taken in the present document includes page set-up (Section 3.1 on page 10), the use of `array.sty` (Section 6.2 on page 29), and the verbatim commands (Section 2.4 on page 8). We include these commands in the small `esametax.tex` file:

`esametax.tex`

---

<sup>27</sup>Including the present document.

```

\esapagelayout
\usepackage{array}
\esaverbatim

```

Given this file (which may be empty), a document can be set up in the following way:

```

\documentclass[a4paper]{report}
\usepackage{../styleslocal/esameta}
\input{../styleslocal/esametax}
\esalargebook
\setboolean{PrintBook}{false}
\hypersetup{pdftitle={Schumpeter's Evolution}}
%\hypersetup{draft} %Removes hyperlinks from output
%\esaglossortworks %For changing sorting of the translations table
%\usepackage[notref,notcite]{showkeys} %Showing labels for reference
\sloppy %To avoid too much fine-tuning

\begin{document}
...
\end{document}

```

Thus, the document class definition (e.g. `report.cls`), the `esameta.sty`, and the extras in `esametax.tex` define the core facilities for the document code. However, additional commands are needed. Some of these commands are ‘commented out’ by the `%` character (but they are ready for quick inclusion in the definition of the document). The real comments follow the `%` that comes after a command. Some of the relevant commands are:

`\esalargebook`: This command and its twin (`\esasmallerbook`) make the selection of fonts discussed in Section 3.2 on page 13. The alternatives are `\esatimes` and `\esapalatino`. The commands also influence the formatting of the headers.

`PrintReport`: This Boolean variable is solely used for the correct formatting of headers in report and book.

`\hypersetup`: We have already defined many options for the `hyperref` package (see Section 7.4 on page 36). The document-specific options might include the document title (for the information in Adobe Reader’s Document Properties) and the hiding of the hyperlinks by the `draft` option.

`\esaglossortworks`: A command that changes the sorting of the table of translations (see Section 4.3 on page 18). Even if this option is not chosen, it is important to remember to run the document with both `LaTeX=>PDF` and `LaTeX=>PDF(trans/glos)`.

`\usepackage{showkeys}`: If the package is loaded, the labels for cross-references will be printed in the margin of the document (see Section 7.3 on page 35).

`\sloppy`: This command serves to avoid too much attention to the fine-tuning of hyphenation and similar issues (see Section 3.1 on page 10).

The real document is placed between `\begin{document}` and `\end{document}`. This part is split up into files for the individual chapters (each in their own folder), and these files are included by means of the `\input` command. The master file exists in two versions: one for the production of the whole book and one for individual chapters. The production of partial documents is described in a paragraph on that subject on page 43—so here we should just note that this production implies the commenting out of nearly

all the other input files. For instance, we may produce the chapter on ‘Social Evolution’ (`/social/social.tex`) by the following commands:

```
\documentclass[a4paper]{report}
\usepackage{../styleslocal/esameta}
...
\setboolean{PrintChapter}{true} %Produces a front page
...
\begin{document}
%...
%\input{cycles/cycles}
\input{social/social}
%\input{science/science}
%...
\input{ref/schumpworks}
\input{ref/otherworks}
\input{index/index}
\end{document}
```

In conclusion, we may note that the use of ‘literate programming’ in the form of a `dtx` file—that produces both the present document and the `esameta.sty`—has introduced some complexities. Thus, the efficient use of the document-specific commands presupposes document templates or the copying of commands from old to new documents. This requirement, however, is compensated by the possibility of easy and consistent development of the personal metapackage.

### 8.3. Selecting output formatting

As just demonstrated, the Schumpeter books and their parts can be formatted in a large number of ways. During the development of the manuscripts, the applied document class is `report` but occasionally the `book` document class will be used. The real problem is how to modify these classes when taken into consideration both issues of on-screen viewing and printing.

**Fonts:** For viewing on a computer screen, the font Computer Modern (CM) is readable even when the text is scaled down. However, CM makes the print files large and printing slow, so we turn to Times or Palatino (that are also acceptable on screen).

- The large book: Considerations of size and speed are especially important with respect to the large Schumpeter book. Therefore, Times is an appropriate choice (either through the `pslatex` package or `\esatimes`).
- The smaller book: The book for the Palgrave series emphasise reader-friendliness so here the larger and more fancy Palatino solution seems better. The use of `\esapalatino` also serves to differentiate the two manuscripts.

**Hyperlinks:** While hyperlinks are very helpful for on-line viewing, they make the reading of the printed output difficult (hyperlinks become grey or coloured). Furthermore, hyperlinks add considerably to the size of the print file. Therefore, we shall maintain two sets of downloadable files: one for on-line viewing and one for printing.

**Partial documents:** The availability of individual chapters (with their own index and list of references) is important for obtaining feedback from some of commenting experts. To systematise the production of stand-alone chapters, we maintain versions of the book master documents in which all chapters except one are commented out.

The `ifthen` package is used for distinguishing between the printing of the full manuscript and the individual chapters by the Boolean variable `\boolean{PrintChapter}`. The command that uses this variable is `\esachapfrontpage`. This command calls the title page for chapters that is contained in the file `schumpchapfront.tex`. This file's definition the front page produces the chapter title by means of the contents of the command `\esachaptitle`, which should be redefined in the beginning of each chapter.<sup>28</sup> We also add Boolean variables for separate printing of 'Translations', 'Schumpeter's Works', 'Other References', and 'Index'.

```
PrintChapter
\esachapfrontpage
schumpchapfront.tex
\esachaptitle
```

```
319 \newboolean{PrintChapter}
320 \setboolean{PrintChapter}{false}
321 \newcommand{\esachaptitle}{Insert the chapter title}
322 \newcommand{\esachapfrontpage}%
323   {\ifthenelse{\boolean{PrintChapter}}%
324     {\input{schumpchapfront}}{}}
325 \newboolean{PrintTranslations}\setboolean{PrintTranslations}{false}
326 \newboolean{PrintSchumpWorks}\setboolean{PrintSchumpWorks}{false}
327 \newboolean{PrintOtherWorks}\setboolean{PrintOtherWorks}{false}
328 \newboolean{PrintSchumpIndex}\setboolean{PrintSchumpIndex}{false}
```

To make this system functional, we need to insert something like the following commands in the beginning of each chapter and appendix:

```
\renewcommand\esachaptitle{Introduction}
\esachapfrontpage
\chapter{\esachaptitle}
\label{ch:intro}
```

**Notes:** As mentioned in Section 5.1 on page 21, footnotes are a dominant feature of the large book. On the other hand, they are turned off in the smaller Schumpeter book by the command `\hidefootnotes` that at the same time loads the `endnotes` package. In both book manuscripts there are also a need of turning off the marginal notes (that are largely addressed to the author). They are produced by the command `\esamarpar` and turned off by `\hideesamarpar`.

**The list of translations:** A decision made at when the pdf file is produced concerns the sorting of the table of translations (see Section 4.3 on page 18). If we want to check the translations, the best sequence is that of the book. If we check the foreign-language quotes with the text of the originals, sorting should be according to the quoted works.

**Aggregating decisions:** To simplify switching between the two book projects, we introduce two summarising commands. The command `\esalargebook` implies a style that is rather close to that of the present document, but it omits the ruler beneath the header. The command `\esasmallerbook` applies the Palatino font combination and includes commands concerning endnotes (see Sections 3.2 and 5.1).

```
\esalargebook
\esasmallerbook
```

```
329 \newcommand{\esalargebook}{
```

<sup>28</sup>The file `schumpchapfront.tex` ends by setting the page counter to 2 in order to avoid the `hyperref` problem with references to page 1 in reports with title pages.

```
330 \usepackage{pslatex}
331 \renewcommand{\headrulewidth}{0pt}}
332 \newcommand{\esasmallerbook}{%
333 \esapalatino}
```

#### 8.4. Exploiting error messages and warnings

The production of  $\LaTeX$  output will normally require the correction of errors of different types. Therefore, an Integrated Development Environment like  $\TeX$ nicCenter includes a window that provides information about the many steps that are performed by the underlying  $\LaTeX$  engine (Mik $\TeX$ ) during the compilation of the document—and about the errors and problems that occur in some of these steps (Mittelbach and Goossens, 2004, 889–946). The error messages and warnings are located to numbered lines in the processed files, and there are easy ways of jumping to the erroneous or problematic lines in the `tex` and `sty` files. More extensive information is found in the `log` file that includes all the information on errors, warnings, badly formatted paragraphs (‘boxes’), etc. that  $\LaTeX$  directly and indirectly produces during its processing of the `tex` file.

The basic rule for successful production of  $\LaTeX$  documents is that they have to be developed incrementally. The reason is that  $\LaTeX$  has a huge number of available commands and the misspelling of a single of them can hinder the system in producing any output. Therefore, it is a hopeless strategy to produce a document with a large number of commands before trying out the output. Instead, we in  $\TeX$ nicCenter very frequently hit the `F7` key when working in  $\TeX$ nicCenter. This will save the file and build (but not show) the output. During the process, a lot of information is shown in the bottom window. The last line that is shown in the bottom window gives a summary. This message might include errors and warnings:

```
LaTeX-Result: 5 Errors, 27 Warnings, 199 Bad Box(es), 381 Page(s)
```

Let us consider these items one by one:

**Pages:** If the message ends with `0 Page(s)`, then the  $\LaTeX$  system has for some reason been unable to produce any output and we have to look at the error messages.

**Errors:** To check the errors within  $\TeX$ nicCenter, we select the menu `Build/Next Error` (or press `F9`), read the message, and inspect the code in the corresponding line of the `tex` file. Sometimes it is easy to figure out what went wrong. It is often the last error that creates the problem.<sup>29</sup> For instance, the path to an included file may be written wrongly. Rewrite the file, and press `F7`, and repeat this process until output is produced. The functioning file often does not have to be fully debugged from ‘errors’. However, it is unwise not to correct easily removable errors since they will distract attention from the important ones.

**Warnings:** Although many of the warnings are unimportant, we want to remove as many of them as we can do easily. The reason of that we often have to browse through the warnings because some of them are important (e.g. misspecified cross-references and references to `BIB $\TeX$` ). To browse the warnings, we in  $\TeX$ nicCenter select the menu

<sup>29</sup>Occasionally, the corrections have no influence. The problem might be due to an unchanging `aux` file with wrong code. Then the solution is to delete this file.

Build/Next Error, make corrections if it is important or easily removable, and repeat this process.<sup>30</sup>

**Bad Boxes:** During the development of the manuscript we may largely ignore messages about ‘Bad Boxes’—since  $\LaTeX$  has unrealistic requirements about the quality of its layout of paragraphs, etc. To reduce the number of warnings, we include the command `\sloppy` in most documents. A few of the remaining bad-box messages might be an indication of a significant error, like forgetting to close a mathematical environment (if we start with `$,wegetunbrokenmathsuntilwefinallymeetthenext$`).

**User-oriented errors and problems:** Most of the errors and problems in a document cannot be discovered by  $\LaTeX$  because they concern spelling, punctuation, factual information, and logic. It is advisable to remove these problems immediately by inspecting the output file very frequently. Therefore, we need good integration between the  $\LaTeX$  engine (MikTeX),  $\TeX$ nicCenter, and Acrobat Reader.<sup>31</sup> This integration means that we can inspect the output file at the point where we were before by pressing `F5`—or we can combine building and viewing by `Ctrl+F5`.

**Errors and problems in the metapackage:** Difficulties might emerge in some of our documents after each change in the metapackage. To ensure an early detection of such difficulties, we try the changed version of `esameta.sty` on different types of our documents. For instance, the author experienced problems after changing the chapter command so that it could be followed by a local table of contents (see the paragraph on the topic on page 26). The most obvious problems was that the use of stars in sectioning commands did not work correctly but even after solving this problem the new feature did not work when a list of text boxes was produced. The early detection and solution of such problems require that `esameta.sty` is put on a ‘test bed’ of complex documents.

## 9. Discussion

As described in the Section 1 on page 2, the process of change of the `esameta.sty` package has some similarity with an evolutionary process with retention, innovation, and selection. The code of the metapackage is repeatedly confronted with new needs. If the new needs at time  $t$  can be satisfied by new packages and/or the the writing of new commands, and if these solutions are compatible the  $t - 1$  version of the package, then they are included into the  $t + 1$  version. If the wanted features are incompatible, a choice has to be made. We may drop the new features, make them specific to the time  $t$  project, or remove apparently less important features from the  $t - 1$  version of `esameta.sty`. In this respect, the situation for writers of *personal* metapackages is radically different from writing ordinary  $\LaTeX$  packages—for which users require that existing features more or less are ‘frozen’. The only restriction on the features of the metapackage is that we do not want to create too many difficulties for the reuse of `tex` files created with previous

<sup>30</sup>Since a complex system of referencing needs 3–4  $\LaTeX$  runs, it is wise to ignore the first count of e.g. 2,000 Warnings that emerge the first time we compile after an unsuccessful run.

<sup>31</sup>To use  $\TeX$ nicCenter efficiently, we need good integration with Acrobat Reader. In the Define Output Profiles, go to the Viewer of LaTeX => PDF and define the following DDE commands:  
`[DocOpen("bm.pdf")] [FileOpen("bm.pdf")] [MenuItemExecute("GoBack")],`  
`[DocOpen("bm.pdf")] [FileOpen("bm.pdf")] [MenuItemExecute("GoBack")], and`  
`[DocClose("bm.pdf")] (all with the options acroview and control).`

versions of `esameta.sty`. Here ‘too many difficulties’ is a very flexible notion. Furthermore, basic compatibility problems might be revealed later so it would be nice to be able to revert to a previous stage of the ‘evolutionary’ process. This is actually possible in Version Control Systems (in which we can restore the state of a document to any previous point of time)—but we abstain from this solution. The reason is that the irreversibility of the change process serves to enforce a consistent set of  $\LaTeX$  documents (through the rewrite of old documents).

The emphasis on consistency of the set of documents related to the  $\LaTeX$  metapackage means that its documentation needs little emphasis on the (evolutionary) history of changes. Instead, this documentation largely describes ‘the state of the art’. However, this state clearly reflects the history of the metapackage. For instance, we have seen that the history is heavily influenced by the double nature of the output produced with the help of the metapackage: on-line documents with hyperlinks and printer-friendly pdf files. As a consequence, important  $\LaTeX$  packages that exploits the PostScript format have been included in an indirect way.

The application of ‘literate programming’ needs to be subjected to Cost–Benefit Analysis. The costs include the extra time spent on documenting new features as soon as they are introduced in the metapackage. The benefits include (1) the reduced working time needed for using a previously introduced feature that has largely been forgotten; (2) the ease of detecting the consequences of removing a feature which is incompatible with new features; (3) the increased ease of informing interested students about  $\LaTeX$  features. Although the contents of the present document might suggest that too much time has been spent on ‘literate programming’, its author is convinced that the benefits are significantly larger than the costs.

## Bibliography

- Andersen, Esben Sloth (1994): *Evolutionary Economics: Post-Schumpeterian Contributions*, London: Pinter. See page(s) 2
- Andersen, Esben Sloth (1997): Elements of  $\LaTeX$  Systems, Working Paper, Department of Business Studies, Aalborg University. See page(s) 2
- Andersen, Esben Sloth (2001a): “Satiation in an evolutionary model of structural economic dynamics,” *Journal of Evolutionary Economics*, **11**(1): 143–164. See page(s) 2
- Andersen, Esben Sloth (2001b): Toward a Multiactivity Generalisation of the Nelson–Winter Model, Working Paper, Paper presented at DRUID’s Nelson and Winter Conference, Aalborg, 12–15 June 2001. URL [www.business.auc.dk/evolution/esapapers/esa01/andersen.pdf](http://www.business.auc.dk/evolution/esapapers/esa01/andersen.pdf), See page(s) 2
- Andersen, Esben Sloth (2006a): Elements of the  $\LaTeX$  system for the development of large publications, Working Paper, Department of Business Studies, Aalborg University. URL [www.business.aau.dk/evolution/esapapers/esa06/MyLaTeX.pdf](http://www.business.aau.dk/evolution/esapapers/esa06/MyLaTeX.pdf), See page(s) 2, 4, 20, 35
- Andersen, Esben Sloth (2006b): The evolution and use of a personal  $\LaTeX$  metapackage: Appendix on PostScript graphics, Working Paper, Department of Business Studies, Aalborg University. URL [www.business.aau.dk/evolution/esapapers/esa06/esametagraph.pdf](http://www.business.aau.dk/evolution/esapapers/esa06/esametagraph.pdf), See page(s) 31, 32
- Andersen, Esben Sloth (2006c): Schumpeter’s Engine of Evolution: Emergence and Structure of a Dramatic Theory of Social and Economic Evolution, Manuscript for a large book, Department of Business Studies, Aalborg University. See page(s) 3

- Date, C. J. (2004): *An Introduction to Database Systems*, 8th edn, Boston, Mass.: Addison–Wesley. See page(s) 4
- Goossens, Michel and Rahtz, Sebastian (1999): *The L<sup>A</sup>T<sub>E</sub>X Web Companion: Integrating T<sub>E</sub>X, HTML, and XML*, Reading, Mass.: Addison–Wesley. See page(s) 36
- Goossens, Michel, Mittelbach, Frank, and Samarin, Alexander (1994): *The L<sup>A</sup>T<sub>E</sub>X Companion*, Reading, Mass.: Addison–Wesley. See page(s) 9
- Goossens, Michel, Rahtz, Sebastian, and Mittelbach, Frank (1997): *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion: Illustrating Documents with T<sub>E</sub>X and PostScript*, Reading, Mass.: Addison–Wesley. See page(s) 32
- Grätzer, George (1996): *Math into L<sup>A</sup>T<sub>E</sub>X: An Introduction to L<sup>A</sup>T<sub>E</sub>X and A<sub>M</sub>S-L<sup>A</sup>T<sub>E</sub>X*, Boston: Birkhäuser. See page(s) 20
- Haralambous, Yannis (1991): “Typesetting old German: Fraktur, Schwabacher, Gotisch and initials,” *TUGboat*, **12**(1): 129–138. See page(s) 14
- Knuth, Donald E. (1990): *The T<sub>E</sub>X Book*, Computers and Typesetting, Vol A, 2nd edn, Reading, Mass.: Addison–Wesley. See page(s) 4, 8
- Knuth, Donald E. (1992): *Literate Programming*, Stanford: Center for the Study of Language and Communication. See page(s) 5, 7
- Knuth, Donald E. (1997): *The Art of Computer Programming*, 3 vols, 2nd–3rd edn, Reading, Mass.: Addison–Wesley. See page(s) 3
- Kopka, Helmut and Daly, Patrick W. (2004): *Guide to L<sup>A</sup>T<sub>E</sub>X*, 4th edn, Boston, Mass.: Addison–Wesley. See page(s) 2, 14, 32, 36
- Lamport, Leslie (1994): *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*, 2nd edn, Reading, Mass.: Addison–Wesley. See page(s) 8
- Mittelbach, Frank and Goossens, Michel (2004): *The L<sup>A</sup>T<sub>E</sub>X Companion*, 2nd edn, Boston, Mass.: Addison–Wesley. See page(s) 4, 5, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 39, 44
- Stroustrup, Bjarne (1994): *The Design and Evolution of C++*, Reading, Mass.: Addison–Wesley. See page(s) 3
- Talbot, Nicola L.C. (2006): glossary.sty v 2.4: L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> package to assist generating glossaries, Web Document, School of Computing Sciences, University of East Anglia. URL [tug.ctan.org/tex-archive/macros/latex/contrib/glossary/glossary.pdf](http://tug.ctan.org/tex-archive/macros/latex/contrib/glossary/glossary.pdf), See page(s) 18
- Wilson, Peter R. (2004): The layouts package: user manual, Web Document, Herries Press. See page(s) 10