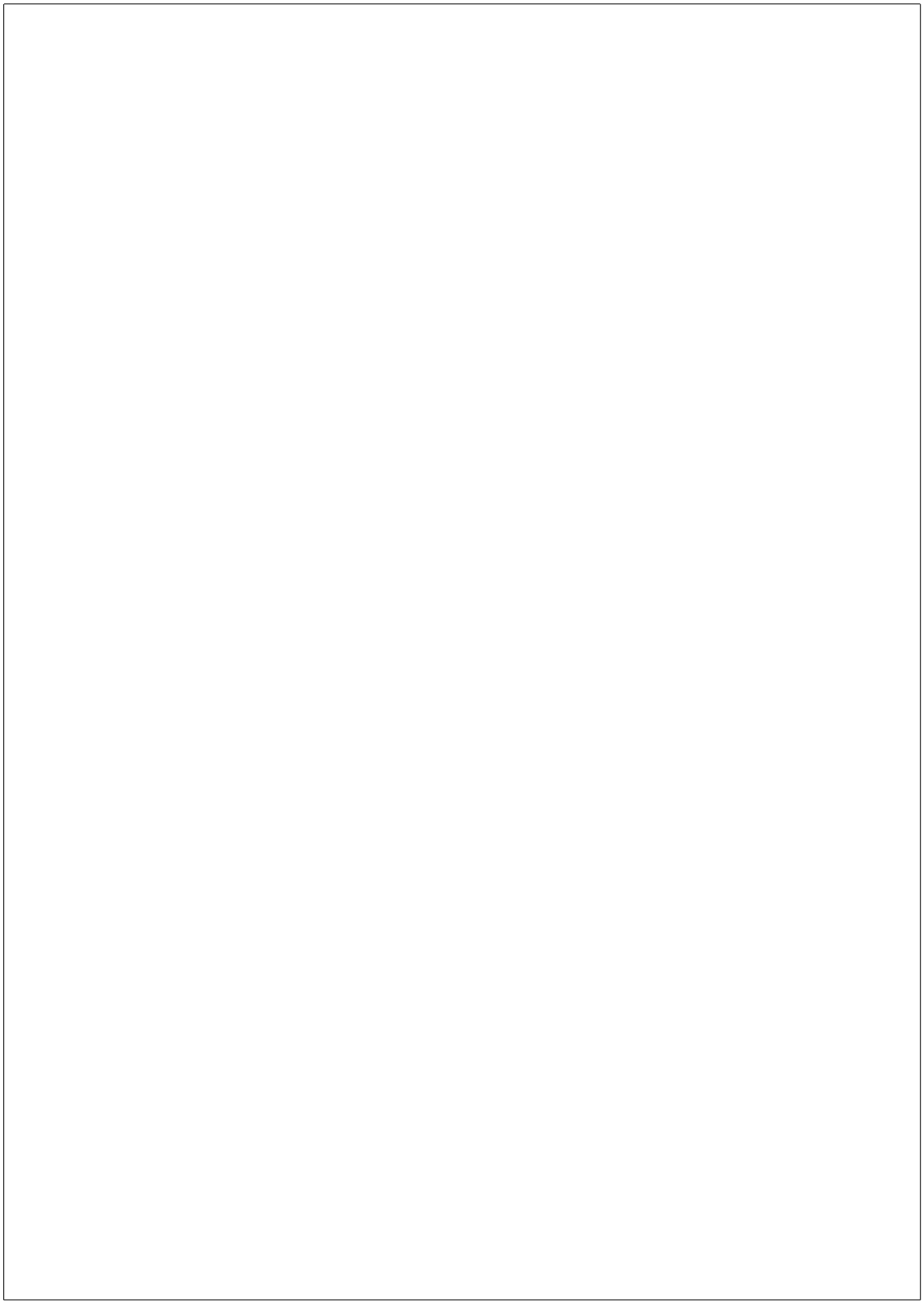
Manual for the `latex-maven-plugin` and for an according ant-task Version 1.5

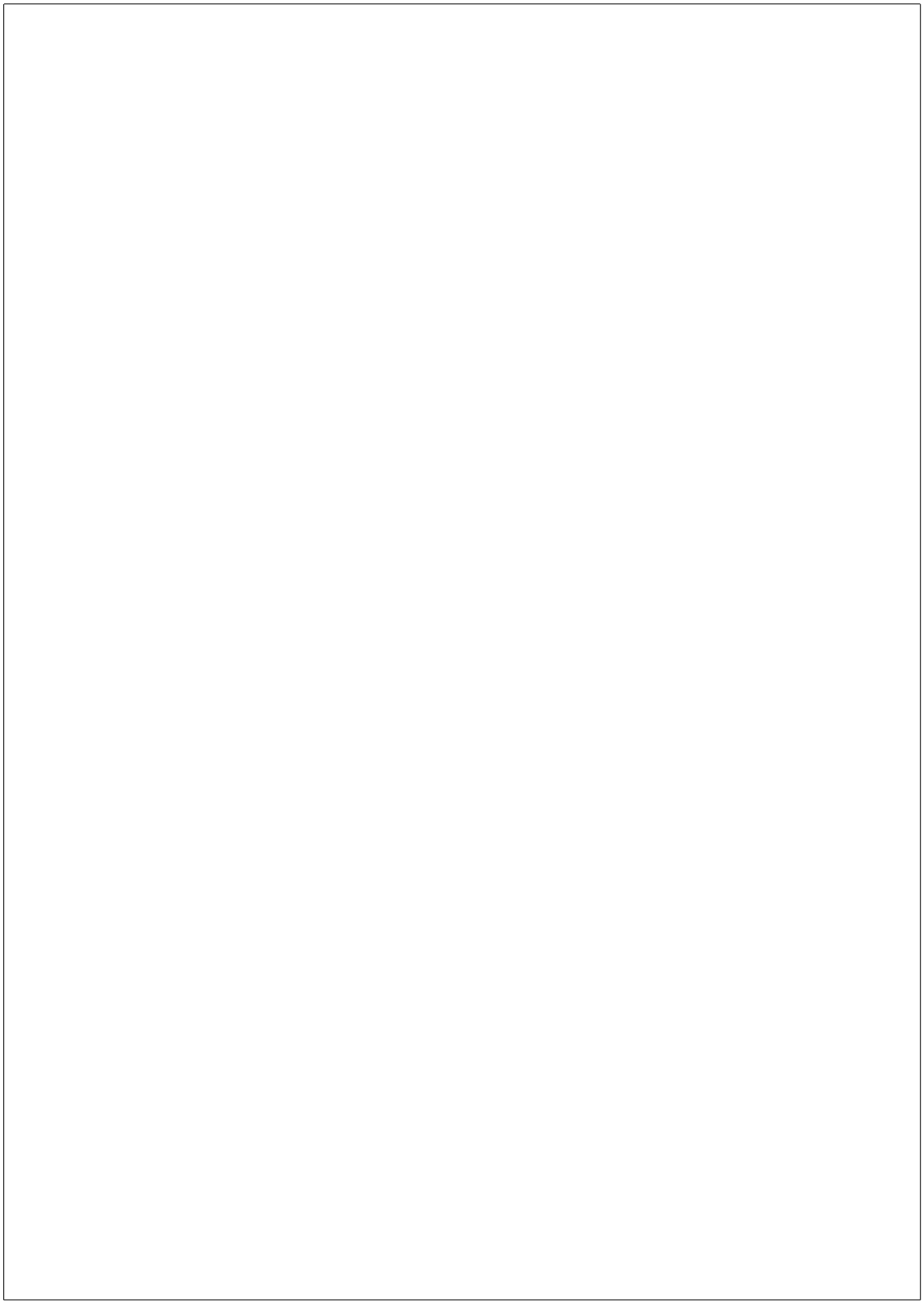
Ernst Reissner (rei3ner@arcor.de)

2022-02-03



Contents

List of Figures



List of Tables

List of Listings

Chapter 1

Introduction

This document is created with `pdflatex` or that like with output format pdf. The package `tex4ht` is not loaded.

LaTeX is a beautiful way to create printable documents, in our days preferably as portable document format (pdf)-files, with a particular strength in typesetting formulae like

$$\pi = \sqrt{12} \sum_{k=0}^{\infty} \frac{(-3)^{-k}}{2k+1}. \tag{1.1}$$

Here, portability of the format pdf is a vital feature. In the past, normally device independent file format (dvi) (device independent) described in [ner17] has been used and still creation of external formats like hypertext markup language (html), open document text (odt) and current document format for MS word (docx) are based on an intermediate dvi-file. It is much more lightweight than pdf specified in [ISOa], [ISOb] and in [ISO12].

This piece of software implements both an ant-task and a maven-plugin generating documentation of various formats from LaTeX-files in an uniform way. Chapter 2 shows how to install both the maven-plugin and the ant-task and Chapter 3 describes the usage. Note that the maven-plugin is both easier to install and more versatile to be used.

From the LaTeX-files, the latex main files must be extracted, only these must be compiled. It is very usual to endow LaTeX-files with figures. On the other hand, there are many graphic formats which cannot be included directly in a LaTeX-file but must be preprocessed. If there is some format needed but not yet provided, please write an email to the author.

Graphic files must be preprocessed before processing latex main files, as described in Chapter 4. Then follows the proper processing of latex main files including creation of index and glossaries as described in Chapter 5. Besides pdf, these

formats include the web-formats html and extensible hypertext markup language (xhtml), open offices format odt, Microsoft's word formats like docx and finally plain text.

Uniformity of ant-task and a maven-plugin means in particular, that the settings which may be passed to the task and those allowed for the plugin are in a one-to-one relation. They are both described in Chapter 6. It is a design goal, that the auxiliary programs used by this software are fully configurable via parameters, that aspects not completely specified can be handled flexibly, there are parameters supporting information development and that for the parameters are default values which allow to do without explicit parametrization in most of the cases. Both, the ant-task and the maven-plugin rely on the same code base which form the package `org.m2latex.core`. The code specific for the ant-task is in `org.m2latex.antTask` and that specific for the maven-plugin is in `org.m2latex.mojo`.

The creation process supports an index, a glossary and a bibliography. Again, further functionality can be added by demand.

The present manual is created by the maven-plugin or the ant-task described here. There should be no difference in the result. This manual is designed in a way that it covers the most important features but also to demand the most important features. That way, creating this manual is a top level test for the underlying software. The maven-plugin is somehow superior because it better supports the design process for the LaTeX sources.

If something goes wrong in the build process, or there is an indication of some deficiency in the result of the build process, processing must be aborted if going on does not make sense and there must be some error or warning logging as described in Chapter 7.

The configuration of the maven plugin and of the ant-task are given in Chapter 8 in Listing 8.2 and in Listing 8.3, respectively.

The author found some gaps, i.e. desirable features which are not yet implemented. To prioritize further work, all these gaps are collected in Chapter 9. Accordingly, the most important bugs are collected in Chapter 10. The user is encouraged to contribute with feature requests and bug reports and to vote for realization of features and on fixing bugs. Software quality is ensured mainly through tests which are described in Chapter 11.

Chapter 2

Installation

Both the ant-task and the maven-plugin just direct parameters from ant and from maven, respectively, to the programs that do the proper work. Thus installation of the ant-task and of the maven-plugin requires that all needed programs are installed. These prerequisites are collected in Section 2.1.

2.1 Prerequisites

The ant-task is tested with

```
Apache Ant(TM) version 1.10.10 compiled on December 22 1969}
```

(of course the year is not correct but this is the version string displayed by that release) and the maven-plugin with

```
Apache Maven 3.8.1
```

Both, ant and maven are written in java and require a java installation. The java version used for tests is `11.0.13, vendor: Oracle Corporation` but java 8 seems sufficient.

So, a java installation is the base for running either the ant-task or the maven-plugin. Also this plugin is written in java. To use the maven-plugin, of course maven must be installed and to use the ant-task, ant must be installed.

The ant-task just passes parameters in the build file to the core and accordingly the maven-plugin passes parameters in the pom to the core of this software. The core just invokes various programs to do the actual work.

Besides plain building of documentation, this software also supports development of documents. LaTeX and related programs are based on text files mainly and so a good editor is required for development. The author recommends and uses good old `GNU Emacs 24.3.1 (x86_64-suse-linux-gnu, GTK+ Version`

3.16.7) together with several packages to support various file formats. To list the available packages type `M-x list-packages`. For comfortable development with LaTeX, the `auctex` package, version `11.88` is recommended. The version is displayed from within Emacs by typing `C-h v AUCTeX-version RET`. For an overview on `auctex` see [TAK+14].

FIXME: gnuplot-mode expects file extension gp. Should be made configurable.

To edit metapost, the mode built-in mode `Metamode` is used.

Built-in mode `Docview` to view pdf, ps and dvi.

latexmk

Builtin modes bib-mode and bibtex

Built in reftex-modes

Useful: ac-math, auto-complete-auctex

Depending on what kinds of graphic formats are used, the following programs are required:

- To convert the native file format for xfig (fig)-files into pdf-files, by default `fig2dev` is used. It makes sense to have `xfig` installed to create and edit fig-files, but this is not mandatory.
- To convert gnuplot files into pdf-files, there is no alternative, to have installed `gnuplot`. It serves as an interpreter and also as a converter. Strictly speaking, only the latter functionality is required here.
- To convert metapost (mp)-files into encapsulated postscript (eps)-files, the interpreter `mpost` or equivalent is required. This comes with a standard tex-installation. With the standard configuration, the resulting eps-file can be viewed with `ghostscript` and for developing it is recommended to have `ghostscript` installed.
- To include Scalable Vector Graphics (svg)-files into LaTeX, `inkscape` must be installed. It also serves to create and to edit svg-files.

Currently for including pdf-files in both cases, the driver `dvipdfmx` must be installed. Strictly speaking, this is required only for html-creation and related. Note that if no pictures created by `fig2dev`, `gnuplot`, `mpost` or by `inkscape` are used, of course, neither `fig2dev` nor `gnuplot`, `mpost`, `inkscape` nor `dvipdfmx` is needed. To include graphics, the graphics bundle described in [Car16] is required, except for svg-files which requires the svg-package described in [Ilt12].

As the set of required software depends on the graphic formats which shall be imported, it depends also on the set of output-formats to be supported:

- To create pdf-files from LaTeX-files we use `pdflatex` or some other kind of LaTeX creating pdf-files like `xelatex` or `lualatex`. LaTeX uses several auxiliary programs. Above all `bibtex` to create the bibliography and `makeindex`

and `splitindex` for the index and `makeglossaries` for the glossary. The latter two also require the latex packages `makeidx`, optionally `showidx`, both described in [BLC+14], the package `splitidx` documented in [Koh16] and `glossaries` specified in [Tal16b]. Note that `makeglossaries` either invokes `makeindex` or `xindy`, depending on the parametrization of `glossaries`. Both, `makeglossaries` and `xindy` are written in Perl, which shall also be installed if a glossary is required.

The package `rerunfilecheck` is in any standard LaTeX-installation. It is almost mandatory because this software presupposes that package is present to ensure that the table of contents, list of figures, list of tables, the index and the glossary are up to date.

It is standard to endow a pdf-file with hyperlinks. To support this, the package `hyperref` is required.

****

- To create html-files, or to be more precise any kind of Standard generalized markup language (sgml) and extensible markup language (xml), from LaTeX-files, `htlatex` or alternatively `htxelatex` is used. Currently the author is not aware of any alternative to the two. This includes also creating open office documents like odt-files. Thus open office documents are created in two steps, the first is to create pdf-files with the according tools, the second one is done by `htlatex` or that like.

- To create rtf-files, currently `latex2rtf` is used. Note that this does not require `pdflatex`. As a drawback, not all LaTeX-packages are supported.

- MS word documents are created from open office documents via the command `odt2doc` and thus require three steps and so the according tool chain.

- Finally, there is a way, to create plain text files from the pdf-files via `pdftotext`. The way from LaTeX to text via pdf makes sense because that text is well formatted math mode symbols like π. and because table of contents, index, glossary and that like are included. So, for that task, besides `pdftotext` the whole toolchain to create pdf-files is required.

- An application which does not create a target, i.e. a file in the target directory is `chktex` which just checks the latex main files and associated files.

So to run this software, the aforementioned programs or at least the subset used, must be installed. To obtain reproducible results, the versions must fit. This version is checked with the executables with versions given by Listing 8.1 in Chapter 8.

There are also several LaTeX-packages needed or at least recommended. The recommended ones are

- `geometry` described in [Ume10] to control page layout.
- `microtype` described in [Sch16] improve readability and make the document look nicer. It also helps to avoid bad boxes.
- `hyperref` described in [RO22] to insert hypertext marks, which i do not want to miss in larger documents.
- `srcltx` described in [SU06] which allows to jump from the DVI file to the .tex source and back.
- `showframe` if `geometry` is not used with option `showframe`. There seems to be no package documentation for package `showframe`.
- `booktabs` described in [Fea16]
- `fix-cm` described in [SMCR15] and `anyfontsize` described in [Sza07] to allow arbitrary font sizes, eliminating certain warnings.

Almost required are

- `rerunfilecheck` described in [Obe16c] which writes additional rerun warnings to the log file if some auxiliary files have changed. This software relies on these warnings to control rerun latex and other applications.
- `ifthen` described in [Car14] which provides the `ifthenelse`-command which is needed to create both pdf and html and also to create rtf.
- `ifpdf` described in [Hei16] which provides the `\ifpdf`-command to detect pdf-mode. This is required to distinguish creation of pdf and text from html, odt, doc and others, based on dvi.
- The graphics packages described in [Car16], in particular `graphicx`, `xcolor` and `transparent`, the latter two described in [Ker16] and in [Obe16d], respectively. Sometimes also `bmpsize` described in [Obe16a] if pixel graphics is used.
- `import` described in [Ars09] e.g. to import nested graphic files from arbitrary directories.

- `inputenc` described in [JM15] to select an input encoding `fontenc` to select a font encoding. [Tea00] describes font selection in general, with Section 5 on font encoding and Section 5.1 on the `fontenc` package. This package is almost indispensable if you do not write English, e.g. to access German umlauts. Note that [MFL16] describes font encoding in more detail.
- `makeidx` and `showidx` described in [BLC+14] or something comparable for creating indices.
- `glossaries` described in [Tal16b] with tutorial [Tal16a] or something comparable for creating glossaries.
- `tocbibind` described in [WP10] to include bibliography and index (what about glossaries?) into the table of contents.
- `nag` described in [Sch11] which performs certain checks unveiling deficiencies not filtered by the compiler nor by another check tool.
- `babel` described in [BB16] for language support. This is not used by this manual, because it is in English.

Useful packages with which this software is tested:

- The ams-packages **** `amsmath`
- `longtable` described in [Car98] for long tables, i.e. tables exceeding a page.
- `listings` described in [HMH15] for listings.
- `fancyvrb` described in [Zan10] provides useful environments to mark verbatim text.
- `ifxetex` described in [Rob10] and `ifluatex` described in [Obe16b] are to determine the tex-engine (regardless the output format).

2.2 Setting pom.xml and build.xml

If this software is used as a maven plugin, it need not explicitly be installed, maven itself does this by need based on the entries of the pom.

Unfortunately, this plugin did not yet make it into maven central. Thus one has to add the providers repository to the pom as shown in Listing 2.1.

Then it can be used from command line, e.g. to create pdfs as `mvn latex:pdf` or for cleanup `mvn latex:clr` with with default configuration just adding the coordinates in the builts-plugin section of the pom as shown in Listing 2.2.

```
<project ...>
  ...
  <repositories>
    <repository>
      <id>publicRepoAtSimuline</id>
      <name>repo at simuline</name>
      <url>https://www.simuline.eu/RepositoryMaven</url>
    </repository>
  </repositories>
  ...
</project>
```

Listing 2.1: The source repository for this plugin

```
<project ...>
  ...
  <build>
    ...
    <plugins>
      ...
      <!-- create html and pdf and other formats from latex -->
      <plugin>
        <groupId>eu.simuline.m2latex</groupId>
        <artifactId>latex-maven-plugin</artifactId>
        <version>1.5</version>
      </plugin>
      ...
    </plugins>
    ...
  </build>
  ...
</project>
```

Listing 2.2: The coordinates of this plugin

```
<plugin>
  <groupId>eu.simuline.m2latex</groupId>
  <artifactId>latex-maven-plugin</artifactId>
  <version>1.5</version>
  <configuration>
  ...
  </configuration>
  <executions>
    <execution>
      <id>process-latex-sources</id>
      <!-- grp, dvi, pdf, html, rtf, odt, docx, txt, chk -->
      <goals><goal>cfg</goal></goals>
    </execution>
    <execution>
      <id>clear-latex-sources</id>
      <goals><goal>clr</goal></goals>
    </execution>
    <execution>
      <id>validate-converters</id>
      <goals><goal>vrs</goal></goals>
      <configuration>
        <versionsWarnOnly>true</versionsWarnOnly>
       </configuration>
    </execution>
  </executions>
</plugin>
```

Listing 2.3: The executions of this plugin

To make the plugin available within a build, one has to add executions, e.g. as shown in Listing 2.3: Typically, this plugin is used in the `site` lifecycle phase to process latex sources, but it must also be used to clean up the source directory in phase `clean`, because during document development that directory may be polluted. Finally, it is recommended to add a check of the converter versions right in the phase `validate`. Note that typically one will use goal `cfg` to create documentation because this allows to configure the output formats, but it may be also perfectly appropriate to stick to a single format as `pdf`. Cleanup is recommended to make the individual runs of this plugin independent. Finally, it is recommended to check the validity of the installed converters. Note the special configuration for that task which seems appropriate to skip info output on the console and have warnings if something goes wrong.

Note the section `configuration` in Listing 2.3 which is empty and can be skipped in a default configuration creating `pdf`- and `hmtl`-documentation. However, Listing 8.2 on page 102 lists the full configuration with default values and executions.

Chapter 6 describes the settings individually.

To check whether the installation of the plugin succeeded, in the directory of the pom command

```
mvn latex:vrs
```

which shall return meta info, above all the version of the plugin, and a list all converters needed together with the actual versions and the expected versions as displayed in Listing 3.3. Note that not all converters need to be installed, only the needed ones. For details see Section 3.3.

As you can see, the `taskdef`'s refer to java classes. Unlike maven which loads jars with the classes inside automatically from

```
https://www.simuline.eu/RepositoryMaven
```

the jar for the tasks, `latex-maven-plugin-1.5-antTask.jar`, must be downloaded manually from

```
https://www.simuline.eu/RepositoryMaven/eu/simuline/m2latex/latex-maven-plugin/1.5
```

Moreover, ant expects to find the jar files in an according folder. In my installation it is `/usr/share/ant/lib/`; as can be seen in the ant documentation, in general it is in folder `lib` in ant's installation directory.

The ant buildfile is given in Listing 8.3 on page 118. From that, one has to copy the following into the `build.xml` file in the current project:

- The properties `antJarDir` and `createdJar`,
- The path element with the id `latex.classpath`
- The taskdefs `latexCfg` and `latex:Clr`
- The targets `latex:cfg` and `latex:clr`

As for the maven plugin, for the ant task, add configuration, where a deviation from the default requires to do so.

2.3 Installation from source

The first step to install from source, is to clone from the repository by

```
git clone https://github.com/Reissner/maven-latex-plugin
```

of course assuming that `git` has been installed. Then change into the root repository where `pom.xml` for maven and also `built.xml` for ant are located.

To install the maven-plugin, ensure that maven is installed. One is tempted just to type

```
mvn clean install
```

but this does not work since the plugin needs itself to be installed to perform even `clean`. To solve that problem just comment out all its executions in the local `pom.xml` by enclosing them in `<!-...->`. In fact this is a minor bug, since, to be strict, only the executions for verification and clearing must be deactivated. For processing, it would be sufficient to add `<phase>site</phase>` to execution `process-latex-sources`.

Since the author develops with maven, including the development of the ant task, the maven built, creates the file `latex-maven-plugin-1.5-antTask.jar` defining the ant task. To this end, also `mvn clean package` is sufficient. After that, installation proceeds like described in Section 2.2 copying that jar file ant's lib-folder where ant can find it.

With root access and after having checked the proper paths, the build file `build.xml` can be used to perform copy task by `ant install`, to insert an according link by `ant link` to remove it again with `ant uninstall`. The build file `build.xml` works only if `latex-maven-plugin-1.5-antTask.jar` is placed where ant can find it or if the parts are deactivated below the line

```
<!-- deactivate the following unless the ant task is installed already -->
```

I feel building with maven and linking the jar created is a very good way to develop the ant task, because after changes the new ant task is available immediately.

For typical changes in the sources, it is possible to recompile and package the ant task by `ant jar` also cleanup is possible with `ant clean`. Finally, the ant task can be tested with `ant latex:cfg` and `ant latex:clr`.

In the long run, it should be possible to build the ant task from sources with ant alone.

Chapter 3

Usage of Plugin and Task

This software offers both, a maven plugin and an according ant task but the emphasis is on the maven plugin. Thus all sections of this chapter are either general or apply to the maven plugin. Only Section 3.5 specifically refer to the ant task. Usage presupposes installation as described in Chapter 2 including settings in `pom.xml` as described in Section 2.2.

This plugin may be used both if the LaTeX-sources are finished to create the output described by them and also to support development of the LaTeX sources. Accordingly, this chapter has Section 3.1 devoted to the form of the sources, including directory structure, LaTeX-files and others, mainly graphic files included and a section, 3.2 on exporting into various formats.

There is a very special usage, called development of documents, which means while the document is under construction. The features and goals tied to this phase are collected in Section 3.3.

In contrast, Section 3.4 is on usage of the maven plugin within the lifecycles. This can be used during development of documents but is more appropriate for small changes or when development finished at a stage.

3.1 The source files

The LaTeX-files and also files included via `\input` are in the *tex-source directory*, which is by default `./src/site/tex`, where `.` is the *base directory* of this maven-project. The LaTeX-files to be compiled top level, typically not inputted anywhere via `\input`, are called *LaTeX main files*. As an example, in the *tex-source directory* of this software, `manualLatexMavenPlugin.tex` is a LaTeX main file, whereas the file `header.tex` is not although also a LaTeX-file. LaTeX main files are detected automatically.

The included files may be again LaTeX-files but also graphic files in various

formats. There are four kinds of graphic formats, as regards the way their files are included in LaTeX-files, all included in the tex-source directory.

1. The first can be included into LaTeX-files directly via `\input`. These formats are essentially LaTeX and are defined in an according package. Examples are `eepic` described in [Kwo88] and above all `tikz` described in [Tan15].

2. the second one via the command `\includegraphics` defined by the package `graphicx` which is described in [Car16]. Chapter 2 therein mentions the supported drivers, among these are also `dvipdfm` and `dvipdfmx`. It is not the package but the driver which decides on the support of graphic formats. The dvipdfm user manual, [Wic99] lists the allowed formats metapost-output (i.e. metapost's postscript like output including text (mps)), postscript, pdf, Graphics format developed by the Joint Photographic Experts Group (jpg) and Portable Network Graphics (png).

3. the third one must be transformed into a graphics format of one of the former two kinds using an external tool for transformation. Here, of course, only a limited support is possible, because there is a broad variety of formats. We have chosen

   - the fig-format described in [ner16] because of its simplicity,
   - the gnuplot format , described in [WK20], because it allows computation of function plots,
   - scaleable vector graphics svg format specified in [Da11] as it is important for construction and the counterpart of pixel oriented formats. As the specification is hard to digest, we refer to the tutorial [DHH02].
   - likewise metapost (mp-format), described in [Hob14] because it is native to LaTeXand quite versatile.

4. finally the fourth kind of graphics formats has to be transformed into one of the kinds one or two but unlike in type three, this is not done explicitly by an external tool but by a latex-package during the LaTeX-run. Note that, although not required to be explicitly transformed, those graphics files induce additional files by running LaTeX. Essentially, each of the abovementioned type of format can be included that way but currently, this is done for the svg-format only included by the package `svg` (see [Ilt12]). The author personally refrains from using packages like that because of the lack of flexibility and further advantages.

The LaTeX-files and the graphic files belonging to a LaTeX main file are assumed to be in one single folder. If one file is included by two different main files, a link shall be used.

Note that unlike former version, the current version of this software does not create a working directory by cloning the tex-source directory. Instead, it operates directly on the tex-source directory also creating intermediate files. The advantage of processing that way is, that this allows cooperation between this software and other toolchains which are better suited for developping latex files. Details are described in Section 3.3.

The downside is that a file residing in the tex-source directory risks being overwritten by this software, if it does not stick to the rules. The rules are simple: For each graphic file, being transformed, i.e. of types 3 or 4 above, additional files are created with the same name up to the suffix. Thus for these graphic files no file with the same name up to the ending is allowed. The same is true for the LaTeX main files.

Besides the LaTeX-files and the graphics files there is a third kind of file supported: Bibliographies in bib-files. This software treats them automatically.

3.2 Exporting in various formats

After having added the configuration of the plugin to the `pom.xml`, minimally the one given in Section 2.2, it can be used directly invoking maven through `mvn latex:cfg`. Here `latex` is the (short) name of the plugin and `cfg` is the goal. It can also be interpreted as `mvn <source>:<target>`: The source files are in `latex`-format and the target-formats are read from the *configuration* in the pom (*configuration* is what `cfg` stands for).

By default, the target formats are `pdf` and `html`. The following Listing 3.1 shows a configuration with the full range of output formats including in addition the open office document format `odt`, the MS word-formats `doc(x)` and `rtf` and also plain text format `txt` in utf8 encoding. Note that the target `docx` converts by default into docx but may also be configured to produce the old fashioned outdated document format for MS word (doc) format.

The resulting files in the given output formats are copied to the site directory, which is `./target/site` in a default maven project.

Sometimes it is more convenient to specify the output formats not via the pom but directly as a goal on the command line. In particular, one may write `mvn latex:pdf` to create documentation in pdf-format only. Likewise command `mvn latex:dvi` to get good old dvi files or even `mvn latex:txt` for plain text, just as examples. Note that the `-X` switch activates debugging which results in a more verbose output. Example: `mvn -X latex:cfg`.

For creating the graphic files in the tex-source directory, there is a goal *graphics*, invoked by `mvn latex:grp`. This goal does not create any output in the site directory. Instead it populates the source directories with graphic files which can

```
<!-- create html and pdf and other formats from latex -->
<plugin>
  <groupId>eu.simuline.m2latex</groupId>
  <artifactId>latex-maven-plugin</artifactId>
  <version>1.5</version>

  <configuration>
    <settings>
      <targets>pdf, html, odt, docx, rtf, txt</targets>
    </settings>
  </configuration>
</plugin>
```

Listing 3.1: Configuration with full range output formats

be directly included into the LaTeX-file and so it allows to run the LaTeX-compiler on the latex main files from within a development environment. Thus the goal *graphics* is thus a vital feature for development of documentation.

Finally, there is another target for clearing the tex-source directory recursively, invoked by `mvn latex:clr`. For more details on the last three goals, see Section 3.3.

In a standard maven project, the above minimal configuration should be sufficient. Only if the folder structure deviates from the standard or if the LaTeX sources require special configuration, parameters have to be given explicitly, because they deviate from the default values. Chapter 6 summarizes all available parameters, giving the default value and a description.

For sake of uniformity, the name of the ant-task is `latex:cfg` and it can be invoked via `ant latex:cfg`. Unlike the maven-plugin, the ant-task does not allow to specify a target on the command line. The `-d` switch activates debugging which results in a more verbose output. Example: `ant -d latex:cfg`.

Whereas by default the target directory and in particular the target site directory with all output of this plugin is deleted in maven's `clean` life-cycle. As is described in more detail in Section 3.3, this software creates target documents and also intermediate files in the tex source directory, at least with cleanup disabled. To eliminate the created files from the source directory, just type `mvn latex:clr`. By default, the goal `clr` is also executed in maven's `clear` life-cycle.

There is an according ant-task `latex:cfg` which can be invoked via `ant -d latex:cfg`. FIXME: ant `latex:clr` has duplicate parameters. This can be fixed only by properties. Another problem is, to provide a complete subset of parameters which apply to `latex:cfg` and to `latex:cfg`, respectively.

If this ant-task is used in an ant project with folder structure conforming with a maven project and if the LaTeX sources do not require a special configuration, the

above configuration is sufficient. Otherwise, parameters have to be given explicitly overwriting the default values.

3.3 Development of Documentation

During development, it is comfortable, to have the log-file in the same directory as the LaTeX main file. Also if pdf- and tex-files are synchronized, also the pdf-file should be in the same directory. Likewise, files in graphic formats which cannot be included into a LaTeX-file without conversion, that converted file shall be in the same directory as the original one. So, all files, manually created files and files arising from automatic conversions shall be in the same folder, at least during development. Also, typically, one wants to mix creation by this maven-plugin or ant-task with at least partial creation through external tools. For example, if writing LaTeX-files with Emacs, it is much more convenient, to compile the LaTeX main file via `pdflatex` from within Emacs or to create a pdf-file from a fig-file through `xfig`'s export dialog, than using this maven-plugin or this ant-task. Also these tools work best, if all is in one folder.

On the other hand, conventionally, in a maven project, sources are held in folder `src`, whereas created files occur in the folder `target`. Likewise for ant. The compromise, this maven-plugin and this ant-task take, is, that at the end of a run, at most the files present at the beginning of the run may be present in the source directory. So, this software builds in the following steps:

- Store a list of all files present at the beginning of a run.
- Process all graphics files of the formats requiring preprocessing.
- Determine the LaTeX main files.
- Run the LaTeX converter, e.g. the one creating pdf-output or docx-output.
- Copy the result files (if any) into the target folder.
- Remove all files not present at the begin of a run, by default.

To keep e.g. the resulting pdf, just create it via compilation through Emacs, even if not all graphic files to be included are present or just by a `touch`-command. Then in the next run of this plugin, this pdf will be re-created, that time complete with the graphics output. That way, synchronization between LaTeX- and pdf-files is possible. Likewise, to keep the log-file or the aux-file, just touch it. This technique is really valuable for debugging.

To keep all created files after a run of this maven-plugin, set the parameter `cleanUp` in the pom to `false` as illustrated in Listing 3.2. For the ant-task likewise

But how can one get rid of all these newly created files? That is what is the goal `latex:clr` is for: `mvn latex:clr` removes all created graphic files and for each latex main file, it removes all files with "similar" names including log files, index files and that like. Typically, this suffices, to remove all files created. If not, try to modify parameter `$patternCreatedFromLatexMain` in the pom accordingly. If this does not help either, please inform the developer of this software. Of course, if further software is used which creates additional files, like Emacs creates a folder `auto`, these files cannot be removed by this maven-plugin or this ant-task. Note that `latex:clr` also removes exported files as listed in Section 3.2 from the target folder.

During development of a LaTeX-main file, it is often more convenient to compile from within an editor like Emacs. The problem is, that compilation fails if the graphic files are missing. This is what the goal *graphics* accessible via

```
mvn latex:grp
```

is for: It creates all graphic files required to compile the LaTeX-main files.

Still this does not create a bibliography, an index or a glossary. With *auctex*, an Emacs-package for editing LaTeX, bibliography and index are well supported. To create a glossary, auctex has to be modified a little.

That way also the log-files required are created: In case of this manual, the files `manualLatexMavenPlugin.xxx` are created where `xxx` is

- `log` for LaTeX,
- `blg` for `BibTeX`,
- `glg` for `makeglossaries` and
- `ilg` for `makeindex`.

The last goal regularly used for development of documentation is *check*. It is invoked via

```
mvn latex:chk
```

and runs `chktex`, described in [Thi16], on each latex main file after having created graphic files as for goal *graphics*. As a result, a log-file with suffix `.clg` is created but not copied to the target folder. If the log-file contains an entry, an according message is logged. Note that, with default configuration, `chktex` requires the LaTeX-package `booktabs` described in [Fea16].

Finally, we have the goal `latex:vrs` to display meta information, above all version information:

```
mvn latex:vrs
```

displays something like what is displayed in Listing 3.3. As one can see, there may be converters with inappropriate versions which is indicated by a warning WMI02 and there may be converters which are excluded according to the configuration `convertersExcluded`.

Note that in the given version and in the installation of the author, of course, all converters are installed and are up-to-date to be able to check validity. The according messages are for illustration only.

3.4 Goals in the maven lifecycle

The goal `latex:cfg` exporting in the formats configured is tied to the lifecycle phase `site` so is invoked when commanding

```
mvn site
```

or subsequent phase.

Also, the goal `latex:clr` cleaning created files both from source directory and from target directory is tied to phase `clean` so is invoked when commanding

```
mvn clean
```

Finally, the goal `latex:vrs` displaying versions of converters is tied to the phase `validate`. Thus it is thus invoked when commanding

```
mvn validate
```

which is invoked not only in installation, but also by the site plugin. This ensures, that the converters are checked for correct version before being used. Note that by default, `mvn latex:vrs` displays complete version info, whereas `mvn validate` only displays warnings if appropriate.

3.5 The ant-tasks

This section is missing. What has to be described are the two tasks.

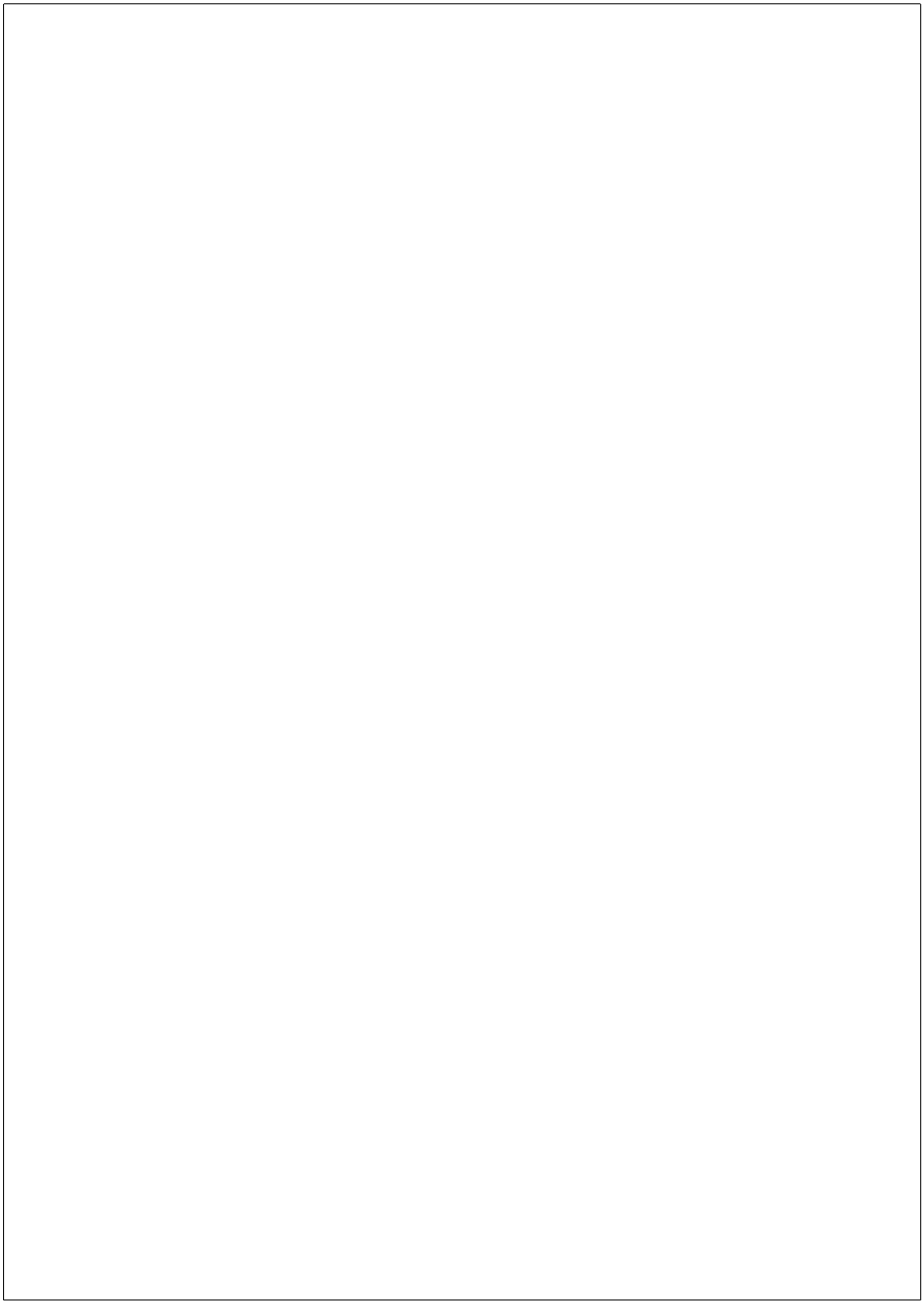
```
<!-- create html and pdf and other formats from latex -->
<plugin>
  <groupId>eu.simuline.m2latex</groupId>
  <artifactId>latex-maven-plugin</artifactId>
  <version>1.5</version>

  <configuration>
    <settings>
      <targets>pdf</targets>
      <cleanUp>false</cleanUp>
    </settings>
  </configuration>
</plugin>
```

Listing 3.2: Configuration without cleanup

```
[INFO] --- latex-maven-plugin:1.5-SNAPSHOT:vrs (default-cli) @ latex-maven-plugin ---
[INFO] Manifest properties:
[INFO] MANIFEST: (1.0)
[INFO]         Implementation-Version: '1.5-SNAPSHOT'
[INFO] PackageImplementation-Version: '1.5-SNAPSHOT'
[INFO] pom properties:
[INFO] coordinate.groupId:     'eu.simuline.m2latex'
[INFO] coordinate.artifactId: 'latex-maven-plugin'
[INFO] coordinate.version:     '1.5-SNAPSHOT'
[INFO] git properties:
[INFO] build version:  '1.5-SNAPSHOT'
[INFO] commit id desc: 'latex-maven-plugin-1.4-194-g5f46a4e-dirty'
[INFO] buildTime:      '2022-03-02T22:12:59+0100'
[INFO] tool versions:
[INFO] ?warn?    command              'actual version'(not)in[expected version interval]
[INFO]           pdflatex:            '1.40.22'in[1.40.21;1.40.22]
[INFO]           lualatex:            '1.13.2'in[1.12.0;1.13.2]
[INFO]           xelatex:             '0.999993'in[0.999992;0.999993]
[INFO]           latex2rtf:           '2.3.18 r1267'in[2.3.16 r1254;2.3.18 r1267]
[INFO]           odt2doc:             '0.9.0'in[0.9.0]
[INFO]           pdftotext:           '22.01.0'in[21.04.0;22.01.0]
[INFO]           dvips:               '2021.1'in[2020.1;2021.1]
[INFO]           dvipdfm:             '20210318'in[20210318]
[INFO]           dvipdfmx:            '20210318'in[20200315;20210318]
[INFO]           xdvipdfmx:           '20210318'in[20200315;20210318]
[INFO]           dvipdft:             '20090604.0046'in[20090604.0046]
[INFO]           gs:                  '9.54.0'in[9.52.0;9.54.0]
[INFO]           chktex:              '1.7.6'in[1.7.6]
[INFO]           diff-pdf-visually:   '0'in[0]
[INFO]           diff-pdf:            '300'in[300]
[INFO]           diff:                '3.8'in[3.8]
[INFO]           pdfinfo:             '22.01.0'in[22.01.0]
[INFO]           exiftool:            '12.39'in[12.39]
[INFO]           bibtex:              '0.99d'in[0.99d]
[INFO]           bibtexu:             '3.71'in[3.71]
[INFO]           bibtex8:             '3.71'in[3.71]
[INFO]           makeindex:           '2.15'in[2.15]
[INFO]           splitindex:          '0.1'in[0.1]
[INFO]           makeglossaries:      '4.45'in[4.45]
[INFO]           pythontex:           '0.17'in[0.17]
[INFO]           latexmk:             '4.70b'in[4.70b]
[INFO]           mpost:               '2.01'in[2.00;2.01]
[INFO]           ebb:                 '20210318'in[20200315;20210318]
[INFO]           gnuplot:             '5.4 patchlevel 3'in[5.4 patchlevel 0;5.4 patchlevel 3]
[INFO]           inkscape:            '1.1.1'in[1.0.2;1.1.1]
[INFO]           fig2dev:             '3.2.8b'in[3.2.7b;3.2.8b]
[INFO] excluded tools:
[INFO] upmendex, xindy
[INFO] ------------------------------------------------------------------------
```

Listing 3.3: Output of goal `latex:vrs`



Chapter 4

Graphical Preprocessing

While LaTeX is really strong in text processing and also in formula processing, in itself it is weak in its graphical abilities. To include graphics into a LaTeX-document, several LaTeX-packages are required and must be installed:

graphicx is the basic graphics package which provides the command `\includegraphics` which allows to include graphics natively in the formats pdf, eps, jpg and png at least. For details see [Car16].

xcolor allows to use colors in graphics. Even if the author does not use graphics, several formats, like fig, Gnuplot file format (gp) and svg offer it and so the according converters transforming them into the native formats create color information and so the package must be installed if using those formats. For details see [Ker16].

transparent allows to specify transparency in graphics. Here the same applies as for color; Even if you do not use the feature, some source formats do (in fact only svg) does and so the according converters create according information and so LaTeX must get along with it. For details see [Obe16d].

bmpsize is needed for bitmap formats like jpg and png only. Used to extract resolution and bounding box. FIXME: needed more information. For details see [Obe16a].

import is strictly speaking no graphics package. According to its documentation [Ars09], it allows an imported file to find its own inputs (using "`\input`", "`\includegraphics`" etc.) in that directory. This is vital for the graphic formats for which a tex-file is imported which itself imports a pdf/eps file located in the same folder but not in the folder of the importing file.

We support figures created in the fig by the program xfig, figures created by the plotting utility `gnuplot`, the tex-specific graphic language metapost, figures in

svg-format and finally figures in formats which can be included into a LaTeX-file without preprocessing like png and jpg. Further functionality on LaTeX figures can be easily added. If there is some need, please write an email to the author.

Table 4.1 gives an overview over the supported formats together with the graphic converters, the name of the parameter to configure which one is used and the default value. Of course this converter must be installed. It is advisable to have also an editor installed. Sometimes the editor is used also as converter. For human readable formats like `fig`, it often makes sense, to use both the graphical editor and the textual one.

Graphic format	conversion tool	editor
fig	`fig2devCommand`, e.g. `fig2dev`	xfig, emacs
gnuplot (gp)	`gnuplotCommand`, e.g. `gnuplot`	emacs
metapost (mp)	`metapostCommand`, e.g. `mpost`	emacs
svg	`svg2devCommand`, e.g. `inkscape`	inkscape
jpg, png	—	gimp

Table 4.1: Overview over the supported graphic formats

Besides the converter external to LaTeX, also several LaTeX-packages are required to use graphics.

This section describes the conversions of graphical source files into target files in detail.

But pdf also occurs as an intermediate format for pictures. For historical reasons, still eps is used. Section 4.1 shows that it suffices to stick to pictures in pdf-format. Section 4.2 shows how `fig2dev` converts fig-files into LaTeX-files containing text and including graphics in as pdf-files. Likewise, Section 4.3 describes how gnuplot converts gnuplot-files into pdf-files. An interesting alternative to gnuplot for computing pictures is metapost described in Section 4.4. A more elaborate alternative to fig-pictures are svg-pictures described in Section 4.5 Also several formats collected in Section 4.6 may be included as is.

4.1 Including pdf-files

Modern LaTeX implementations directly create pdf-files. Thus it makes sense, to allow including graphics as pdf-files in LaTeX-files. This is done in pdf mode by the LaTeX-packages `xcolor` and `graphicx`. In contrast, traditionally LaTeX produced output in the dvi-format which is still used to create html-output, this is not supported by the default driver, Instead it shall be used the driver `dvipdfmx`. To obtain a LaTeX-file which works both for creating pdf and html, insert in the header of the LaTeX main file given by Listing 4.1.

```
\newboolean{texFhtLoaded}
\setboolean{texFhtLoaded}{false}

% only with pdflatex, warnings for xelatex and for lualatex
% ifxetex, ifluatex, ifpdf
\usepackage{ifpdf,ifxetex,ifluatex}
\ifpdf%
\else
\makeatletter
\@ifpackageloaded{tex4ht}{%
  \setboolean{texFhtLoaded}{true}
}{%
}% tex4ht not loaded
```

Listing 4.1: The header of the tex-source of this manual

```
\Preamble{xhtml}
%\makeatletter
\Configure{graphics*}
  {pdf}
  {
    \Needs{"convert       \csname Gin@base\endcsname.pdf %
                          \csname Gin@base\endcsname.eps"}%
    \Picture[picture not present]       {\csname Gin@base\endcsname.eps}%
    \special{t4ht+@File: \csname Gin@base\endcsname.eps}
  }
\begin{document}
\EndPreamble
\input{myxhtml.cfg}
```

Listing 4.2: Configuration file `myxhtml.cfg`

An alternative, which we do not use is replacing the pdf by an eps via configuration file `myxhtml.cfg` which is given by Listing 4.2. It applies when converting LaTeX into html and that like with htlatex if invoking something like `htlatex xxx.tex myxhtml,...` where `myxhtml` is located in the same directory as the LaTeX-file to be processed. The configuration file essentially contains a graphics configuration, which applies conversion to pdf-files included in the LaTeX-file `xxx.tex` via `\includegraphics`.

4.2 Conversion of fig-files

A simple but still useful tool to draw figures is `xfig` which stores graphics in a native format described in [ner16] with file extension `.fig`. The file extension `.fig` is also used by matlab to store plots but this is something different. Graphics in

```
\begin{picture}(0,0)%
\includegraphics{F4_01fig2dev}%
\end{picture}%
%
%  Conversion of xxx.fig into xxx.ptx, xxx.pdf and xxx.eps
%
\setlength{\unitlength}{2072sp}%
%
\begingroup\makeatletter\ifx\SetFigFont\undefined%
\gdef\SetFigFont#1#2#3#4#5{%
  \reset@font\fontsize{#1}{#2pt}%
  \fontfamily{#3}\fontseries{#4}\fontshape{#5}%
  \selectfont}%
\fi\endgroup%
\begin{picture}(8492,4797)(1114,-4621)
\put(1351,-2311){\makebox(0,0)[lb]{\smash{{\SetFigFont{10}{12.0}{\rmdefault}{\mddefault}{\
      updefault}{\color[rgb]{0,0,0}\texttt{xxx.fig}}%
}}}}
\put(6976,-286){\makebox(0,0)[lb]{\smash{{\SetFigFont{10}{12.0}{\rmdefault}{\mddefault}{\
      updefault}{\color[rgb]{0,0,0}\texttt{xxx.pdf}}%
}}}}
\put(6976,-2311){\makebox(0,0)[lb]{\smash{{\SetFigFont{10}{12.0}{\rmdefault}{\mddefault}{\
      updefault}{\color[rgb]{0,0,0}\texttt{xxx.ptx}}%
}}}}
\put(6976,-4336){\makebox(0,0)[lb]{\smash{{\SetFigFont{10}{12.0}{\rmdefault}{\mddefault}{\
      updefault}{\color[rgb]{0,0,0}\texttt{xxx.eps}}%
}}}}
\put(4726,-1636){\makebox(0,0)[b]{\smash{{\SetFigFont{10}{12.0}{\rmdefault}{\mddefault}{\
      updefault}{\color[rgb]{0,0,0}\texttt{fig2dev -L pdftex\_t}}%
}}}}
```

Listing 4.3: The ptx-file for a fig-file

xfig format cannot be directly included in latex files but must be exported into a LaTeX-readable format.

To export a file `xxx.fig` residing in directory `yyy` into several external formats, `xfig` uses `fig2dev`. A look in [ner16], Section 3.4 shows that texts with set "special"-flag are interpreted as latex-code. For these texts the appropriate export language would be `latex`. On the other hand, `latex` is weak in graphics and `pdf` would be the ideal export format for all kinds of objects, except for texts with set "special"-flag. In `pdf` format, texts are interpreted literally, independent of the "special"-flag. Thus `fig2dev` offers a mixed solution: export `xxx.fig` in format `pdftex` which yields a pdf-file `xxx.pdf` containing all but text with set "special"-flag and complementary `pdftex_t` which yields a tex-file `xxx.ptx` including the pdf-file and the texts with set "special"-flag. The exported files are in the same directory `yyy` as the original file `xxx.fig`.

For example, the fig-file `F4_01fig2dev.fig` defining Figure 4.1, is transformed into a file `F4_01fig2dev.ptx` in format `pdftex_t` which starts as given by Listing 4.3.

The file `xxx.ptx` is "imported" into the tex-file of this manual by the command

```
\import{yyy}{xxx.ptx}
```

and includes `xxx.pdf` automatically the file `xxx.pdf` via `\includegraphics{xxx}` (line 2). Note the following remarkable details:

- Observe that we can drop the suffix of the included file `xxx.pdf` which is expressed as "`xxx`" because LaTeX chooses the right suffix: If instead of

xxx.pdf there is a file xxx.eps, the latter is chosen if no suffix is specified. As we will see below, omitting the suffix is crucial to make xxx.ptx work for both LaTeX-output formats: the pdf-format can include pdf-files, whereas the dvi-format which is required to create html- and odt-files can include eps-files.

- If xxx.pdf is included in xxx.ptx with the full path name, we may use \input{xxx.ptx} instead of \import{yyy}{xxx.ptx}.

  If in contrast, xxx.pdf is included in xxx.ptx with the short name only, xxx.pdf is assumed to be in the same directory as the file inputting xxx.ptx. So in general, i.e. if this is not yyy, we need import \import{yyy}{xxx.ptx}. If the directories coincide, in the import the string yyy may be empty. If the string yyy is not empty, it must end with the path delimiter, i.e. / for Unix like systems and \ for win-like systems.

As indicated in Section 4.1, the commands in xxx.ptx require the packages graphicx and xcolor. Also the \import command requires the import package.

To export xxx.fig into xxx.ptx and xxx.pdf this software invokes two commands:

```
fig2dev -L pdftex   <fig2devGenOptions> <fig2devPdfEpsOptions>        xxx.fig xxx.pdf
fig2dev -L pdftex_t <fig2devGenOptions> <fig2devPtxOptions>    -p xxx xxx.fig xxx.ptx
```

Both commands specify the input file xxx.fig, both use the options given by the parameter fig2devGenOptions while each invocation allows to specify also specific options, fig2devPdfEpsOptions and fig2devPtxOptions, respectively, and both use the option -L to specify the output format ("language").

The parameters specific for pdftex are called fig2devPdfEpsOptions because the options available are the same as for output format pstex creating eps-files. An example for a common option would be -b width which shall specify the same boundary for both formats; otherwise they do not fit.

For the output format pdftex_t, the option -p xxx says, that the string xxx must be included in xxx.ptx as \includegraphics{xxx}. Note that the option -p shall not be specified in fig2devPtxOptions, because it is automatically added.

Equivalent to mixed export with formats pdftex and pdftex_t which is appropriate for LaTeX-output format pdf, is the mixed export with the according formats pstex and pstex_t appropriate for LaTeX-output format dvi. The difference is that pstex creates an eps-file instead of a pdf-file with the same content and pstex_t creates a tex-file which looks like that created by pdftex_t except including the eps-file instead of the pdf-file. If the suffix is not given, pstex_t and pdftex_t create identical files. Thus exporting xxx.fig via

```
fig2dev -L pstex    <fig2devGenOptions> <fig2devPdfEpsOptions>        xxx.fig xxx.eps
fig2dev -L pdftex   <fig2devGenOptions> <fig2devPdfEpsOptions>        xxx.fig xxx.pdf
fig2dev -L pdftex_t <fig2devGenOptions> <fig2devPtxOptions>    -p xxx xxx.fig xxx.ptx
```

and "inputting" `xxx.ptx` works for both LaTeX output formats.

Note that `xfig` chooses its file suffix according to Table 4.2 which deviate from those used by this software. The suffixes used here, better reflect the file formats. We opted for the quite unusual suffix `.ptx` instead of `.tex` to avoid that tex-files may be both, source files and created files, but this is not compulsory, since the same holds and is accepted for pdf-files.

Output format (language)	xfig suffix	our suffix	format
pstex	pstex	eps	eps
pstex_t	pstex_t	ptx	tex
pdftex	pdf	pdf	pdf
pdftex_t	pdf_t	pdf	pdf

Table 4.2: Suffixes used by xfig as opposed to suffixes used here and actual file format

Maybe xfig is intended to export from within the export dialog and not directly via a script like `fig2dev`. This may be the reason why the magnification must be set in the export dialog, but it is stored in the fig-file nevertheless.

Figure 4.1 shows the transformation of figures with `fig2dev` and the inclusion of the eps-file and of the pdf-file in the ptx-file. Note that the `fig2dev`-command is configurable via the parameter `fig2devCommand`, but there will be hardly any command with the same command line interface performing exactly the transformations given in Figure 4.1, except `fig2dev` itself.

At the same time, Figure 4.1 is an example for a LaTeX-file `xxx.ptx` created from a fig-file and embedded in this LaTeX-file with the `\input`-command. More than that, Figure 4.1 describes the way it has been created. Note that all text labels are specified with set "special"-flag, and are thus included as LaTeX-text, except the text postscript which is typeset with a postscript font to make the difference visible.

4.3 Conversion of gnuplot-files

The term "gnuplot" refers to a file format and to a program `gnuplot` which can read this format, both described in [WK20].

Note that there seems no official file extension to identify gnuplot files. From the most common extensions `.plt`, `.gpi` and `.gp` we have chosen the one with the least collision and supported by Emacs and by my file browser: `.gp`.

The gnuplot format is a textual command language you can even program with and may thus be created with any editor but for sake of reproducibility it is recommended to use only files created by `gnuplot`. To ensure that a hand-written gnuplot file `xxx.gp`, e.g. with a single line like

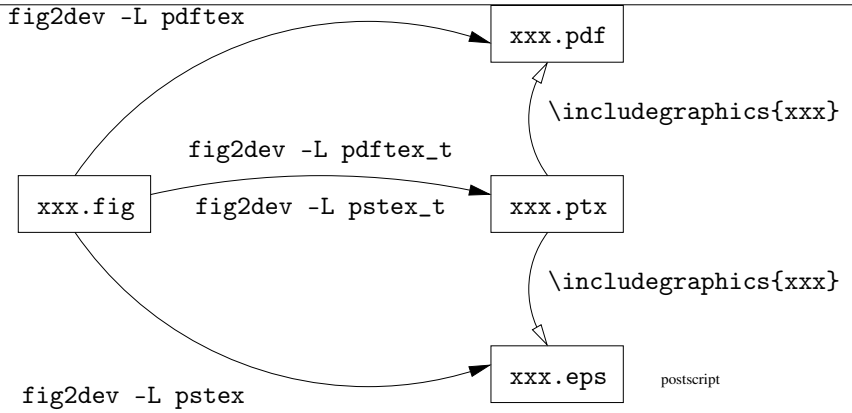


Figure 4.1: Conversion of a fig-file into pdf-, eps- and ptx-files with inclusions

```
plot [-10:10] sin(x), atan(x), cos(atan(x))
```

really works with the current `gnuplot` and to see how it is interpreted, it is recommended to convert it via

```
gnuplot -persist -e "load 'xxx.gp'; save 'xxx.gp'"
```

If you have a look inside, you can see, that in a comment line the current version of `gnuplot` is documented and also all the settings implicitly used. The original line is the last but one.

Also if a gnuplot file is created with an old version of `gnuplot`, it is recommended to update version with the same command. Note that `gnuplot` does not offer full backward compatibility.

This software supports including figures stored in `.gp`-files created by `gnuplot`. To export a file `xxx.gp` into several external formats, it uses `gnuplot` itself. According to the manual [WK20], Part IV, `gnuplot` supports output formats through so called *terminals*. Among those are several ones intended for inclusion into LaTeX-files, like `Cairolatex`, `Eepic`, `Epslatex`, `Latex`, `Lua`, `Mf`, `Mp`, `Postscript`, `Ps(la)tex`, `Pstricks`, `Texdraw` and `Tpic`. Note that also export into the fig-format via the terminal `Fig` is supported which in turn may be included in latex as described in Section 4.2. Also `gnuplot` pictures may be exported in metapost format which in turn may be included in latex as described in Section 4.4.

This software supports the export of a file `xxx.gp` only via the terminal `Cairolatex` which offers export to mixed pdf and LaTeX: graphics in pdf and text in LaTeX which yields the fonts typical for LaTeX. This is as described for fig-files in Section 4.2, except that text is generally converted in LaTeX-format, and not selectively those text marked with special flag.

Accordingly, the export yields two files `xxx.ptx` and `xxx.pdf`, both in the directory `yyy` in which `xxx.gp` resides. The file `xxx.ptx` must be imported via

```
\import{yyy}{xxx.ptx}
```

It contains the texts and includes `xxx.pdf` via `\includegraphics{xxx}` without specifying a suffix.

Unlike for fig-files, `xxx.ptx` and `xxx.pdf` are created with a single command:

```
gnuplot -e "set terminal cairolatex pdf <gnuplotOptions>;
            set output 'xxx';
            load 'xxx.gp'"
```

Accordingly, `xxx.ptx` and `xxx.eps` are created with a single command:

```
gnuplot -e "set terminal cairolatex eps <gnuplotOptions>;
            set output 'xxx';
            load 'xxx.gp'"
```

Note that this writes another but identical file `xxx.ptx`. Thus performance is not optimal, but `gnuplot` offers no way to avoid this. If being strict, `xxx.ptx` is perfectly correct only for output `eps`, if comments and error messages are taken into account but as long as no error occurs, the result is perfectly ok also for `pdf`.

As for inclusion of fig-files, packages `graphicx` and `color` are needed.

Figure 4.2 shows the transformation of the plots and the inclusion of graphic files. In addition, Figure 4.3 shows an example of a LaTeX-file created from a gnuplot file and embedded in this LaTeX-file.

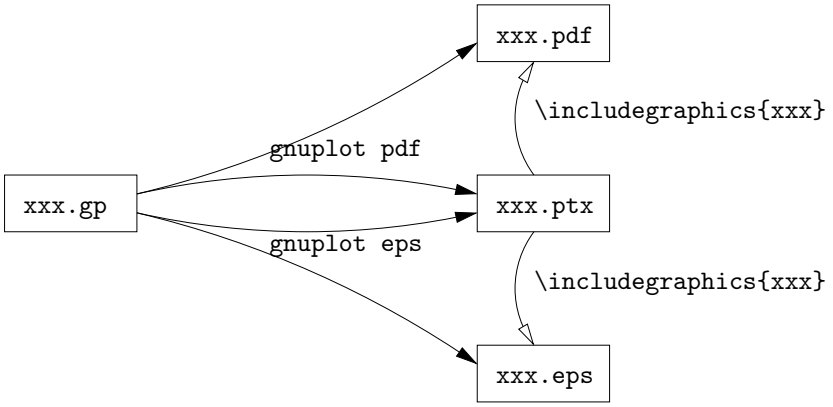


Figure 4.2: Conversion of a gnuplot-file into pdf-, eps- and ptx-files with inclusions

4.4 Inclusion of Metapost files

A graphic format, very native to TeX is `metapost`, a derivative of `metafont` originally used to describe shape of fonts. Although seemingly supported by TeX

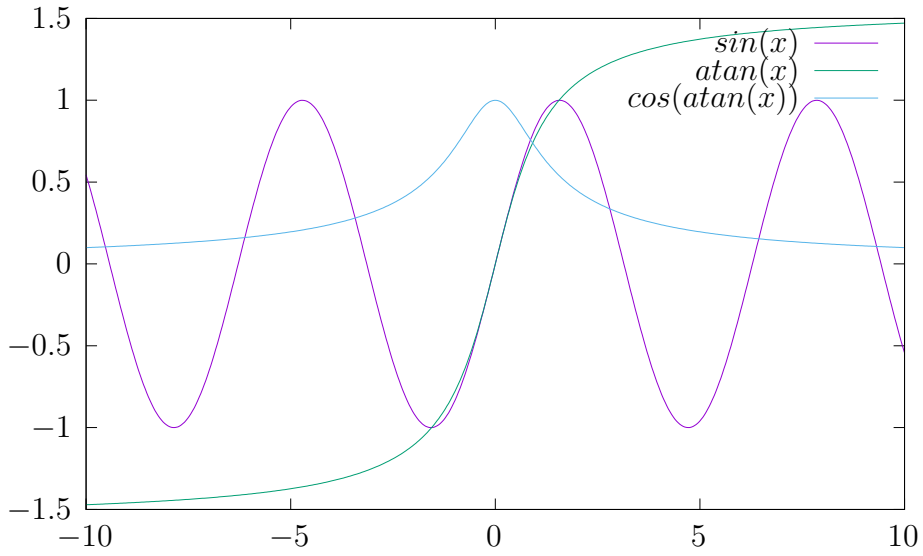


Figure 4.3: Converted sample gnuplot-file into ptx and pdf files

only, `metapost` is interesting at its own, as it is a graphical programming language, Turing complete, much like postscript. Files containing `metapost` have the ending `.mp`. Note that there are other graphic formats like monochrome pictures in TIFF-format which are identified with the same extension but the metapost format has nothing to do with this.

The interpreter for `metapost` is given by the parameter `metapostCommand` which defaults to `mpost`.

Figure 4.4 illustrates how `mpost` converts an mp-file `xxx.mp` by default:

- into one or more mps-files `xxx1.mps`...`xxxn.mps`, (as the mp-file may declare more than one figure)
- creating a log-file `xxx.log` much like LaTeX does
- and an metapost tex output: texts (mpx)-file `xxx.mpx` containing the text of the figure.

The mps-files look like postscript, or more like encapsulated postscript, but are not completely valid, since the prologue is missing, by default. Inserting the line "`prologues := 2;`" in the first line of the mp-file solves the problem as [Hob14], Section 14.2.1 explains. Then it can be viewed in ghostscript.

A much better alternative is to specify the the option

Roughly speaking, the mpx-file contains text parts of the mp-file; more knowledge is not necessary, as the texts are also in the mps-files and so the mpx-file is not used by this software.

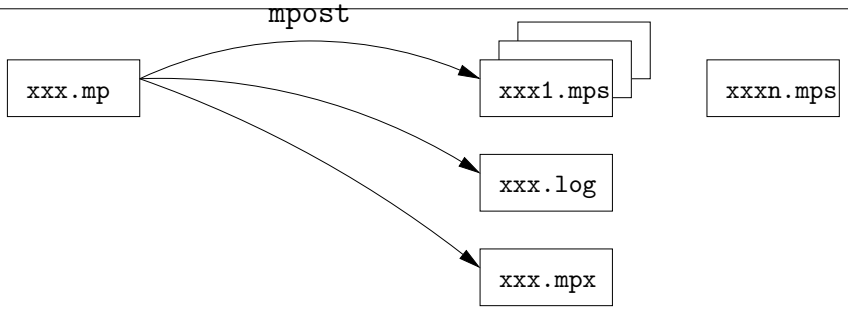


Figure 4.4: Conversion of a metapost-file into an mps-file

Figure 4.5 gives an example of a metapost file included in this LaTeX-file as ab mps-file created from the metapost file and embedded in this LaTeX-file with the `\includegraphics`-command.

Figure 4.5: Converted sample metapost-file included as mps-file

Note that `metapostCommand` may also besides eps output svg and png, just by setting

```
outputformat:="svg"
```

or that like (caution: case sensitive, assuming silently `eps` if not recognized). One would also adapt

```
outputtemplate:="\%j\%c.svg"
```

to control the names of the mps-files created. Here `%j` represents the jobname and %*c* the number. One may well define alternatively

```
outputtemplate:="\%j-\%c.svg"
```

Finally we may set:

```
prologues := 3;
```

So the mp-file starts

```
outputformat:="svg";
outputtemplate:="\%j\%c.svg";
prologues := 3;
```

As an alternative, `mpost` compiler may be invoked with option

```
-s<key>=<value>
```

described in [Hob14], Appendix B.2.1. That way we can set internal variables with values overwriting those given by the mp-files. We recommend specification of internal variables through options if required globally.

This would lead to invocation

```
mpost -s 'outputformat="svg"' -s 'outputtemplate:="%j%c.svg"' -s prologues=3 file.mp
```

Note that the output

4.5 Inclusion of svg-files

Comparable with the xfig-format described in Section 4.2 but much more elaborate and widely used is the svg-format. There is a huge up-to-date official SVG 1.1 specification, [Da11] and a specification for SVG Tiny 1.2, [Aa08] which is itself quite short and more readable and gives also a good overview on "SVG Big". For a tutorial, see [DHH02]. As stated in [Aa08], Section 1.1, svg-files may contain vector graphics, raster images and text. It may also contain video and audio elements and may be interactive and dynamic, which goes beyond what can be included in LaTeX-files.

Figure 4.7 shows a picture in svg-format. As pdf-files are included directly via the `\includegraphics`-command, using the LaTeX-packages `xcolor` and `graphicx`, virtually, `xxx.svg` can be included directly via

```
%\includesvg[width=0.5\textwidth]{xxx}%
```

using the LaTeX-packages `svg` described in [Ilt12]. Note that the suffix of the file name shall be omitted.

A closer look shows, that graphic preprocessing is done behind the scenes in the course of a LaTeX-run creating files `xxx.pdf` and `xxx.pdf_tex`. As described for fig-files in Section 4.2 and for gnuplot-files in Section 4.3: The latter is a LaTeX-file containing text and including the former. To include `xxx.pdf` of course the LaTeX-packages `xcolor` and `graphicx` are required. Moreover, it may happen that the LaTeX-package `transparent` is required also, depending on the features used in`xxx.svg`.

As indicated in [Ilt12], Section 1, the `svg`-package delegates the transformation of `xxx.svg` `xxx.pdf` and `xxx.pdf_tex` to `inkscape`. This is a graphical editor with export functions which can be invoked in batch-mode also. Of course using the `svg`-package has the advantage that no explicit preprocessing is required, the created files updated by need. It is worth thinking about whether it is worthwhile writing according packages `fig` and `gnuplot`.

On the other hand, this breaks the workflow this software normally applies to graphic files. In particular, the package creates latex main files which are not removed after the latex run if parametrized accordingly or if something goes wrong. Also, the `svg`-package does not provide the full flexibility of a standard solution. Since this software is still under construction and more than that, is in an experimental phase, we provide explicit preprocessing of svg-files using `inkscape`. Another problem with the `svg`-package is, that according to [Ilt12], Section 1, it does not work on windows platforms.

Some research shows, that `inkscape` in the version current at time of this writing exports mixed pdf and latex: If invoked as

```
inkscape -D --export-filename=xxx.pdf --export-latex xxx.svg
```

`inkscape` creates a file `xxx.pdf` containing all graphics but text and another file `xxx.pdf_tex` containing text and including `xxx.pdf`. The file `xxx.pdf_tex` can be be integrated into the latex document as

```
\def\svgwidth{0.5\textwidth}
\import{yyy}{xxx.pdf\_tex}%
```

Unlike `fig2dev` and `gnuplot`, specifying the files with their full path, has no effect, i.e. inclusion uses the file name only. Thus `\import` cannot be replaced by `\input` and so the LaTeX-package `import` is required.

This is essentially the same technique as applied for fig-files and for gnuplot-files as described in Sections 4.2 and 4.3.

Analogously,

```
inkscape -D --export-filename=xxx.eps --export-latex xxx.svg
```

exports files `xxx.eps_tex` and `xxx.eps`.

In older versions of `inkscape`, there was a configuration allowing `xxx.eps_tex` to include uniformly both `xxx.pdf` and `xxx.eps`. Thus `xxx.pdf_tex` could be deleted and `xxx.eps_tex` moved to `xxx.ptx` which in turn could be included into the main document.

As shown in Figure 4.6, for the current version of `inkscape`, this software filters `xxx.eps_tex` into `xxx.ptx` "manually" so that both `xxx.pdf` and `xxx.eps` are included in `xxx.ptx`. Then it deletes the original files `xxx.pdf_tex` and `xxx.eps_tex`.

The author has filed a bug report to the inkscape-team, to avoid this workaround in future.

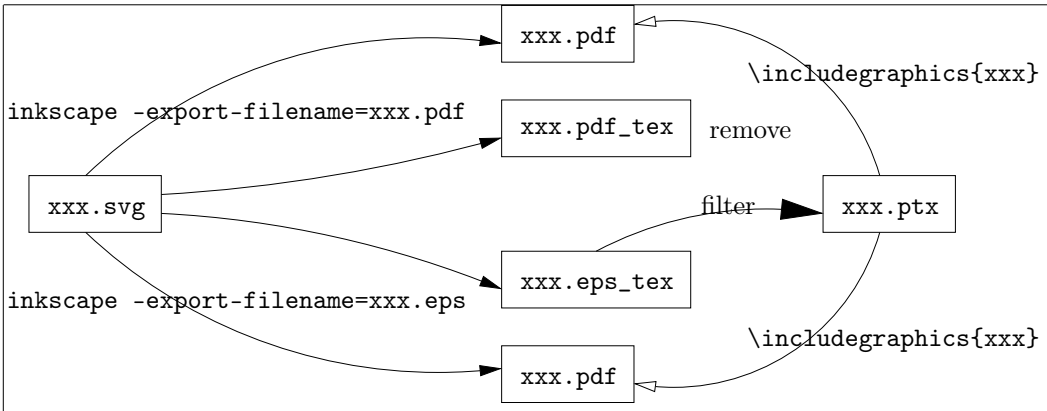
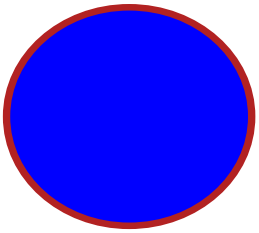


Figure 4.6: Conversion of an svg-file into pdf-, eps- and ptx-files with inclusions



Hello World
From An SVG File!

Figure 4.7: Some svg-picture with text FIXME: uniformity

4.6 Pictures which are not transformed

Figure 4.8 shows some picture included as jpg. This is done as usual with the command `\includegraphics` provided by the package `graphicx`. According to the documentation [Car16], page 13, the bounding box must be provided somehow. This may be done via the package `bmpsize` but also using the command `ebb`. FIXME: further research and further documentation is required.

Note that both for `pdflatex` and siblings creating pdf-output and for `htlatex` in conjunction with `dvipdfmx` (see Section 4.1) files in the format pdf, png, jpg are supported. **** what about Graphics Interchange Format, allows also animations (gif)? This list may be incomplete.

As an example, Figure 4.9 shows the same picture as png-file.

FIXME: at the moment, htlatex does not work with pictures at all.

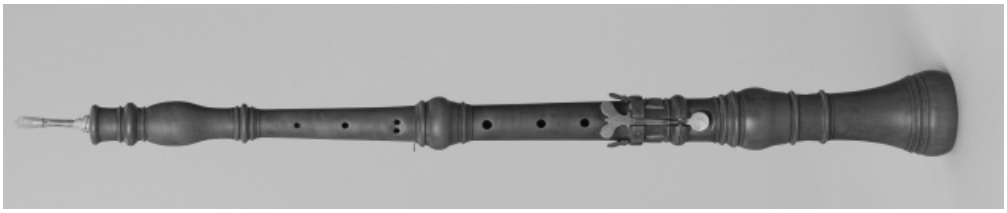


Figure 4.8: Some jpg-picture, directly included

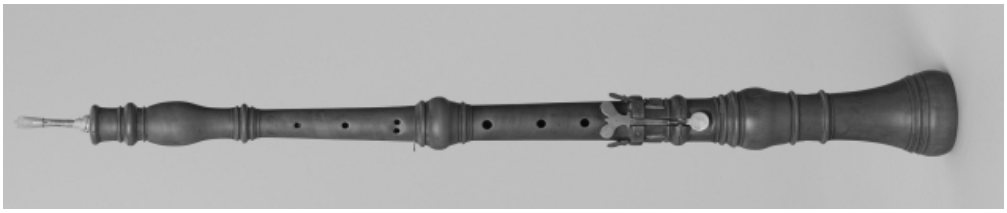


Figure 4.9: Some png-picture, directly included

Chapter 5

Processing of LaTeX main files

Given graphics in formats includeable in tex-files, which may require preprocessing described in Chapter 4, this section describes the conversions of latex main files into target files in detail. The most important target file formats is pdf. Conversion in this format is described in Section 5.1. Note that pdf also occurs as source format for included pictures and as intermediate files. Specific for LaTeX is the dvi format, which is supported mainly for historical reasons.

Almost independent of the format created, Inclusion of bibliographies, indices and glossaries requires additional conversions done by several auxiliary programs. Bibliographies are described in Section 5.2, indices in Section 5.3 and glossaries in Section 5.4. Sections 5.5 and 5.6 are special in that they describe rerunning several programs, which may be necessary even if certain lists are present like table of contents list of figures or list of tables.

Section **??** is special in that it is not related with conversion but with checking reproducibility.

Besides the output formats traditional for LaTeX, pdf and dvi describing e.g. books, Section 5.8 describes creation of html, Section 5.9 the creation of odt and Section 5.10 creation of MS word formats like docx. Finally, also pure text can be generated as described in Section 5.11.

5.1 Transforming LaTeX-files into pdf-files

The next step is to create a pdf-file from the tex-files. LaTeX distinguishes master tex-files from tex-files intended to be inputted from elsewhere. Not tanking comments and that like into account, master tex-files have the form

```
\RequirePackage[l2tabu, orthodox]{nag} % optional
\documentclass{...}

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

To satisfy this task, one may apply a LaTeX to pdf converter `latex2pdf` to a master tex-file `xxx.tex`. The LaTeX-to-pdf converter `latex2pdf` is configurable via the parameter `latex2pdfCommand`. Possible values are `pdflatex`, `lualatex` and `xelatex`, where the first is the default for which this software is also tested. It is also possible to pass parameters to the LaTeX to pdf converter.

In fact, the converter `latex2pdf` does much more than converting tex to pdf. Figure 5.1 shows for `latex2pdf` set to `pdflatex`, that besides the pdf-file also a log-file and an aux-file is created. The log-file contains logging information on the run of the conversion and the aux-file transports information from one run to the next. Thus conversion goes without it but it is read if present. This is why it is depicted at input side in dashed lines.

What is in fact in the aux-file depends on the package. Among other information, also citations and the location of the bibliography file with ending bib are present. This cannot be used directly in the next `latex2pdf` run to create the bibliography, because the entries must be retrieved from the bib-file, collected and sorted. This is done by invoking `bibtex` between two `latex2pdf` runs. Based on the aux-file, `bibtex` creates a bbl-file containing the bibliography, which is read in in the next `latex2pdf` run. For details see Section 5.2.

If an index is demanded, in addition `latex2pdf` creates an index file containing unsorted and multiple index entries (idx)-file. As the citations, it cannot be used directly to create an index in the next `latex2pdf` run, because the index entries must be collected and sorted before. This is done by invoking `makeindex` between the two `latex2pdf` runs. Based on the idx-file, `makeindex` creates an index file containing sorted, unified and formatted index entries (ind)-file containing the index, which is read in in the next `latex2pdf` run. For details see Section 5.3.

If a glossary is demanded, this can be read off the auxiliary file (aux)-file and a glossary file containing unsorted and multiple glossary entries (glo)-file is created. As the index, it cannot be used directly to create a glossary in the next `latex2pdf` run, because the glossary entries must be collected and sorted before. This is

done by invoking `makeglossaries` between the two `latex2pdf` runs. Based on the glo-file, `makeglossaries` creates a glossary file containing sorted, unified and formatted glossary entries (gls)-file containing the glossary, which is read in in the next `latex2pdf` run. For details see Section 5.4.

In addition, if a table of contents, a list of figures, a list of tables or a list of listings is required, also a toc-file, a lof-file, a lot-file and a lol-file is created, respectively, collecting the according information. If such a file is present, it is read in and is used to create a table of contents, a list of figures, of tables and of listings in the second run of `latex2pdf`.

To summarize, if a table of contents, a list of figures, a list of tables or a bibliography, an index or a glossary is present, a second LaTeX run is required to make them appear in the pdf-output.

If a table of contents and at the same time a bibliography, an index or a glossary is present, even two further LaTeX runs are required: After the first one, the bibliography, the index or the glossary occurs in the pdf-file but not yet in the table of contents. This happens after the second additional LaTeX run. As described in Sections 5.5 and 5.6, further runs of `makeindex`, resp, `splitindex`, `makeglossaries` and of the LaTeX-processor `latex2pdf` may be necessary.

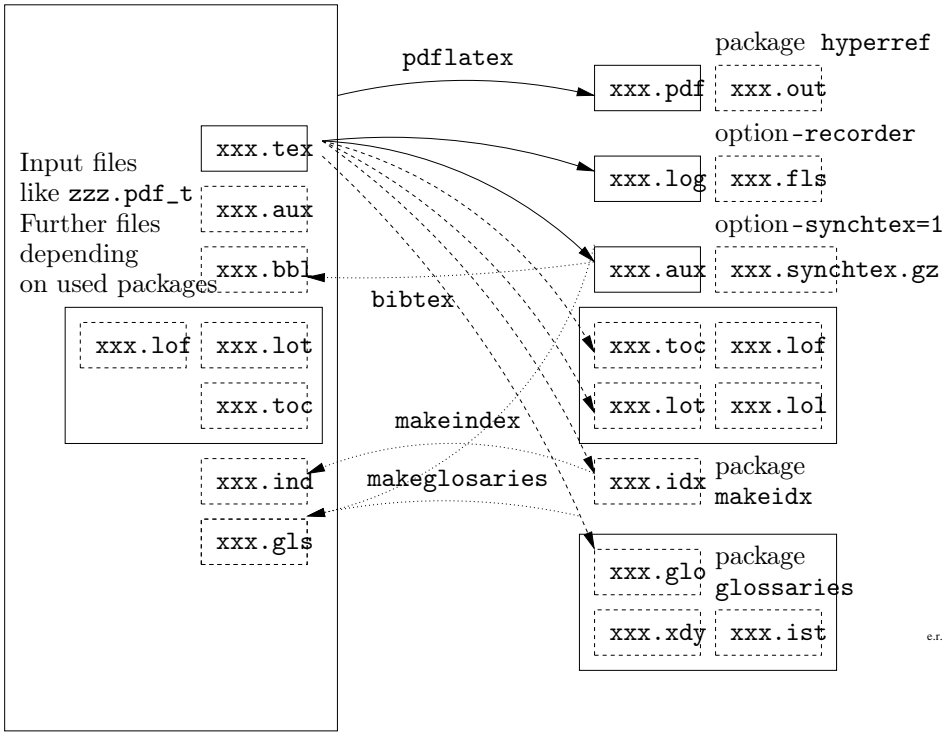


Figure 5.1: Conversion of a tex-file into a pdf-file (accordingly into a dvi-file)

5.2 Bibliographies

In case the LaTeX to pdf converter writes bibliographic information, into its aux-file, a bibliography must be generated. For each ocurrence of a `\cite`-command in the tex-file, `latex2pdf` writes an according entry `\citation`. Moreover, a `\bibliography`-command in the tex-file writes a link to the bib-file containing the bibliography data into the aux-file as `\bibdata`. Optionally, a `\bibliographystyle`-command in the tex-file writes a link to the bibliography style file into the aux-file as `\bibstyle`.

To create a bibliography, a `bibtexCommand` must be run after the LaTeX run. The default command is the traditional `bibtex` but there are more modern alternatives like `bibtexu` and `bibtex8` supporting utf8 encoding and others.

Essentially, `bibtex` extracts the citations in the aux-file, unifies them, i.e. a citation is listed once even if it is used more than once, retrieves the according entries from the bib-file, sorts these, performs formatting and writes all into a bbl-file which can be included in the next run of `latex2pdf`.

Note that after a `bibtex`-run, two LaTeX runs are required: The first one just puts the bibliography found in the bbl-file into the pdf-file and the labels of the citations into the aux-file as `\bibcite`-commands. The second run places the labels of the citations found in the aux-file at the citations given by `\cite`. The package `tocbibind` described in [WP10], then writes the headline of the bibliography into the table of contents. The package `rerunfilecheck` described in [Obe16c], ensures that `latex2pdf` is rerun if needed, provided loaded with option `aux`.

This software presupposes, that `bibtex` reads the aux-file and creates a bbl-file and also an blg-file with logging output as illustrated by Figure 5.2. From the blg-file this software may determine whether `bibtex` emitted an error or warnings.

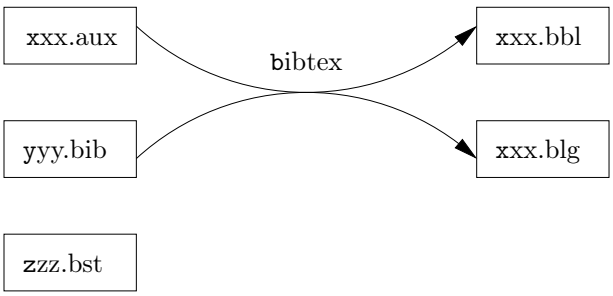


Figure 5.2: Conversion of an aux-file into a bbl-file using a bibliography

Vital information on `bibtex` can be found in [Pat88] and in [Mar09]. Also Chapter 10 in [Grä96] gives vital information on `bibtex`.

5.3 Indices

In case the LaTeX to pdf converter writes index information, into its idx-file, at least one index must be generated. Since the idx-file contains nothing but index information, an index is created if and only if the idx-file is created. Essentially, the command `\makeindex` tells `latex2pdf` to open the idx-file for writing. Then for each occurrence of the `\index`-command or similar (details see below) in the tex-file, an entry is written sequentially to the idx-file as `\indexentry` comprising the keyword given by the `\index`-command and the page number where the `\index`-command occured. For example `\index{ant-task}` creates an entry

```
\indexentry{ant-task|hyperpage}{3}
```

in `xxx.idx`.

Then the `makeindex`-command is applied to the idx-file which sorts keywords and for each keyword collects the according page numbers, sorts it and and writes the result into an ind-file. In the next run of `latex2pdf`, the `\prindindex`-command includes the index as a separate section; typically at the end of the pdf-file. The most basic package to provide this command is `makeidx` described in [BLC+14]. In addition, `makeidx` provides the command `\see` which is for cross reference within an index. The package `tocbibind` described in [WP10], then writes the headline of the index into the table of contents. The package `rerunfilecheck` described in [Obe16c], ensures that `latex2pdf` is rerun if needed, provided loaded with option `aux`.

The same document, [BLC+14] also describes the package `showidx` which prints index entries at the margin of the document. This is for debugging only.

The main restriction of the package `makeidx` is, that only a single index can be created. The reason is that, `latex2pdf` creates a single idx-file and, as illustrated in Figure 5.3, `makeindex` creates a single ind-file from that, representing a single index.

To overcome this restriction, replace package `makeidx` and `makeindex` with package `splitidx` and `splitindex` both described in [Koh16].

The package `splitidx` is used in conjunction with the program `splitindex`. It must be possible to create a single index without using `splitidx` and `splitindex`.
****

Package option `split` makes `latex2pdf` creating idx-files `xxx-y.idx` directly. Here `y` represents the identifier of an individual index. These idx-files can be transformed individually with `makeindex` into ind-files as illustrated in Figure 5.4. Since `latex2pdf` can keep open only up to 16 output streams, not all of which can be used to create a file `xxx-y.idx`, this approach allows a limited number of indices and is thus not recommended and not supported. Another reason is, that this approach undermines the package `rerunfilecheck` described in [Obe16c], and

so it is not guaranteed that `latex2pdf` is rerun if needed. This explains why option `split` is not allowed.

Instead, without option `split`, `latex2pdf` creates a single idx-file. The program `splitindex` splits it up into several idx-files and applies `makeindex` to each of them separately as illustrated in Figure 5.5.

For usage of further packages supporting multiple indices which are not intended to be used with this software, see Chapter 9.

This software presupposes, that `makeindex` converts the idx-file into an ind-file containing the index and creating also an ilg-file with logging output as shown in Figure 5.3. From the ilg-file this software may determine whether `makeindex` emitted an error or warnings.

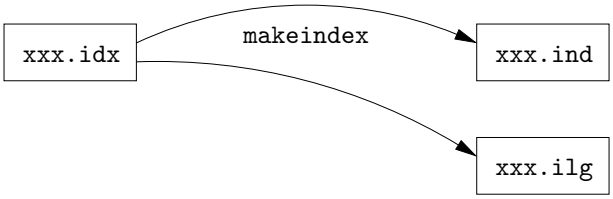


Figure 5.3: Conversion of an idx-file into an ind-file

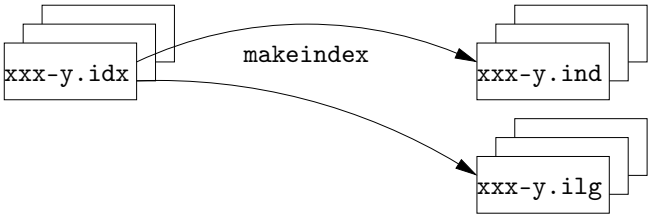


Figure 5.4: Not supported: Conversion of idx-files into ind-files

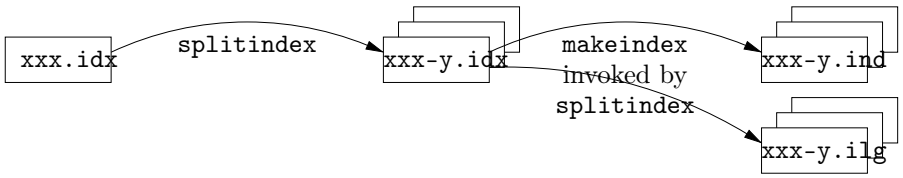


Figure 5.5: Conversion of an idx-file into ind-files

It is possible to configure the makeindex-command and to pass arbitrary options. CAUTION: For the usual `makeindex`-command, the options `-o` specifying an output file and `-t` (transcript) specifying the logging file are not allowed, because this breaks the expectation to find the sorted index in file `xxx.ind` and bypasses the

detection of errors and warnings of this software, respectively. Also specifying a style file via option `-s` is not recommended because this is used to create a glossary and so breaks glossary creation as described in Section 5.4.

Information on the makeindex program can be found in [Mös98] and in [Lam87]. Also there is a site [LRZ] describing all available options for `makeindex`.

As indicated above, the program `splitindex` invokes `makeindex`. Its options are described in [Koh16], Section 3.10. Since the long option names are not understood in all environments, only the short options are recommended.

Since `splitindex` must satisfy the interface given by Figure 5.5, the option `-help` and its shortcut `-h` are not allowed. Likewise for option `-version` and its shortcut `-V`. The option `-makeindex <makeindex>`, resp. `-m <makeindex>`, is used with the `makeindex` command used for single indices. Thus this may not be given explicitly but is specified implicitly. Also, the option `-identify <regex>`, resp. `-i <regex>` must be set implicitly because it must be the same expression as used to ***** Then splitindex.tlu is not allowed, because this has another expression.

Only allowable seems `-V`, the short cut for `-verbose`.

Then comes the name of the index file to be processed without suffix.

The program `splitindex` invokes `makeindex`. The option option `-` coming after the filename, indicates that all following options are passed to `makeindex`

5.4 Glossaries

Creating glossaries requires the package `glossaries` described in [Tal16b]. Note that despite of the headline of this section, an despite `glossaries` itself supports multiple glossaries, this software supports only a single glossary and also sorting and unifying is done either via `makeindex` as for indices or via `xindy`, whereas the option to do without external programs offered also by package `glossaries` is not supported by this software.

For generalizations see Chapter 9.

As for creating indices there is a LaTeX-command `\makeindex`, to create a glossary there is a LaTeX-command `\makeglossaries` but the latter is not builtin as `\makeindex` but provided by the package `glossaries`. If `xxx.tex` is the LaTeX main file, `\makeglossaries` opens the glo-file `xxx.glo` containing glossary entries for writing. As the builtin command `\index` writes entries into the idx-file defining the index, the command `\gls` defined by the package `glossaries` writes an entry into the glo-file. Note that `xxx.glo` typically contains entries more than once and that the entries are not sorted.

To perform sorting, formatting and typically also unification, the package `glossaries` allows three mechanisms. This software supports two of them: via the shell command `makeindex`, which is also used for indices, and via the shell

command `xindy`. Using `makeindex` is the default but can also be activated through `\usepackage[makeindex]{glossaries}`. Using `xindy` instead of `makeindex` is triggered through `\usepackage[xindy]{glossaries}`. Accordingly, for option `makeindex` the aux-file receives lines

```
\providecommand\@istfilename[1]{}
\@istfilename{manualLatexMavenPlugin.ist}
```

whereas for option `xindy`, there are lines

```
\providecommand\@istfilename[1]{}
\@istfilename{manualLatexMavenPlugin.xdy}
...
\providecommand\@xdylanguage[2]{}
\@xdylanguage{main}{english}
\providecommand\@gls@codepage[2]{}
\@gls@codepage{main}{}
```

This software neither invokes `makeindex` nor `xindy` directly. Instead it invokes the shell command `makeglossaries` invoked without file ending which determines from the aux-file whether to invoke `makeindex` nor `xindy`. Accordingly, it writes the style definition by creating an ist-file `xxx.ist` or an xdy-file `xxx.xdy` if `makeindex` or `xindy` is specified as package option, respectively.

Seemingly, `makeglossaries` relies on the aux-file to determine whether to invoke `makeindex` or `xindy` for sorting and unification. Then it invokes the according command and writes a log-file with ending `glg`, redirecting the logging output of `makeindex` or `xindy` adding own output so that a glg-file may be written, even if e.g. `makeindex` is invoked and does not. In any case, if the glg-file is written, `makeglossaries` writes text matching

```
(^\*\*\* unable to execute: )
```

in the glg-file if an error occurs, no matter whether `makeindex` or `xindy` is invoked. Possibly, there are cases where an error causes no glg-file to be written. If no error occurs, a glg-file is written and if warnings are emitted, they either come from `makeindex` or from `xindy`. Thus warnings may be detected with the patterns defined by `makeindex` and by `xindy`.

The style `list` (which is the default) is set in the form

```
\usepackage[style=list]{glossaries}
```

where [Tal16b], Section 15 lists predefined styles. So, the style determines the content of the style definition, whereas the options `makeindex` and `xindy` specify the form in which the style is encoded and thus the ending of the style file, which is either `ist` or `xdy`.

Sorting the glo-file, as said above, currently is only supported using the command `makeglossaries`. The allowed options are essentially those making sense for `makeindex` and those making sense for `xindy`. If the shell command `makeglossaries` invokes `makeindex` of course only the according options are passed supplemented by additional options `-s`, `-t`, `-o`, to specify the ist-file, the glg-file (the transcript-file) and the gls-file, respectively, which is the result of sorting, the output file, and contains the entries of the glo-file just sorted, formatted and unified. So for a tex main file `xxx.tex` `makeglossaries` invokes

```
makeindex  -s "xxx.ist" -t "xxx.glg" -o "xxx.gls" "xxx.glo"
```

Accordingly, if the shell command `makeglossaries` invokes `xindy` of course only the according options are passed supplemented by additional options `-M`, `-t`, `-o`. This is illustrated in Figure 5.6.

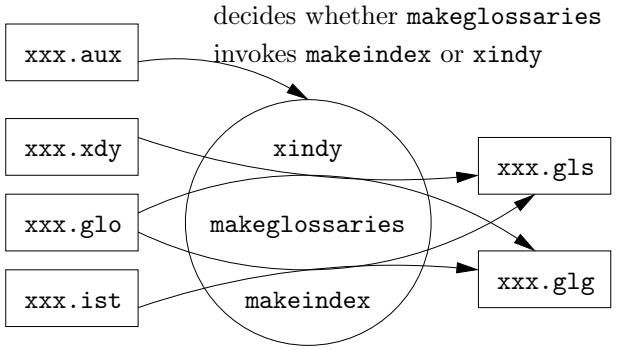


Figure 5.6: Conversion of a glo-file into a gls-file using `makeglossaries`

5.5 Rerunning the index- and glossary processor

As described in Section 5.1, running a LaTeX-to-pdf converter as `latex2pdf` may detect the presence of a bibliography, an index and/or of a glossary and writes raw files to describe them. After that, an intermediate step is required, sorting, unifying and formatting the entries. This is always done by an external program.

In the next step the LaTeX processor must read in the unified entries again. Whereas a LaTeX-run does not affect the bibliography, it may well invalidate the page numbers of the entries of the index or of the glossary. Thus the sorted index and glossary must be rebuild before the next LaTeX run makes them visible in the pdf-file.

To that end, we use the package `rerunfilecheck`.

```
\usepackage[index, glossary]{rerunfilecheck}
```

must be put before

```
\usepackage{makeidx}
\makeindex
\usepackage[toc, xindy]{glossaries}
%, xindy or even [xindy={language=english, codepage=utf8}]
% mainly for index and glossaries
\makeglossaries
```

in particular before `\makeindex` and `\makeglossaries`.

Note that the package `hyperref` already loads `rerunfilecheck` but with the wrong options. Thus the above declaration must come before

```
\usepackage[...]{hyperref}
```

to avoid error

```
Option clash for package rerunfilecheck
```

Package `rerunfilecheck` detects almost safely changes of the raw index file writing an according message into the log-file. That way, it can be determined whether it is necessary to rerun `makeindex` and `makeglossaries`. After that, of course LaTeX must be rerun at least once. Note that Section 5.6 describes when to rerun LaTeX without prior running *****

5.6 Rerunning the LaTeX processor

FIXME: a word on change in toc, lof, lot and lol.

As indicated in the previous sections, `latex2pdf` must be rerun, if `bibtex` or `makeindex` `splitindex` or `makeglossaries` had been run to read in the bibliography created by `bibtex` or the index created by `makeindex` or the glossary created by `makeglossaries`. Likewise, if a toc-file, a lof-file, a lot-file or a lol-file had been created in the first `latex2pdf` run, another run is needed to read in these files to create a table of contents, a list of figures or a list of tables, respectively. Note that for all these cases, the log-file does not allow to detect that `latex2pdf` has to be rerun, by matching a fixed pattern.

After the second run of `latex2pdf`, the table of contents, the list of figures, the list of tables and the list of listings are included and a section with the bibliography, the index and the glossary are inserted. It takes a third run of `latex2pdf` to include the bibliography the index and the glossary into the table of contents. Also it takes that third run to replace the citations with the proper labels given in the bibliography.

Inserting the table of contents, the list of figures, the list of tables and the list of listings may shift the subsequent text which may require another run of `latex2pdf`

to get the page numbers right. As described in Section 5.5 intermediate runs of `makeindex` and `makeglossaries` may be required and these also require another run of `latex2pdf` also to get the page numbers right. The package `rerunfilecheck` allows to detect this need to rerun by pattern matching on the log-file almost for sure: Still there is some chance that the lengths and the md5-sum of all relevant files remain the same, although there is a relevant change. In this case, this software fails to update triggering another `latex2pdf` run.

Note that there are several packages which require additional runs, such as the longtable-package, which may vary dimensions of tables. This software presupposes, that all these reruns may be detected by matching a fixed pattern in the log-file. Since packages are frequently changed and new packages are written, also the pattern cannot be fixed. Thus it is configurable.

Note that, if a package requires running other programs between two runs of `latex2pdf`, this would require a change in this software.

5.7 Checking reproducibility

There are use cases, where on the one hand side we want to deliver the sources but it is extremely important that the according artifacts are really reproducible. One obvious case is integration test for this software by ensuring that the created artifacts is equivalent with with a confirmed version.

Currently, this is done for pdf files only. The problem with pdf files is, that besides visible contents it contains also meta-data (see [PDF], Section 14.3), which depends on the run of the conversion. For example the timestamp of conversion goes into and so do many more aspects.

There are two strategies to deal with the problem:

- make the build process reproducible. The advantage of this is that diffing is quite simple, fast and reproducible: it is byte by byte. This is easily done with a fixed installation but tends to break with update of tools. Also at time of this writing, the different latex engines cannot be treated uniformly.
- use diff tools implementing a weaker notion of equivalence, in a sense visibility equivalence of some degree.

We support both approaches. The first one imposes requirements on the tex source file. It excludes that one displays a creation date changing each run. Listing 5.1 lower part illustrates that: We replaced the creation date by a “version date”. A date can be only date of checkin of a version control system or that like. As an alternative, one could also refrain from using dates altogether. In the `\date` command one could display also other pieces of information identifying the document.

But the visible date is not all we have to avoid. Listing 5.1 shows how to eliminate metadata and to replace with adequate one. There is metadata which can be eliminated in a uniform way for all latex engines using the package `hyperref` which the author generally advises. Using`hyperref` with `\hypersetup` one can set metadata such as author, title, subject and keyword as described in [RO22], Section 5.8. On the other hand, one could eliminate `CreationDate` and `ModDate`. For newer versions of `pdflatex` this can be done alternatively with `\pdfinfoomitdate1`.

The metadata which can be eliminated only in a engine specific way are the PTEX keys including the banner, and the `pdftrailerid`. Note that the latter is *intentionally* a unique identifier for the individual document and so strictly speaking it is not advised to eliminate this but eliminating is necessary because it is incompatible with reproducibility. Note that `xelatex` seems not to write PTEX keys at all so only the `pdftrailerid` has to be eliminated or fixed.

If a user renounces independence of the engine and sticks to `pdflatex` only, the package `hyperref` could be replaced by specific commands. For convenience also `pdfprivacy` described in [Sio17] can be used.

Now that we have described how to ensure reproducible pdf artifacts, just by designing the tex source appropriately, we have to explain how to check that a newly created artifact coincides with a blueprint provided a priori. First of all, note that currently such a check is performed only if the result is in pdf format. Even then the check is performed only if configured so. Then the actual artifacts are compared to predefined artifacts using a diff-tool. If the actual artifacts do not coincide with predefined ones according to the chosen diff tool, a build exception is thrown as specified in Table 7.7.

5.8 Creating hypertext

To create html and xhtml from LaTeX-files, a `tex4htCommand`-command is used Together with its parameters, it is described in Section!6.7. This may be `htlatex`, the default based on `latex` and `htxelatex` based on `xelatex`.

Figure 5.7 shows the steps `htlatex` performs: From the input LaTeX-file `xxx.tex` another LaTeX-file `yyy.tex` is created which arises from `xxx.tex` by adding

```
\usepackage[...]{tex4ht}.
```

Then `htlatex` runs `latex` on `yyy.tex` which results in `yyy.dvi`. Note that this is in contrast to `pdflatex` which would create some `yyy.pdf` unless otherwise specified.

Then comes the converter `tex4ht` into the game which creates several html-files among those also `xxx.html`. The other files, `yyy.idv` and `yyy.lg`, are further processed by `t4ht` creating the stylesheet `xxx.css` and graphic files.

```
% for pdflatex
\pdfsuppressptexinfo-1
% pdfsuppressptexinfo-1 is the same as pdfsuppressptexinfo15
% 1 -> PTEX.Fullbanner
% 2 -> PTEX.FileName
% 4 -> PTEX.PageNumber
% 8 -> PTEX.InfoDict (/Producer /Creator /CreationDate /ModDate /Trapped)
\pdftrailerid{}

% for lualatex:
%   1 -> PTEX.FullBanner
%   2 -> PTEX.FileName
%   4 -> PTEX.PageNumber
%   8 -> PTEX.InfoDict
%  16 -> Creator
%  32 -> CreationDate
%  64 -> ModDate
% 128 -> Producer
% 256 -> Trapped
% 512 -> ID
\pdfvariable suppressoptionalinfo \numexpr32+64+512\relax

% for xelatex
\special{pdf:trailerid [
    <00112233445566778899aabbccddeeff>
    <00112233445566778899aabbccddeeff>
]}

% general
\usepackage{hyperref}
...
\hypersetup{
  pdfinfo={
    Author      ={Ernst Reissner},
    Title       ={The dvi-format and the program dvitype},
    CreationDate={unknown},
    ModDate     ={unknown},
    Subject     ={dvi and dvitype},
    Keywords    ={LaTeX;dvi;dvitype}
  }
}

% alternative for pdftex only
% \pdfinfoomitdate1
% \pdfsuppressptexinfo-1
% \pdftrailerid{}
% \pdfinfo{
%   /Author       (Ernst Reissner)
%   /Title        (The dvi-format and the program dvitype)
%   /CreationDate (unknown)
%   /ModDate      (unknown)
%   /Subject      (dvi and dvitype)
%   /Keywords     (LaTeX;dvi;dvitype)
%}
% Replacing pdfinfoomitdate1 in conjunction with
% usepackage[nodocdata=true,nopdftrailerid=true]{pdfprivacy}
% alternative 1 for pdftex only


\title{The dvi-format and the program dvitype}
\author{Ernst Reissner (rei3ner@arcor.de)}
```

Let us make this more precise. The output of latex is a standard dvi-file interleaved with special instructions for the post-processor `tex4ht` to use. Note that `tex4ht` is the name both of the post-processor and of the LaTeX-package. The special instructions come from implicit and explicit requests made in the source file through commands for TeX4ht.

The utility `tex4ht` translates the dvi-code into standard text, while obeying the requests it gets from the special instructions. The special instructions may request the creation of files, insertion of html code, filtering of pictures, and so forth. In the extreme case that the source code contains no commands of TeX4ht, `tex4ht` gets pure dvi-code and it outputs (almost) plain text with no hypertext elements in it.

The special (`\special`) instructions seeded in the dvi-code are not understood by dvi processors other than those of TeX4ht.

`t4ht` This is an interpreter for executing the requests made in the `xxx.lg` script.

`xxx.idv` This is a dvi-file extracted from `xxx.dvi`, and it contains the pictures needed in the html files.

`xxx.lg` This is a log file listing the pictures of `xxx.idv`, the png files that should be created, CSS information, and user directives introduced through the "`\Needs{...}`" command.

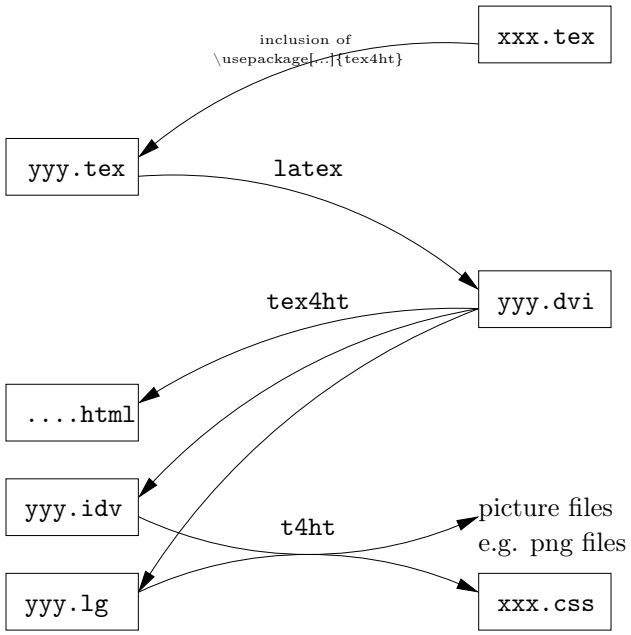


Figure 5.7: Conversion of a tex-file into an xml-file

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/tex4ht.4ht
version 2009-01-07-07:11

```
--------------------------------------
Note --- for additional information, use the command line option ‘info’
--------------------------------------

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht

Note: to remove the <?xml version=...?> processing instruction
use the command line option ‘no-VERSION’

Note: to remove the DOCTYPE declaration
use the command line option ‘no-DOCTYPE’
)

--------------------------------------
Note: for marking of the base font, use the command line option ‘fonts+’
Note: for non active _, use the command line option ‘no_’
Note: for _ of catcode 13, use the command line option ‘_13’
Note: for non active ^, use the command line option ‘no^’
Note: for ^ of catcode 13, use the command line option ‘^13’
--------------------------------------

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht
--------------------------------------
Note: For section filenames that reflect on their titles
use the command line option ‘sec-filename’

Note: for alternative charset, use the command line option ‘charset=...’

Note: to ignore CSS font decoration, use the ‘NoFonts’ command line option

Note: for jpg bitmaps of pictures,
use the ‘jpg’ command line option.
(Character bitmaps are controled only by ‘g’
records of tex4ht.env and ‘-g’ switches of tex4ht.c)

Note: for gif bitmaps of pictures, use the ‘gif’ command line option.
(Character bitmaps are controled only by ‘g’
records of tex4ht.env and ‘-g’ switches of tex4ht.c)

Note: for content and toc in 2 frames,
use the command line option ‘frames’

Note: for content, toc, and footnotes in 3 frames,
use the command line option ‘frames-fn’

Note --- for file extension name xht, use the command line option ‘xht’
--------------------------------------
TeX4ht package options: xhtml,uni-html4,2,pic-tabular,html
--------------------------------------
Note: to ignore CSS code, use the command line option ‘-css

Note: for inline CSS code, use the command line option ‘css-in’

Note: for pop ups on mouse over, use the command line option ‘mouseover’

Note: for addressing images in a subdirectory,
use the command line option ‘imgdir:.../’
)

Note --- for back links to toc, use the command line option ‘sections+’

Note --- for linear crosslinks of pages, use the command line option ‘next’

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/latex.4ht
version 2009-05-21-09:32
--------------------------------------
Note --- for links into captions, instead of float heads, use the command l
ine option ‘refcaption’
--------------------------------------

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht
--------------------------------------
Note --- For mini tocs immediately aftter the header
use the command line option ‘minitoc<’

Note --- for enumerated list elements with valued data,
use the command line option ‘enumerate+’

Note --- for enumerated list elements li’s with value attributes, use the c
ommand line option ‘enumerate-’

Note --- for CSS2 code, use the command line option ‘css2’
```

```
Note --- for bitmap fbox'es, use the command line option 'pic-fbox'

Note --- for bitmap framebox'es, use the command line option 'pic-framebox'

Note --- for inline footnotes use command line option 'fn-in'

Note --- for tracing of latex font commands,
use the command line option 'fonts'
----------------------------------------
----------------------------------------
Note --- for width specifications of tabular p entries,
use the 'p-width' command line option
or a configuration similar to
\Configure{HColWidth}{\HCode{style="width:\HColWidth"}}
----------------------------------------
)
(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4-math.4ht
version 2009-05-18-23:01
----------------------------------------
Note --- for pictorial eqnarray, use the command line option 'pic-eqnarray'

Note --- for pictorial array, use the command line option 'pic-array'

Note --- for pictorial $...$ environments,
use the command line option 'pic-m' (not recommended!!)

Note --- for pictorial $...$ and $$...$$ environments with latex alt,
use the command line option 'pic-m+' (not safe!!)

Note --- for pictorial array, use the command line option 'pic-array'
)
(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/unicode.4ht
version 2010-12-18-17:40
)
(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4-uni.4ht))


(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht
----------------------------------------
Note --- for tocs without * entries, use command line option 'notoc*'

Note --- for tocs without * entries, use command line option 'notoc*'

Note --- to eliminate mini tables of contents,
use the command line option 'nominitoc'

Note --- for frames-like object-based table of contents,
use the command line option 'obj-toc'

Note --- for files named derived from section titles,
use the command line option 'sec-filename'

Note --- for i-columns index,
use the command line option 'index=i' (e.g., index=2)
----------------------------------------
)

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht

Note --- if included graphics are of degraded quality,
try the command line options 'graphics-num' or 'graphics-'.
The 'num' should provide the density of pixels in the bitmaps (e.g., 110).

Note --- for key dimensions try the option 'Gin-dim';
for key dimensions when bounding box is unavailable
try 'Gin-dim+'; neither is recommended
)

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht
Note --- for URL encoding within href use the command line option 'url-enc'
)

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht

Note --- for pictorial longtable,
use the command line option 'pic-longtable'
)

(/usr/local/texlive/2014/texmf-dist/tex/generic/tex4ht/html4.4ht

Note --- to ensure proper alignments use fixed size fonts (see listings.dtx
)
)
```

`tex4ht` yields

```
----------------------------
tex4ht.c (2012-07-25-19:36 kpathsea)
tex4ht
--- error --- improper command line
tex4ht [-f<path-separator-ch>]in-file[.dvi]
   [-.<ext>]            replacement to default file extension name .dvi
   [-c<tag name>]       choose named segment in env file
   [-e<env-file>]
   [-f<path-separator-ch>]        remove path from the file name
   [-F<ch-code>]        replacement for missing font characters; 0--255; default 0
   [-g<bitmap-file-ext>]
   [-h(e|f|F|g|s|v|V)]  trace: e-errors/warnings, f-htf, F-htf search
                            g-groups, s-specials, v-env, V-env search
   [-i<htf-font-dir>]
   [-l<bookkeeping-file>]
   [-P(*|<filter>)]     permission for system calls: *-always, filter
   [-S<image-script>]
   [-s<css-file-ext>]   default: -s4cs; multiple entries allowed
   [-t<tfm-font-dir>]
   [-u10]               base 10 for unicode characters
   [-utf8]              utf-8 encoding for unicode characters
   [-v<idv version>]    replacement for the given dvi version
   [-xs]           ms-dos file names for automatically generated gifs
```

`t4ht` yields

```
----------------------------------------------------------------------
t4ht [-f<dir char>]filename ...
  -b     ignore -d -m -M for bitmaps
  -c...  choose named segment in env file
  -d...  directory for output files       (default:  current)
  -e...  location of tex4ht.env
  -i     debugging info
  -g     ignore errors in system calls
  -m...  chmod ... of new output files (reused bitmaps excluded)
  -p     don't convert pictures           (default:  convert)
  -r     replace bitmaps of all glyphs    (default:  reuse old ones)
  -M...  chmod ... of all output files
  -Q     quit, if tex4ht.c had problems
  -S...  permission for system calls: *-always, filter
  -X...  content for field %%3 in X scripts
  -....  content for field %%2 in . scripts

Example:
   t4ht name -d/WWW/temp/ -etex4ht-32.env -m644
----------------------------------------------------------------------
```

5.9 Creating odt-files

5.10 Creating MS word files

The best way to convert LaTeX-files into MS word files is via odt files. Conversion from LaTeX to odt is already described in Section 5.9. The last step can be done by `odt2doc` which can create both doc-format and docx-format and many others which is illustrated in Figure 5.8.

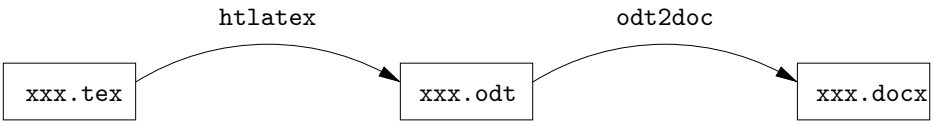


Figure 5.8: Conversion of a tex-file into a docx-file

5.11 Creating plain text files

Why should one create plain text from LaTeX-files? Maybe this is the minimal format the receiver can work with. Another common application is word-count, in particular if writing a paper for a journal.

Plain text files can be created from LaTeX-files just by stripping off the tex-commands. The disadvantage is, that references, bibliography, index, glossary, table of contents, list of figures, list of tables, ... and symbols get lost. Thus, the first step we take is complete creation of a pdf-file except display of warnings like bad boxes as described in Section 5.1. This creates an appropriate pdf-file, with correct numberings and links, possibly with overfull boxes and that like. As a final step, we convert the pdf-file into a text file using, as a default `pdftotext` with ending `txt`. Figure 5.9 illustrates the translation process.

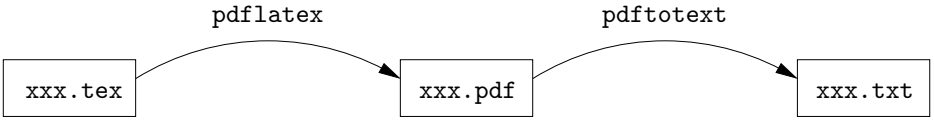
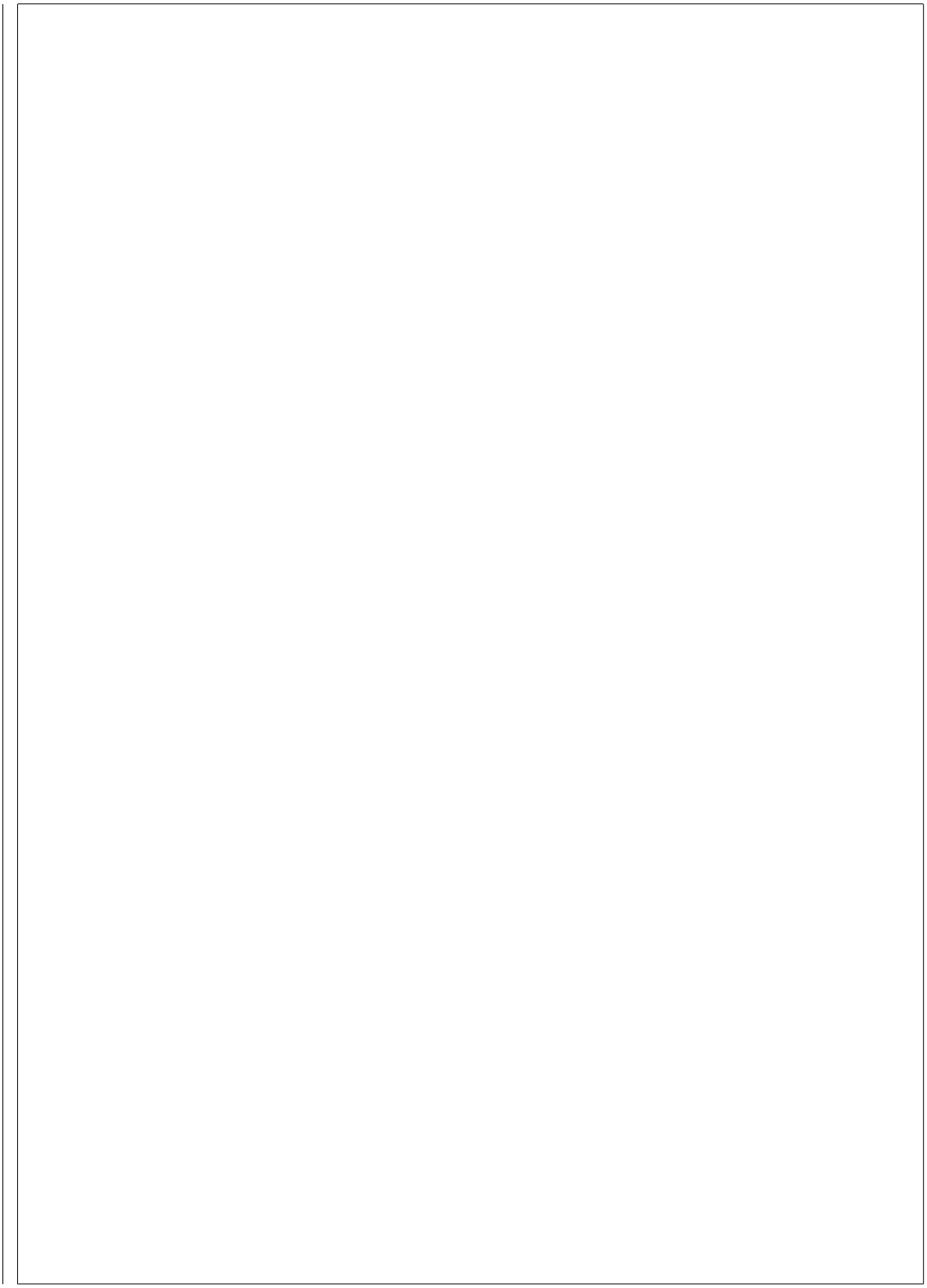


Figure 5.9: Conversion of a tex-file into an txt-file

Note that `pdftotext` produces a text file with page numbers and signifies the end of a page (to see how, just have a look at the end of the file), so that one can identify page numbers as such. Thus references, index, glossary, table of contents and that like referring to page numbers carry valuable information. Also symbols available in utf8 encoding are preserved. In contrast, heavily stacked

formulae become unreadable, because `pdftotext` displays them line by line and drops fraction bars completely. Also formulae with complex subformulae in a root operator become unreadable because the root operator becomes just a root symbol. Likewise for integrals and that like.

Aspects of figures kept are the captions of course but also the LaTeX-texts. This is displayed line-wise. What gets lost is the postscript/pdf-parts, i.e. the plain graphics.



Chapter 6

Parameters resp. Settings

This section describes the parameters of both the ant-task and the maven-plugin. The parameters are listed in Tables 6.1 through 6.8 with names, default values and short explanations. Note that neither of the parameters is mandatory, as there are always valid default values.

Each of the tables is described in a separate section. Table 6.1 shows parameter controlling the general conversion process described in detail in Section 6.1. These are directories with names `xxxDirectory` and further parameters not following a naming convention. The other tables show parameters after a certain naming scheme: Command names have the form `xxxCommand` and the parameter with the according options have the form `xxxOptions`. Here `xxx` represents a certain converter. This is one of

fig2dev	The converter of fig-files into mixed latex- and pdf-files.
gnuplot	The converter of gnuplot-files into mixed latex- and pdf-files.
metapost	The converter of metapost-files into mixed latex- and pdf-files.
latex2pdf	The converter of latex-files into pdf-files.
bibtex	The creator of a bibliography from an aux-file.
MakeIndex	The makeindex utility creating an index.
MakeGlossaries	The makeglossaries utility creating a glossary.
tex4ht	The converter of latex into html and also into odt, depending on the parameters.
latex2rtf	The converter of latex-files into rtf-files.

odt2doc The converter of odt-files into doc(x)-files.

pdf2txt The converter of pdf-files into txt-files.

chktex A code-checker converting in a sense a latex main file into a log-file containing errors, warnings and further messages.

It is a little more complicated with the parameters in Section 6.7.

There are some parameters of the form `patternXxxYyy`, referring to a pattern in the log-file of the converter `Xxx` indicating an event `Yyy` which is one of the following:

ReRun indicates that `Xxx` needs to be rerun.

Err indicates that `Xxx` had an error.

Warn indicates that `Xxx` had a warning.

Essentially, there are two kinds of parameters: Most are just passed to the converters invoked by this software. The parameters of this software are so that the choice of the converter, i.e. the name of the application can be configured and also each converter can be almost freely configured.

Parameters not passed to an application, are either really crucial or are included to allow also development of latex files.

6.1 General Parameters

This section describes the parameters of Table 6.1.

Parameter Explanation	Default
`versionsWarnOnly`	`false`

Indicates whether the `VersionMojo` displays warnings only or also creates infos. Infos refer to the version of this plugin and also on its git commit, but also on the versions of the converters found and lists the converters excluded, i.e. those not used and thus not tested on version.
Warnings are emitted e.g. if the version of a converter does not fit the expectations, the version of a converter could not be retrieved, e.g. because it is not installed or if the converter specified is unknown altogether. This defaults to `false` displaying also info.
The latter is appropriate for using in command line `mvn latex:vrs`, whereas in builds by default the pom overwrites this to have output only in case something goes wrong.

`texSrcDirectory` `src/site/tex`
The latex source directory as a string relative to `$baseDirectory`, containing `$texSrcProcDirectory`. This directory determines also the sub-directory of `$outputDirectory` to lay down the generated artifacts. The default value is "`src/site/tex`" on Unix systems.

`texSrcProcDirectory` `.`
The latex source processing directory as a string relative to `$texSrcDirectory` containing all tex main documents and the graphic files to be processed and also to be cleaned. Whether this is done recursively in sub-folders is specified by `$readTexSrcProcDirRec`. The default value is "`.`".

`readTexSrcProcDirRec` `true`
Whether the tex source directory `$texSrcProcDirectory` shall be read recursively for creation of graphic files, i.e. including the sub-directories recursively. This is set to `false` only during development of documentation.

`outputDirectory` `.`
The generated artifacts will be copied to `outputDirectory` relative to `$targetSiteDirectory` which is by default '`$targetDirectory/site`' on Unix systems.

`targets` `pdf, html`
A comma separated list of targets to be stored in `$targetSet`.

`convertersExcluded` empty
A comma separated list of excluded @link Converters given by their command. Excluded converters need not be installed but their names must be known. They don't show up in the version check of target 'vrs' and of course they are not allowed to be used.

`patternLatexMainFile` see Section 6.1.1

The pattern to be applied to the contents of tex-files which identifies a LaTeX main file. Here we assume that the latex main file should contain the declaration "`\documentclass`" or the old fashioned "`\documentstyle`" preceded by a few constructs. Strictly speaking, this is not necessary. For a more thorough discussion, and for an alternative approach, consult the manual.
The default value is chosen to match quite exactly the latex main files, no more no less. Since the pattern is chosen according to documentation collected from the internet, one can never be sure whether the pattern is perfect.
If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.

`texPath` empty string
Path to the TeX scripts or null. In the latter case, the scripts must be on the system path. Note that in the pom, <`texPath/`> and even <`texPath`> </`texPath`> represent the null-File. The default value is null.

`cleanUp` `true`
Clean up the working directory in the end? May be used for debugging when setting `false`.

`patternCreatedFromLatexMain` see Section 6.1.2
This pattern is applied to file names and matching shall accept all the files which were created from a latex main file '`xxx.tex`'. It is neither applied to directories nor to '`xxx.tex`' itself. It shall not comprise neither graphic files to be processed nor files created from those graphic files.
This pattern is applied in the course of processing graphic files to decide which graphic files should be processed (those rejected by this pattern) and to log warnings if there is a risk, that graphic files to be processed are skipped or that processing a latex main file overwrites the result of graphic preprocessing.
When clearing the tex source processing directory `$texSrcProcDirectory`, i.e. all generated files should be removed, first those created from latex main files. As an approximation, those are removed which match this pattern.
The sequence '`T$T`' is replaced by the prefix '`xxx`'. The sequence '`T$T`' must always be replaced: The symbol '`$`' occurs as end-sign as '`)$`' or as literal symbol as '`$`'. Thus '`T$T`' is no regular occurrence and must always be replaced with '`xxx`'.
Spaces and newlines are removed from that pattern before matching.
This pattern may never be ensured to be complete, because any package may create files with names matching its own patterns and so any new package may break completeness. Nevertheless, the default value aims completeness while be tight enough not to match names of files not created.
If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.

Table 6.1: General parameters

6.1.1 The parameter patternLatexMainFile

The default pattern for identifying a LaTeX main file is

```
\A(\\RequirePackage\s*(\[(\s|\w|,)*\])?\s*\{\w+\}\s*(\[(\d|\.)+\])?|
\\PassOptionsToPackage\s*\{\w+\}\s*\{\w+\}|
%.*$|
\\input\{[^{}]*\}|
\s*)*
\\(documentstyle|documentclass)
```

This means that the latex main file is the one containing `\documentclass` for modern classes and `\documentstyle` for old fashioned ones. This may be preceded only by

- the command `\RequirePackage`,
- the command `\PassOptionsToPackage`,
- by comment lines,
- by `\input`, and
- by space.

Note that `\A` represents the beginning of the stream.

Strictly speaking, also `\(re)newcommands` and other constructs are possible but not allowed here. Strictly speaking also `\documentclass` is not required, it could be hidden in some `\input`-file. If we stick to the wise convention to open and close the same environment in the same file and to have `\documentclass`, `\begin{document}` and `\end{document}` in the same file also, then a latex main file without `\documentclass` cannot contain very much material.

So we ask the user to have `\documentclass` declared in the latex main file.

For users of Emacs with package auctex, there is a valuable alternative:

Latex main files are marked with an end section as this file:

```
%%% Local Variables:
%%% mode: latex
%%% TeX-command-extra-options: "-src-specials -recorder -shell-escape"
%%% TeX-master: t
%%% End:
```

The vital line in this context is `%%% TeX-master: t`. In contrast to this, a non-master file either has no end-section at all or has an end section declaring the according master file (if it is unique) explicitly as the following one from `header.tex`:

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: "manualLatexMavenPlugin"
%%% End:
```

So a pattern for latex main files could also be

```
^%%% Local Variables: $
^%%% TeX-master: t$
^%%% End: ^
\s*
```

The problem is only in case of exchange of the source with a developer which does not use Emacs/auctex.

6.1.2 The parameter `patternCreatedFromLatexMain`

The files created from a latex main file depend strongly on the compiler options and on packages used in the latex main file and in the tex files inputted. The default value '`^T$T\.[^.]*`' is appropriate for most parameters and packages: most packages create files with names only which coincide with the name of the latex main file, except the suffix. This is all sufficient even for programs doing postprocessing such as `bibtex`, `makeindex`, `xindy` and `makeglossaries`. The only exception is `splitindex` which requires in addition '`^T$T\-.+(idx|ind|ilg)`'.

Package '`srcltx`' requires in addition '`^T$T\.synctex\.gz`' and finally package '`tex4ht`' is for all the rest. The pattern is designed to match quite exactly the created files, not much more and at any case not less. In particular it has to comprise the files matching pattern `$patternT4htOutputFiles`. Nevertheless, since any new package may break the pattern, and not every package is well documented, this pattern cannot be guaranteed to be final.

If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.

The default value for this pattern is currently:

```
^(
T$T             (\.([^.]*|synctex\.gz|out\.ps)|
   (-|ch|se|su|ap|li)?\d+\.x?html?|
            \d+x\.x?bb|
            \d+x?\.png|
            -\d+\.svg)|
zzT$T\.e?ps|
(cmsy)\d+(-c)?-\d+c?\.png|
(pdf)?latex\d+\.fls)$
```

6.2 Parameters for graphical preprocessing

This section describes the parameters for graphical preprocessing given in Table 6.2.

TODO: do this.

Parameter Explanation	Default
`fig2devCommand`	`fig2dev`
The fig2dev command for conversion of fig-files into various formats. Currently only `pdf` combined with `ptx` is supported.	
`fig2devGenOptions`	empty
The options for the command `$fig2devCommand` common to both output languages. For the options specific for the two output languages '`pdftex`' and '`pdftex_t`', see the explanation of the parameters `$fig2devPtxOptions` and `$fig2devPdfEpsOptions`, respectively.	
`fig2devPtxOptions`	empty
The options for the command `$fig2devCommand` specific for the output language '`pdftex_t`'. Note that in addition to these options, the option '`-L pdftex_t`' specifies the language, `$fig2devGenOptions` specifies the options common for the two output languages '`pdftex`' and '`pdftex_t`' and '`-p xxx.pdf`' specifies the pdf-file to be included.	
`fig2devPdfEpsOptions`	empty
The options for the command `$fig2devCommand` specific for the output language '`pdftex`'. Note that in addition to these option1s, the option '`-L pdftex`' specifies the language and `$fig2devGenOptions` specifies the options common for the two output languages '`pdftex`' and '`pdftex_t`'.	
`gnuplotCommand`	`gnuplot`
The command for conversion of gnuplot-files into various formats. Currently only pdf (graphics) combined with pdf_t (latex-texts) is supported.	
`gnuplotOptions`	empty
The options specific for `$gnuplotCommand`'s output terminal "cairolatex", used for mixed latex/pdf-creation. Note that the option '`pdf|eps`' of the terminal '`cairolatex`' is not available, because it is set internally.	
`metapostCommand`	`mpost`
The command for conversion of gnuplot-files into metapost's postscript.	
`metapostOptions`	see Section 6.2.1
The options for the command `$metapostCommand`. Leading and trailing blanks are ignored. A sequence of at least one blank separate the proper options.	
`svg2devCommand`	`inkscape`
The command for conversion of svg-files into a mixed format.	
`svg2devOptions`	`-z -D -export-latex`

The options for the command `$svg2devCommand` for exporting svg-figures into latex compatible files. For more details see Section 6.2.2.	
`ebbCommand`	`ebb`
The command to create bounding box information from jpg-files and from png-files. This is run twice: once with parameter '`-m`' to create '.bb'-files for driver '`dvipdfm`' and once with parameter '`-x`' to create '.xbb'-files for driver '`dvipdfmx`'.	
`ebbOptions`	`-v`
The options for the command `$ebbCommand` except '`-m`' and '`-x`' which are added automatically.	

Table 6.2: Parameters for graphics preprocessing

6.2.1 The parameter `metapostOptions`

TODO: add

6.2.2 The parameter `svg2devOptions`

The following options are mandatory:

- '-z' or '–without-gui' process files from console.
- '-D' or '–export-area-drawing' Export the drawing (not the page)
- '–export-latex' Export into PDF/PS/EPS format without text. Besides the PDF/PS/EPS files, a LaTeX-file `latexfile.tex` is exported, putting the text on top of the PDF/PS/EPS file. Include the result in LaTeX as: `\input{latexfile.tex}`.

  Note that the latter option is necessary, to create the expected files. It is also conceivable to export text as pdf/eps

The following options are prohibited, because they are automatically added by the software:

- '-A=FILENAME' or '–export-pdf=FILENAME' Export document to a PDF file.
- '-E=FILENAME' or '–export-eps=FILENAME' Export document to an EPS file.

6.3 Parameters for the LaTeX-to-pdf Conversion

This section describes the parameters of the LaTeX-to-pdf converter which are given in Table 6.3.

TODO: do this.

Parameter Explanation	Default
`latex2pdfCommand`	`pdflatex`
The LaTeX command to create a pdf-file with.	
`latex2pdfOptions`	see Section 6.3.1
The options for the command `$latex2pdfCommand`. Leading and trailing blanks are ignored. A sequence of at least one blank separate the proper options. FIXME: -output-format is not allowed because set automatically.	
`patternErrLatex`	`(^! )`
The pattern is applied line-wise to the log-file and matching indicating an error emitted by the command `$latex2pdfCommand`. The default value is chosen to match quite exactly the latex errors in the log file, no more no less. Since no official documentation was found, the default pattern may be incomplete. In fact it presupposes, that `$latex2pdfOptions` does not contain "`-file-line-error-style`". If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.	
`patternWarnLatex`	see Section 6.3.2
The pattern is applied line-wise to the log-file and matching indicates a warning emitted by the command `$latex2pdfCommand`, disregarding warnings on bad boxes provided `$debugWarnings` is set. This pattern may never be ensured to be complete, because any package may indicate a warning with its own pattern any new package may break completeness. Nevertheless, the default value aims completeness while be restrictive enough not to indicate a warning where none was emitted. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.	
`debugBadBoxes`	`true`
Whether debugging of overfull/underfull hboxes/vboxes is on: If so, a bad box occurs in the last LaTeX run, a warning is displayed. For details, set `$cleanUp` to false, rerun LaTeX and have a look at the log-file.	
`debugWarnings`	`true`
Whether debugging of warnings is on: If so, a warning in the last LaTeX run is displayed. For details, set `$cleanUp` to false, rerun LaTeX and have a look at the log-file.	
`pdfViaDvi`	`false`

Whether creation of pdf-files from LaTeX-files goes via dvi-files. If `$pdfViaDvi` is set and the latex processor needs repetitions, these are all done creating dvi and then pdf is created in a final step invoking the command `$dvi2pdfCommand` If `$pdfViaDvi` is not set, latex is directly converted into pdf. Currently, not only conversion of LaTeX-files is affected, but also conversion of graphic files into graphic formats which allow inclusion in the tex-file. If it goes via latex, then the formats are more based on (encapsulated) postscript; else on pdf. Of course, the target dvi is not affected: This uses always the dvi-format. What is also affected are the tasks creating html, odt or docs: Although these are based on htlatex which is always dvi-based, the preprocessing is done in dvi or in pdf. Also the task txt is affected.	
`dvi2pdfCommand`	`dvipdfmx`
The driver to convert dvi into pdf-files. Note that this must fit the options of the packages '`xcolor`' and '`graphicx`'. Sensible values are '`dvipdf`', '`dvipdfm`', '`dvipdfmx`', and '`dvipdft`' (which is '`dvipdfm`' with option '`-t`').	
`dvi2pdfOptions`	the empty string
The options for the command `$dvi2pdfCommand`.	
`patternReRunLatex`	see Section 6.3.3
The pattern is applied line-wise to the log file and matching triggers rerunning `$latex2pdfCommand` if `$maxNumReRunsLatex` is not yet reached to ensure termination. This pattern may never be ensured to be complete, because any package may indicate the need to rerun `$latex2pdfCommand` with its own pattern and so any new package may break completeness. Nevertheless, the default value aims completeness while be tight enough not to trigger a superfluous rerun. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.	
`maxNumRerunsLatex`	`5`
The maximal allowed number of reruns of the LaTeX process. This is to avoid endless repetitions. This shall be non-negative or -1 which signifies that there is no threshold.	

Table 6.3: The LaTeX-to-pdf-converter

6.3.1 The parameter `latex2pdfOptions`

TODO: add

6.3.2 The parameter patternWarnLatex

The patterns given below are just by (unwritten) convention. As a consequence, the pattern has a comprehensive default value covering all warnings known to the author, while not detecting a warning, where there is none. To that end, the pattern requires that the warning text starts with the line of the log file. Still the pattern has to be configurable to allow the user to overwrite the default value not being forced to wait for the developer to change it.

For the current default value, we distinguish

- LaTeX-warnings emitted directly by LaTeXstarting with `LaTeX Warning:` ,
- LaTeX-font-warnings related with fonts/font selection starting with `LaTeX Font Warning:` ,
- Package warnings emitted by a package. By convention, a package emitting a warning identifies itself by its name `<name>` emitting a warning starting with `Package <name> Warning:` ,
- Class warnings emitted by a package. By convention, a class emitting a warning identifies itself by its name `<name>` emitting a warning starting with `Class <name> Warning:` ,
- pdftex-warning
- Fontspec warnings. Please note the leading character "`*`".
- Further warnings not identifying themselves as warnings as the word "warning" does not occur.

The resulting default pattern is

```
^(LaTeX Warning: |
LaTeX Font Warning: |
(Package|Class) .+ Warning: |
pdfTeX warning( \(((\d|\w)+\))?: |
\* fontspec warning: |
Missing character: There is no .* in font .*!$|
A space is missing\. (No warning)\.)
```

6.3.3 The parameter patternReRunLatex

TODO: add

The pattern to

As a consequence, the pattern has a comprehensive default value covering all warnings known to the author, while not detecting a warning, where there is none. To that end, the pattern requires that the warning text starts with the line of the log file. Still the pattern has to be configurable to allow the user to overwrite the default value not being forced to wait for the developer to change it.

For the current default value, we distinguish

The resulting default pattern is

```
^(LaTeX Warning: Label\(s\) may have changed\. Rerun to get cross-references right\.$|
Package \w+ Warning: .*Rerun( .*|\.)$|
Package \w+ Warning: .*$^\(\w+\) .*Rerun .*$|
LaTeX Warning: Etaremune labels have changed\.$|
\(rerunfilecheck\)               Rerun to get outlines right$)
```

FIXME: There is a bug in this pattern. See Section 10.

6.4 Parameters for Creation of the Bibliography

This section describes the parameters or creation of the bibliography which are given in Table 6.4.

TODO: do this.

Parameter Explanation	Default
`bibtexCommand`	`bibtex`
The BibTeX command to create a bbl-file from an aux-file and a bib-file (using a bst-style file).	
`bibtexOptions`	empty
The options for the command `$bibtexCommand`.	
`patternErrBibtex`	`error message`
The pattern is applied line-wise to the blg-file and matching indicates that `$bibtexCommand` failed. The default value is chosen according to the '`bibtex`' documentation.	
`patternWarnBibtex`	`^Warning--`
The pattern is applied line-wise to the blg-file and matching indicates a warning `$bibtexCommand` emitted. The default value is chosen according to the '`bibtex`' documentation.	

Table 6.4: The BibTeX-utility

6.5 Parameters for Creation of the Indices

This section describes the parameters or creation of the indices which are given in Table 6.5.

TODO: do this.

Parameter Explanation	Default
`makeIndexCommand` The MakeIndex command to create an ind-file from an idx-file logging on an ilg-file.	`makeindex`
`makeIndexOptions` The options for the MakeIndex command.	the empty string
`patternErrMakeIndex` The pattern is applied line-wise to the ilg-file and matching indicates that `$makeIndexCommand` failed. The default value is chosen according to the '`makeindex`' documentation.	`(!! Input index error )`
`patternWarnMakeIndex` The pattern is applied line-wise to the ilg-file and matching indicates a warning `$makeIndexCommand` emitted. The default value is chosen according to the '`makeindex`' documentation.	`(## Warning )`
`patternReRunMakeIndex` The pattern is applied line-wise to the log-file and matching triggers rerunning `$makeIndexCommand` followed by `$latex2pdfCommand`. This pattern only matches a warning emitted by the package '`rerunfilecheck`' e.g. used with option '`index`'. The default value is chosen according to the package documentation. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the bug.	see Section 6.5.1
`splitIndexCommand` The SplitIndex command to create ind-files from an idx-file logging on ilg-files. This command invokes `$makeIndexCommand`.	`splitindex`
`splitIndexOptions` The options for `$splitIndexCommand`. Here, one has to distinguish between the options processed by `$splitIndexCommand` and those passed to `$makeIndexCommand`. The second category cannot be specified here, it is already given by `$makeIndexOptions`. In the first category is the option '`-m`' to specify the `$makeIndexCommand`. This is used automatically and cannot be specified here. Since `$splitIndexCommand` is used in conjunction with package '`splitidx`', which hardcodes various parameters which are the default values for `$splitIndexCommand` and because the option may not alter certain interfaces, the only option which may be given explicitly is '`-V`', the short cut for '`-verbose`'. Do not use '`-verbose`' either for sake of portability.	`-V`

Table 6.5: The utilities MakeIndex and SplitIndex

6.5.1 The parameter `patternReRunMakeIndex`

As shown in [Obe16c], Section 2.3, Listing line 166, the pattern is

```
(^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get index right\.$)
```

6.6 Parameters for Creation of the Glossary

This section describes the parameters or creation of the glossary which are given in Table 6.6.

TODO: do this.

Parameter Explanation	Default
`makeGlossariesCommand` The MakeGlossaries command to create a gls-file from a glo-file (invoked without file ending) also taking ist-file or xdy-file into account logging on a glg-file.	`makeglossaries`
`makeGlossariesOptions` The options for the `$makeGlossariesCommand`. These are the options for '`makeindex`' (not for `$makeIndexCommand`) and for '`xindy`' (also hardcoded). The aux-file decides on whether program is executed and consequently which options are used. The default value is the empty option string. Nevertheless, '`xindy`' is invoked as '`xindy -L english -I xindy -M...`'. With option '`-L german`', this is added. Options '`-M<`' for '`xindy`' '`-s`' for '`makeindex`' and '`-t`' and '`-o`' for both, '`xindy`' and '`makeindex`'.	the empty string
`patternErrMakeGlossaries` The pattern is applied line-wise to the 'glg'-file and matching indicates that `$makeGlossariesCommand` failed. The default value '`(^ unable to execute: )`' is chosen according to the 'makeindex' documentation. If the default value is not appropriate, please modify and notify the developer of this plugin.	`(^\*\*\* unable to execute: )`
`patternErrXindy` The pattern in the 'glg'-file which indicates an error when running '`xindy`' via `$makeGlossariesCommand`. If the default value is not appropriate, please modify and notify the developer of this plugin.	`(^ERROR: )`
`patternWarnXindy`	`(^WARNING: )`

	The pattern is applied line-wise to the 'glg'-file and matching indicates a warning when running 'xindy' via $makeGlossariesCommand. The default value '(^WARNING:)' (note the space and the brackets) is chosen according to the 'xindy' documentation. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the deficiency.
patternReRunMakeGlossaries	see Section 6.6.1 The pattern is applied line-wise to the log file and matching triggers rerunning $makeGlossariesCommand followed by $latex2pdfCommand. This pattern only matches a warning emitted by the package 'rerunfilecheck' e.g. used with option 'glossary'. The default value is chosen according to the package documentation. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the bug.

Table 6.6: The MakeGlossaries-utility

6.6.1 The parameter patternReRunMakeGlossaries

As shown in [Obe16c], Section 2.3, Listing line 210, the pattern is

```
(^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get index right\.$)
```

This holds even if splitindex is used.

6.7 Parameters for Conversion LaTeX to html

This section describes the parameters of the LaTeX-to-html converter which are given in Table 6.7.

Parameter Explanation	Default
tex4htCommand	htlatex
tex4htStyOptions	xhtml,uni-html4,2,svg,pic-tabular
tex4htOptions	' -cunihtf -utf8'
t4htOptions	the empty string

The options for '`t4ht`' which converts idv-file and lg-file into css-files, tmp-file and, by need and if configured accordingly into png-files. The value '`-p`' prevents creation of png-pictures.
`patternT4htOutputFiles` see Section 6.7.1 The pattern is applied to file names and matching shall accept exactly the target files of goal '`html`' for a given latex main file '`xxx.tex`'. Matching triggers copying those files to `$outputDirectory`. The patterns for the other targets are hardcoded and take the form '`^T$T\.yyy$`', where '`yyy`' may be an ending or an alternative of endings. This pattern is neither applied to directories nor to '`xxx.tex`' itself. For an explanation of the pattern '`T$T`' see `$patternCreatedFromLatexMain`. Spaces and newlines are removed from that pattern before processing. The pattern is designed to match quite exactly the files to be copied to `$targetSiteDirectory`, for the goal '`html`', not much more and at any case not less. since `$tex2htCommand` is not well documented, and still subject to development, this pattern cannot be guaranteed to be final. If the current default value is not appropriate, please overwrite it in the configuration and notify the developer of this plugin of the bug.

Table 6.7: The LaTeX-to-html-converter

6.7.1 The parameter `patternT4htOutputFiles`

The default value has the following components:

- '`^T$T\.x?html?$`' is the main output file.
- '`^T$Tli\d+\.x?html?$`' are lists: toc, lof, lot, indices, glossaries, NOT the bibliography.
- '`^T$T(ch|se|su|ap)\d+\.x?html?$`' are chapters, sections and subsections or below and appendices.
- '`^T$T\d+\.x?html?$`' are footnotes.
- '`^T$T\.css$`' are cascaded stylesheets.
- '`^T$T-\d+\.svg$`' and '`^T$T\d+x\.png$`' are svg/png-files representing figures.
- '`^T$T\d+x\.x?bb`' are the bounding boxes (suffix `.bb` for `dvipdfm` and suffix `.xbb` for `dvipdfmx`).
- '`^(cmsy)\d+(-c)?-\d+c?\.png$`' represents special symbols.

Note that the patterns for the html-files can be summarized as

```
^T$T((ch|se|su|ap|li)?\d+)?\.x?html?\$
```

This altogether constitutes the default value for this pattern:

```
^(T$T(((ch|se|su|ap|li)?\d+)?\.x?html?|
\.css|
\d+x\.x?bb|
\d+x\.png|
-\d+\.svg)|
(cmsy)\d+(-c)?-\d+c?\.png)$
```

The pattern is designed to match quite exactly the files to be copied to `$targetSiteDirectory`, for the goal "html", not much more and at any case not less. since `$tex2htCommand` is not well documented, and still subject to development, this pattern cannot be guaranteed to be final.

6.8 Parameters for further Conversions

This section describes the parameters of the converter from and to further formats which are given in Table 6.8.

These converters convert latex into rtf directly, they convert odt into doc-like documents and pdf into pure text. A special case is the code-checker in a sense converting latex into a log-file. For each of them, the name of the command can be specified and also the options. Since neither of them, except the code checker, write a log-file, there are no further parameters necessary.

Parameter	Default
Explanation	
`latex2rtfCommand`	`latex2rtf`
The latex2rtf command to create rtf from latex directly.	
`latex2rtfOptions`	the empty string
The options of the command `$latex2rtfCommand`.	
`odt2docCommand`	`odt2doc`
The odt2doc command to create MS word-formats from otd-files.	
`odt2docOptions`	`-fdocx`

	The options of the command `$odt2docCommand`. Above all specification of output format via the option '-f'. Invocation is '`odt2doc -f<format> <file>.odt`'. All output formats are shown by '`odt2doc -show`' but the formats interesting in this context are the following: `doc`, `doc6`, `doc95`, `docbook`, `docx`, `docx7`, `ooxml` and `rtf`. Interesting also the verbosity options '`-v`', '`-vv`', '`-vvv`' the timeout '`-T=secs`' and '`-preserve`' to keep permissions and timestamp of the original document.
`pdf2txtCommand`	`pdftotext`
	The pdf2txt-command for converting pdf-files into plain text files.
`pdf2txtOptions`	the empty string
	The options of the command `$pdf2txtCommand`.

Table 6.8: The parameters of further converters

FIXME: Note that `pdftotext -h` prints a usage message. This is a way to obtain not the specified output. It shows that `pdftotext -q` does not print any messages or errors. This indicates that `pdftotext` normally does display error messages on the standard output. These may be led to a log file to indicate errors and warnings. Here, further research is required.

The option `-htmlmeta` seems not approprate. The option resolution `-r` seems sensible only in conjunction with the crop area defined by `-x` and `-y` which does not make sense in our context. The same holds for specification of the first and the last page via `-f` and `-l`. What does make sense is specifying the encoding via `-enc` with possible values given by `pdftotext -listenc`. What makes sense most is `UTF-8`.

6.9 The code checker `chktex`

Among the applications used by this software, the codechecker plays a special role: it is not really a converter, unless we interprete the log file as artifact. Like for the most converters also for the codechecker we can specify the command ant its options.

Parameter	Default
Explanation	
`chkTexCommand`	`chktex`
	The chktex-command for checking latex main files.
`chkTexOptions`	`-q -b`

The options of the command `$chkTexCommand`, except "`-o output-file`" specifying the output file which is added automatically. For further details see the options below.

Table 6.9: The parameters of the code checker

The options of `chktex` are described in detail in [Thi16], Section 6.1.2.

Here is a list of options useful in this context. The first group of these are muting options:

- '-w', '-e', '-m', Make the message number passed as parameter a warning/an error/a message and turns it on. Messages are not counted.
- '-n' Turns the warning/error number passed as a parameter off.
- '-L' Turns off suppression of messages on a per line basis.

The next group of interesting options are for output control:

'-q' Shuts up about copyright information.

'-o output-file' Specifies the output file. This is added automatically and shall thus not be specified by the user.

'-b0/1' If you use the -o switch, and the named output-file exists, it will be renamed to 'filename.bak' for option `-b1` and not for `-b0`.

'-f format' Specifies the format of the output via a format similar to "`printf()`'. For details consult the manual [Thi16], Section 6.1.2. The codes are listed below.

'-vd' Verbosity level followed by a number 'd' specifying the format of the output according to the listing below. The verbosity number is resolved as a pattern as if given by the option '-f format'. Thus the option '-v' is ignored if the option '-f format' is specified.

The default value `-q -b0` avoids verbose output and backing up the output log-file.

Code

%b String to print between fields (from -s option).

%c Column position of error.

%d Length of error (digit).

%f Current file-name.

%i Turn on inverse printing mode.

%I Turn off inverse printing mode.

%k kind of error (warning, error, message).

%l line number of error.

%m Warning message.

%n Warning number.

%u An underlining line (like the one which appears when using '-v1').

%r Part of line in front of error ('S'-1).

%s Part of line which contains error (string).

%t Part of line after error ('S'+1).

FIXME: to be inserted. From `chktexrc`:

```
OutFormat
{
# -v0; silent mode
%f%b%l%b%c%b%n%b%m!n

# -v1; normal mode
"%k %n in %f line %l: %m!n%r%s%t!n%u!n"

# -v2; fancy mode
"%k %n in %f line %l: %m!n%r%i%s%I%t!n!n"

# -v3; lacheck mode
"!"%f!", line %l: %m!n"

# -v4; verbose lacheck mode
"!"%f!", line %l: %m!n%r%s%t!n%u!n"

# -v5; no line number, ease auto-test
"%k %n in %f: %m!n%r%s%t!n%u!n"

# -v6; emacs compilation mode
"!"%f!", line %l.%c:(#%n) %m!n"
}
```

Note that "`!`" is to escape quotes and newline. More than these can be added to `chktexrc`.

This document is checked with options deviating from the default value:

```
-q -b0 -v1 -g0 -l ${basedir}/src/site/tex/chktexrc
```

The default is `-q -b0`, option `-g0` means that the global `chktexrc` is not used and option

```
-l ${basedir}/src/site/tex/chktexrc
```

specifies a record file tailored to the needs of this project. In particular, the pattern for `-v1` is slightly modified: It is

```
# -v1; normal mode
"%k %n in %f line %l: %m!n %r%s%t!n %u!n"
```

which adds a blank to all lines but the headlines. That way, the kind of issue (%k) is easily parsed. This could be used for emitting an error instead of a warning when processing goal *check*.

6.10 Parameters for Ensuring Reproducibility

For a general description of the reproducibility check see Section 5.7. Here we go into the details and identify the parameters controlling the check and specified in great detail in Table 6.10. As already mentioned in Section 5.7, currently, checks are performed for artifacts in pdf format only; more formally, if the target (which is in parameter `target` described in Table 6.1) is `pdf`.

But if so, the parameter `chkDiff` decides whether a check is performed at all. Note that checking is off by default. Then a diffing tool given by `diffPdfCommand` expects the blueprints in the directory `diffDirectory`. In contrast, the actual artifacts to be checked are in `outputDirectory`, whereas the sources are in `texSrcDirectory`.

The location of a source tex file relative to `texSrcDirectory` is the location of the artifact relative to `outputDirectory`. This path relative to `diffDirectory` is the location of the blueprint. With the actual artifact in `outputDirectory` and the blueprint in `diffDirectory` the diff-tool determines whether the both are equivalent. If so, equivalence is logged as an info, else an exception described in Table reftab:TLP is thrown.

Note that the choiced of the diff tool `diffPdfCommand` determines teh notion of equivalence of the pdf artifacts, ranging from byte equivalence to some kind of visual equivalence.

Parameter	Default
Explanation	
`diffDirectory`	`.`
Diff directory relative to `$baseDirectory` used for diffing actually created artifact against prescribed one inthis directory. This is relevant only if `$chkDiff` is set. The default value is `.`.	
`chkDiff`	`false`
Indicates whether after creating artifacts and copying them to the output directory `$outputDirectory` the artifacts are checked by diffing them against preexisting artifacts in `$diffDirectory` using the diff command given by `$diffPdfCommand`. Note that currently, only pdf files are checkd. This is `false` by default and is set to `true` only in the context of tests.	
`diffPdfCommand`	`diff`
The diff-command for diffing pdf-files strictly or just visually to check that the created pdf files are equivalent with prescribed ones. CAUTION: there are two philsophies: Either the latex source files are created in a way that they reproduce strictly. Then a strict diff command like `diff` is appropriate. Else another diff command is required which checks for a kind of visual equality. The default value is a mere `diff`. Alternatives are `diff-pdf` and `diff-pdf-visually` both implementing a visual diff. Note that unlike for other tools, no options can be passed in this case explicitly.	

Table 6.10: The parameters of the pdf differ

The options of `chktex` are described in detail in [Thi16], Section 6.1.2.

Chapter 7

Exceptions and Logging

If during execution of this software something goes wrong and it is possible to detect that, the user shall be notified.

Maven foresees a mechanism to abort the whole build, i.e. lifecycle phase or a single goal and accordingly ant allows to abort a task. In both cases, abortion is implemented by throwing an exception.

A maven plugin aborts a goal throwing a

`org.apache.maven.plugin.MojoFailureException`

and a

`org.apache.maven.plugin.MojoExecutionException`

to abort the life-cycle phase. Since this plugin is just for documentation, there is no need to abort site creation altogether, so only the former exception occurs.

An ant-task aborts an ant-build throwing a

`org.apache.tools.ant.BuildException`

without further distinction.

This software provides both a maven plugin and an ant task built on the same code base. Thus the maven plugin throws a `MojoFailureException` if and only if the according ant-task throws an `BuildException` in the same situation.

Section 7.1 describes the philosophy of throwing an exception and defines in detail under what circumstances which exception is thrown.

Roughly speaking, an exception is thrown only if something is really wrong, e.g. a non-recoverable error or an indication that the build system is out of control or if this plugin/task is likely to destroy the work of another plugin/task.

If something went wrong, but no exception is thrown, the user must be notified by logging and the build process to go on, skipping a section of a task as small as

possible. Both, maven and ant provide a logging mechanism with the levels error, warning, info and debug. Section 7.2 describes the errors and warnings; the lot of infos and debugging output are not described here.

Verbosity is chosen by the following command line options:

`-e` shows error messages,

`-X` shows debug-messages,

`-q` quiet hides the info-level and shows *only* errors.

There seems no way to get warnings only.

Each exception offers a message and also each warning has a warning message. The messages are endowed with a unique identifier of the form `KCCDD`, where `K` is the kind which is one of

T Throwable, which results in a `MojoFailureException` for the maven-plugin and `BuildException` for the ant-task. This is described in detail in Section 7.1

E logging as ERROR,

W logging as WARNING

I logging as INFO which occurs frequently

D logging as DEBUGGING output, which is lengthy

The shortcut `CC` describes the class where the exception is thrown or the warning is logged:

EX CommandExectutorImpl: a class executing applications on a command line.

PP LatexPreProcessor: preprocessing of LaTeX-files: Processing of graphic files and detection of latex main files.

LP LatexProcessor: processing of LaTeX-main files: conversion into various output formats.

SS Settings: A container holding the values of all parameters. These are either default or read from the configuration in the pom for the maven plugin and in the build file for the ant task.

MI MetaInfo: offering meta information as expected and actual versions of converters.

FU TexFileUtilsImpl: a class providing access to files.

Finally, DD is a two digit number enumerating the messages.

Identifier	Message Explanation
WMI01	Version string from converter $conv did not match expected form: $conv: 'version'not?in$interv Indicates that the version string coming from the converter $conv is not as expected. Programming error excluded, this means that the version does not fit, i.e. is not in $interv.
WMI02	$conv: '$actVersion'not in$interv Indicates that the version of converter $conv can be detected and is $actVersion but does not fit the expectation which is $expVersion.

Table 7.1: The logging for MetaInfo

Identifier	Message Explanation
WFU01	Cannot read directory '$dir'; build may be incomplete. TBD.
XFU02	TBD TBD
WFU03	Cannot close '$file'. TBD
EFU05	Cannot delete file '$file'. TBD
EFU06	Cannot move file '$src' to '$dest'. TBD
EFU07	File '$srcFile' to be filtered cannot be read. WORKAROUND for inkscape filtering eps_tex-file into ptx file: The former is not a readable regular file.
EFU08	Destination file '$destFile' for filtering cannot be written. WORKAROUND for inkscape filtering eps_tex-file into ptx file: The latter is not a writable regular file.
EFU09	Cannot filter file '$srcFile' into '$destFile'. WORKAROUND for inkscape filtering eps_tex-file into ptx file: Either reading a line or writing a line failed.

Table 7.2: The logging for TexFileUtils

TBD: check whether workaround still necessary. TBD: complete list TBD: add missing lists

7.1 Exceptions

Exceptions are thrown only if no substantial part of this maven-goal or this ant-task may be completed as if the tex source directory does not exist or is no directory or if a failure occurs which indicates that the underlying system does not work properly, as if the tex source directory or a sub-directory is not readable or if execution of an external program fails. The latter does not mean that the program returns with an error code, but it means that execution from within java fails.

Identifier	Message Explanation
TEX01	Error running $command. Compare with EEX01 in Table 7.9: Error execution means • the file expected to be the working directory does not exist or is not a directory. • method Runtime.exec(String, String[], File) fails throwing an IOException. • an error inside systemOut parser occurs • an error inside systemErr parser occurs • Wrapping an InterruptedException on the process to be executed thrown by Proces.waitFor(). whereas for EEX01 just a failure code is returned.

Table 7.3: The BuildFailureExceptions of the internal class CommandExecutorImpl

Identifier	Message Explanation
TSS01	The tex source directory '$texSrcDirectoryFile' should be an existing directory, but is not. The tex source directory is given in the pom/build-file with default value ./src/site/tex. It contains or is $texSrcProcDirectoryFile. Thus is must be a directory.
TSS02	The tex source processing directory '$texSrcProcDirectoryFile' should be an existing directory, but is not.

The tex source processing directory is given in the pom/build-file relative to `$texSrcDirectoryFile` with default value `..` It contains all files to be processed. Thus is must be a directory.

`TSS03` `The output directory '$outputDirectory'`
`should be a directory if it exists, but is not.`

The output directory is given in the pom/build-file with default value `./target/site/.`. The output directory is where the result of the goal/-task are copied to. If it does not yet exist, it is created but if it exists and is a regular file, it cannot be created any more.

`TSS04` `The target set '$targets'`
`should be a subset of the registered targets '...', but is not.`

The target set is given in the pom/build-file with default value `pdf, html`. A single target can be given on the command line as e.g. via `mvn latex:pdf` and also in this case, the validity of the target is checked, so that e.g. `mvn latex:invalid` throws an exception, but the mechanism relies directly on mavens ability to check the targets of this plugin.

`TSS05` `The excluded converters '$convertersExcluded'`
`should form a subset of the registered converters '...'.`

From the possible "registered" converters the ones not used may be excluded to avoid that they cause errors when trying to check correctness of version in target `vrs` accessed via `mvn latex:vrs`. These converters may not even be installed.

`TSS06` `Tried to use converter '$convStr'`
`although not among the registered converters '...' as expected.`

Only registered converters may be used.

`TSS07` `Tried to use converter '$convStr'`
`although among the excluded converters '...'.`

Among the registered converters only those may be used, which are not excluded, i.e. listed in configuration in section `convertersExcluded`.

`TSS08` `Tried to use converter '$convStr'`
`in configuration '...' instead of configuration '...'.`

Each converter may occur in a specified configuration only. So e.g. `pdflatex` is only allowed in configuration '`latex2pdfCommand`'. If used in configuration '`makeIndexCommand`' this causes this exception, because in that configuration, e.g `makeindex` is allowed.

`TSS09` `The diff directory '$diffDirectoryFile'`
`should be a directory if it exists, but is not.`

The `$diffDirectoryFile` shall exist and be a directory. In it shall be stored the artifacts the actually created shall be compared with if `chkDiff` is set using the command `diffPdfCommand`. As the name suggests, currently only pdf files are compared.

Table 7.4: The `BuildFailureException`s of Settings

Id. Message Explanation
`TMI01` `Cannot get stream to file '$fileName'.` Stream to file within jar. This may be the manifest file, pom.properties or git.properties.
`TMI02` `Cannot load properties from file '$fileName'.` Provided the stream to the file is ok, could not load property. This may occur for pom.properties or git.properties.
`TMI03` `IOException reading manifest.` Provided the stream to the manifest file is ok, could not read completely.

Table 7.5: The `BuildFailureException`s of MetaInfo

Id. Message Explanation
`TFU01` `Cannot create destination directory '$targetDir'.` This is mainly because of writing permissions.
`TFU04` `Cannot overwrite directory '$destFile'.` Because this plugin shall not turn directories into regular files and vice versa. This failure indicates that another plugin/task disturbs this one.
`TFU06` `Cannot copy file '$srcFileName' to directory '$targetDir'.` This is mainly because of writing permissions.

Table 7.6: The BuildFailureExceptions of TexFileUtilsImpl

Id. Message Explanation
`TLP01` `Artifact '$pdfFileAct' of '$texFile' could not be reproduced.` Processing $texFile' yields $pdfFileAct which is not "alike" the stored version. Currently, that kind of check can be performed for `pdf` files only. Also the diff check is executed only if parameter $chkDiff is set. Then the diff command $diffPdfCommand is performed to determine whether the artifacts are equivalent in the sense given by the diff command. The concrete meaning of that equivalence may range from strict equivalence to some kind of visual equivalence.

Table 7.7: The BuildFailureExceptions of LatexProcessor

FIXME: to be added.

7.2 Logging of Warnings and Errors

The rules for logging warnings and errors is, that the user must be notified, if something went wrong but the run is not aborted, by a warning or an error. It is not required that for each detail going wrong, there is a separate notification, but the user must be sure, that all is ok, if no warning and no error occurs.

To decide whether it is an error or a warning to be logged, one has to distinguish, whether the problem occurs when running an external application or within internal code. In the first case, the decision whether it is an error or a warning is left to that application:

- If the application returns an error code other than 0, it is an error.
- If the application is expected to write a log file but none is found, it is an error. The applications used here, return a nontrivial error code if no log file is written.
- The applications used here, writing a log file distinguish between error and warning. If a log file is written both are logged in the log file and can be distinguished by the form of the entry via pattern matching. If no error occurs, the return code is 0, even if warnings occur.
- If an application writes at least one error into the log file, this software logs an error.
- If an application writes no error into the log file but at least one warning, principally this software logs a warning. There may be parameters to switch off warnings partially or all of them, but there must be also a configuration of parameter values that allow logging all warnings.

If an application does not create the expected output file, this software logs an error. This may be because of an internal error as described above, but also because of wrong parameters. So, e.g. `pdflatex -v xxx.tex` does not create a pdf-file as expected.

Id. Message Explanation
`EEX01 Running $command failed with return code $returnCode.` Compare with TEX01 in Table 7.3: Error execution means that there is even no valid return code.
`EEX02 Running $command failed: No target file '$fileName' written.` FIXME
`EEX03 Running $command failed: Target file '$fileName' is not updated.`

The command $command is expected to write to the file '$fileName' but this file is not updated. This indicates an error executing $command.
WEX04 Cannot read target file '$fileName'; may be outdated.
FIXME
WEX05 Update control may emit false warnings.
FIXME
EAP02 Running $command failed: No log file '$logFileName' written.
The command $command is expected to write a log file '$logFileName' but no such file exists. This indicates an error executing $command.
EAP01 Running $command failed. Errors logged in '$logFileName'.
The command $command logged at least one error in the file '$logFileName', where more details can be found.
WAP03 Running $command emitted warnings logged in '$logFileName'.
The command $command logged at least one warning in the file '$logFileName', where more details can be found. Note that if $command is a latex processor, this warning comes only iff the parameter $debugWarnings is set. Note also that notifications on bad boxes are not counted as warnings here.
WLP03 Running $command created bad boxes logged in '$logFileName'.
Here, $command is a latex processor. It logged at least one bad box, overfull or underfull, horizontal or vertical in $logFileName where more details can be found. Note that this warning comes only iff the parameter $debugBadBoxes is set.
WLP06 Running $command found issues logged in '$logFileName'.
Here, $command is a checker tool. Strictly speaking, unlike the other warnings here, this does not signify that running $command went wrong but uncovered an issue (warning/error/message) logged in a file.
WLP05 Use package 'splitidx' without option 'split' in '$texFileName'.
This indicates that an extended idx-file "xxx-yy.idx" has been found without xxx.idx or without according entry \indexentry[yy]{...}{...} in xxx.idx.

Table 7.8: The errors and warnings on running a command

Id. Message Explanation
WFU01 Cannot read directory '$dir'; build may be incomplete.
FIXME
WPP02 Cannot read tex file '$texFile'; may bear latex main file.
FIXME
WAP04 Cannot read log file '$logFileName'; may hide warnings/errors.
FIXME
WLP02 Cannot read log file '$logFileName'; $kind may require rerun.
FIXME

`WLP04 Cannot read idx file '$idxFileName'; skip creation of index.`
FIXME
`WFU03 Cannot close '$closeable'.`
FIXME
`EFU05 Cannot delete file '$delFile'.`
 	
`EFU06 Cannot move file '$fromFile' to '$toFile'.`
FIXME

Table 7.9: The errors and warnings on files/streams

Id. Message Explanation

`WPP03 Skipped processing of files with suffixes $skipped.`
FIXME
`WPP04 Skip processing '$srcFile': interpreted as target of '$lmFile'.`
FIXME
`WLP01 LaTeX requires rerun but maximum number $maxNumRerunsLatex reached.`
FIXME

Table 7.10: Miscellaneous errors and warnings

FIXME: to be added.

Chapter 8

Listings

First we list the versions of the executables which are run to create this manual. They are given in `../../main/resources/version.properties`

```
# contains version properties:
# Each line except empty or comments
# map the command of a tool used by this software
# to an interval of versions for which it is guaranteed it works.
# tied to the above version of the converters potentially used
#core=${version} TBD: this is included only to demonstrate filtering
#not before pdfTeX 3.14159265-2.6-1.40.17 to enable reproducibility features like
    \pdfomitdate
pdflatex=[1.40.21;1.40.24]
lualatex=[1.12.0;1.15.0]
xelatex=[0.999992;0.999994]
#htlatex is unknown
latex2rtf=[2.3.16 r1254;2.3.18 r1267]
# the following are unused
#latex2png=2.3.16 r1254
#latex2html=2.3.16 r1254
#latex2man=2.3.16 r1254
#latex2nemeth is unknown
odt2doc=[0.9.0]
# maybe replaces odt2doc in the long run, or is not needed at all
#unoconv=0.9.0
# maybe pandoc will play a crucial role in future
#pandoc=2.10.1
# pdftotex is within poppler and there is really a leap from 0.90.0 to 20.10.0
pdftotext=[21.04.0;22.04.0]
# further poppler tools currently not used
#pdftothtml=0.90.0
#pdftops=0.90.0
#pdftocairo=0.90.0
dvips=[2020.1;2022.1]
dvipdfm=[20210318;20211117]
dvipdfmx=[20200315;20211117]
xdvipdfmx=[20200315;20211117]
dvipdft=[20090604.0046]
#dvipdf may be reconstructed from that of gs and dvips
#df2ps=2020.1#                TBC: where needed?
gs=[9.52.0;9.54.0]

# checker tools
```

```
chktex=[1.7.6]
diff=[3.8]
# TBD: replace by true version. Cannot be determined by command line option
diff-pdf=[300]
# TBD: replace by true version. Cannot be determined by command line option
diff-pdf-visually=[0]
pdfinfo=[22.01.0;22.04.0]
exiftool=[12.39;12.41]

bibtex=[0.99d]
# the next two changed version string without change of version
bibtexu=[3.71;3.72]
bibtex8=[3.71;3.72]

makeindex=[2.15;2.16]
upmendex=[0.54;1.00]
splitindex=[0.1]
xindy=[2.5.1]

makeglossaries=[4.45;4.49]

pythontex=[0.17;0.18]
latexmk=[4.70b;4.77]

# texlive-pgf
# see /usr/share/texmf/tex/generic/pgf/pgf.revision.tex"
#texlive-pgf=3.1.5b
mpost=[2.00;2.02]
ebb=[20200315;20211117]
gnuplot=[5.4 patchlevel 0;5.4 patchlevel 3]
inkscape=[1.0.2;1.1.2]
fig2dev=[3.2.7b;3.2.8b]

# TBD: take also latex packages into account,
# but this must be done in another way.
```

Listing 8.1: The versions of the executables potentially used

Next we provide listings of the configuration in the `pom.xml` for the maven-plugin and in the `build.xml` for the ant-task. The listing of the `build.xml` shows which parameters are attributes and which ones are text elements. In `pom.xml` only text elements occur.

The maven-plugin described here, requires the definition in `pom.xml` given by Listing 8.2 on page 102.

```
<plugin>
  <groupId>eu.simuline.m2latex</groupId>
  <artifactId>latex-maven-plugin</artifactId>
  <version>${project.version}</version>
  <!--TBD: This must be updated with the resources plugin -->
  <configuration>
    <!--baseDirectory>${basedir}</baseDirectory-->
    <!--targetDirectory>${project.build.directory}</targetDirectory-->
    <targetSiteDirectory>${basedir}/src/test/resources/integration/target</
        targetSiteDirectory>
    <!-- Indicates whether the 'VersionMojo' displays warnings only
              or also creates infos.
              Infos refer to the version of this plugin
              and also on its git commit,
```

```
                but also on the versions of the converters found
                and lists the converters excluded, i.e. those not used
                and thus not tested on version.

                Warnings are emitted e.g.
                if the version of a converter does not fit the expectations,
                the version of a converter could not be retrieved,
                e.g. because it is not installed
                or if the converter specified is unknown altogether.
                This defaults to 'false' displaying also info.

                The latter is appropriate for using in command line
                'mvn latex:vrs', whereas in builds by default
                the pom overwrites this to have output only
                in case something goes wrong. -->
        <!--versionsWarnOnly>false</versionsWarnOnly-->
        <settings>
          <!--convertersExcluded>diff</convertersExcluded-->
          <!-- The latex source directory as a string
relative to $baseDirectory,
containing $texSrcProcDirectory.
This directory determines also the subdirectory of
$outputDirectory to lay down the generated artifacts.
The default value is 'src/site/tex' on Unix systems. -->
          <texSrcDirectory>src/site/tex</texSrcDirectory>

          <!-- The latex source processing directory as a string
relative to $texSrcDirectory
containing all tex main documents
and the graphic files to be processed
and also to be cleaned.
Whether this is done recursively in subfolders
is specified by $readTexSrcProcDirRec.
The default value is '.'. -->
          <texSrcProcDirectory>.</texSrcProcDirectory>

          <!-- Whether the tex source directory $texSrcProcDirectory
shall be read recursively for creation of graphic files,
i.e. including the subdirectories.
This is set to 'false' only during information development.
The default value is 'true'. -->
          <readTexSrcProcDirRec>true</readTexSrcProcDirRec>

          <!-- The artifacts generated by $latex2pdfCommand
will be copied to this folder
relative to $targetSiteDirectory
which is by default '$targetDirectory/site' on Unix systems.
The default value is '.'. -->
          <outputDirectory>.</outputDirectory>

          <!-- Diff directory relative to 'baseDirectory'
     used for diffing actually created artifact against prescribed one inthis
         directory.
     This is relevant only if 'chkDiff' is set.
     The according file is given by 'diffDirectoryFile'.
     The default value is <code>.</code>. -->
          <diffDirectory>src/main/resources/docsCmp</diffDirectory>

          <!-- A comma separated list of targets
               to be stored in $targetSet.
               Allowed values are pdf, html, txt, odt and docx.
               The default value is 'pdf, html'. -->
```

```
            <targets>pdf</targets>              <!-- , html -->

            <!-- A comma separated list of excluded {@link Converter}s
given by their command, i.e. by {@link Converter#getCommand()}
returned as a set by {@link #getConvertersExcluded()}.
Excluded converters need not be installed.
They don't show up in the version check of target 'vrs'
and of course they are not allowed to be used.
By default, this list is empty. -->
            <convertersExcluded></convertersExcluded>
            <!--convertersExcluded>xindy, upmendex</convertersExcluded-->

            <!-- The pattern to be applied to the contents of tex-files

NOT QUITE: to the beginning
which identifies a latex main file.
The default value is choosen to match quite exactly
the latex main files.
Here we assume that the latex main file should contain
the declaration '\documentclass'
or the old fashioned 'documentstyle'
preceeded by a few constucts.
Strictly speaking, this is not necessary.
For a more thorough discussion,
and for an alternative approach, consult the manual.

The default value is choosen to match quite exactly
the latex main files, no more no less.
Since the pattern is chosen
according to documentation collected from the internet,
one can never be sure whether the pattern is perfect.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency. -->
            <patternLatexMainFile>
\A(\\RequirePackage\s*(\[(\s|\w|,)*\])?\s*\{\w+\}\s*(\[(\d|\.)+\])?|
\\PassOptionsToPackage\s*\{\w+\}\s*\{\w+\}|
%.*$|
\\input\{[^{}]*\}|
\s)*
\\(documentstyle|documentclass)
            </patternLatexMainFile>

            <!-- Path to the TeX scripts or null.
In the latter case, the scripts must be on the system path.
Note that in the pom, <texPath />
and even <texPath>    </texPath> represent the null-File.
The default value is null. -->
            <texPath/>




            <!-- Indicates whether after creating artifacts
                                                and copying them to the output
                                                    directory {@link #
                                                    outputDirectoryFile}
                                                the artifacts are checked by
                                                    diffing them against
```

```
                                                preexisting artifacts
                                            in {@link #diffDirectoryFile}
                                            using the diff command given by {
                                                @link #diffPdfCommand}.
                                            Note that currently, only pdf
                                                files are checkd.
                                            This is <code>false</code> by
                                                default and is set to <code>
                                                true</code> only
                                            in the context of tests. -->
            <chkDiff>true</chkDiff>


            <!-- Clean up the working directory in the end?
                 May be used for debugging when setting false.
                 The default value is 'true'. -->
            <cleanUp>true</cleanUp>

            <!-- This pattern is applied to file names
and matching shall accept all the files
which were created from a latex main file 'xxx.tex'.
It is neither applied to directories
nor to 'xxx.tex' itself.
It shall not comprise neither graphic files to be processed
nor files created from those graphic files.

This pattern is applied
in the course of processing graphic files
to decide which graphic files should be processed
(those rejected by this pattern)
and to log warnings if there is a risk,
that graphic files to be processed
are skipped or that processing a latex main file overwrites
the result of graphic preprocessing.

When clearing the tex source directory $texSrcProcDirectory,
i.e. all generated files should be removed,
first those created from latex main files.
As an approximation,
those are removed which match this pattern.

The sequence 'T$T'
is replaced by the prefix 'xxx'.
The sequence 'T$T' must always be replaced:
The symbol '$' occurs as end-sign as ')$'
or as literal symbol as '\$'.
Thus 'T$T' is no regular occurrence
and must always be replaced with 'xxx'.

Spaces and newlines are removed
from that pattern before matching.

This pattern may never be ensured to be complete,
because any package
may create files with names matching its own patterns
and so any new package may break completeness.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency.
The default value is given below. -->
            <patternCreatedFromLatexMain>
```

```
^(
T$T(                              \.([^.]*|synctex\.gz|out\.ps)|
(-|ch|se|su|ap|li)?\d+\.x?html?|
                    \d+x\.x?bb|
                   \d+x?\.png|
                    -\d+\.svg|
                      -.+\.(idx|ind|ilg))|
zzT$T\.e?ps|
(cmsy)\d+(-c)?-\d+c?\.png|
(pdf)?latex\d+\.fls)$
            </patternCreatedFromLatexMain>

            <!-- The fig2dev command for conversion of fig-files
into various formats.
Currently only pdf combined with pdf_t is supported.
The default value is 'fig2dev'. -->
            <fig2devCommand>fig2dev</fig2devCommand>

            <!-- The options for the command $fig2devCommand
common to both output languages.
For the options specific for the two output langugages
'pdftex' and 'pdftex_t',
see $fig2devPtxOptions and $fig2devPdfEpsOptions,
respectively.
The default value is the empty string.

o '-D +/-rangelist'
Export layers selectively ('+')
or exclude layers from export ('-').
E.g. -D +10,40,55:70,80  means  keep
only layers 10, 40, 55 through 70, and 80.
o '-j'
i18n (internationalization feature)
o '-m mag'
Set the magnification at which the figure is rendered
to 'mag'.
The default is '1.0'.
This is not usable within latex; not even '1.0'.
o '-s fsize'
Set the default font size (in points)
for text objects to 'fsize'.
Refers to the latex-fonts only.
o '-b width'
specify width of blank border around figure (1/72 inch). -->
            <fig2devGenOptions />

            <!-- The options for the command $fig2devCommand
specific for the output languages 'pdftex_t'
and 'pstex_t' which are the same.
Note that in addition to these options,
the option '-L pdftex_t' specifies the language,
$fig2devGenOptions specifies the options
common for the two output langugages
'pdftex' and 'pdftex_t'
and '-p xxx' specifies the full path
of the pdf/eps-file to be included without extension.

Possible options are the following:
(These seem to work for tex only
although according to documentation for all languages. )
Possible values are
```

```
o options specified for $fig2devGenOptions
o '-E num'
set encoding for text translation
(0 no translation, 1 ISO-8859-1, 2 ISO-8859-2)
o '-F'
do not set font family/series/shape,
          so you can set it from latex.
o '-v'
Verbose mode.

The default value for this option is the empty string. -->
            <fig2devPtxOptions />

            <!-- The options for the command $fig2devCommand
specific for the output language 'pdftex'.
Note that in addition to these option1s,
the option '-L pdftex' specifies the language and
$fig2devGenOptions specifies the options
common for the two output langugages
'pdftex' and 'pdftex_t'.
The default value for this option is the empty string. -->
            <fig2devPdfEpsOptions />

            <!-- The command for conversion of gnuplot-files
into various formats.
Currently only pdf (graphics)
combined with pdf_t (latex-texts) is supported.
The default value is 'gnuplot'. -->
            <gnuplotCommand>gnuplot</gnuplotCommand>

            <!-- The options specific for $gnuplotCommand s
output terminal 'cairolatex',
used for mixed latex/pdf-creation.
Note that the option 'pdf/eps'
of the terminal 'cairolatex' is not available,
because it is set internally.
The default option string is empty. -->
            <gnuplotOptions />

            <!-- The command for conversion of gnuplot-files
into metapost's postscript.
The default value is 'mpost'. -->
            <metapostCommand>mpost</metapostCommand>

            <!-- The options for the command $metapostCommand.
Leading and trailing blanks are ignored.
A sequence of at least one blank separate the proper options.
The default value is '-interaction=nonstopmode -recorder -s prologues=2'.
FIXME: does not work:'outputtemplate="%j_%c.mps"' -->
            <metapostOptions> -interaction=nonstopmode -recorder -s prologues=2
            </metapostOptions>

            <!-- The command for conversion of svg-files
into a mixed format.
The default value is 'inkscape'. -->
            <svg2devCommand>inkscape</svg2devCommand>

            <!-- The options for the command $svg2devCommand
for exporting svg-figures into latex compatible files.

The following options are mandatory:
o '-D' or '--export-area-drawing'
```

```
Export the drawing (not the page)
o '--export-latex'
Export PDF/PS/EPS without text.
Besides the PDF/PS/EPS, a LaTeX file is exported,
putting the text on top of the PDF/PS/EPS file.
Include the result in LaTeX like: \input{latexfile.tex}.
Note that the latter option is necessary,
to create the expected files.
It is also conceivable to export text as pdf/eps

The following options are prohibited,
because they are automatically added by the software:
'--export-filename=FILENAME'
The ending of the filename, which is either eps or pdf
determines the file type.

The default value is the minimal value,
'-D --export-latex'. -->
            <svg2devOptions>-D --export-latex</svg2devOptions>

            <!-- The command to create bounding box information
from jpg-files and from png-files.
This is run twice:
once with parameter '-m'
to create '.bb'-files for driver 'dvipdfm' and
once with parameter '-x'
to create '.xbb'-files for driver 'dvipdfmx'.
The default value is 'ebb'. -->
            <ebbCommand>ebb</ebbCommand>

            <!-- The options for the command $ebbCommand
except '-m' and '-x'
which are added automatically.
The default value is '-v'. -->
            <ebbOptions>-v</ebbOptions>

            <!-- The LaTeX command to create a pdf-file.
Possible values are e.g.
'pdflatex', 'lualatex' and 'xelatex'.
The default value (for which this software is also tested)
is 'pdflatex'. -->
            <latex2pdfCommand>pdflatex</latex2pdfCommand>

            <!-- occurs for xelatex but neither for pdflatex nor for lualatex:
LaTeX Font Warning: Font shape 'OMS/cmtt/m/n' undefined
(Font)              using 'OMS/cmsy/m/n' instead
(Font)              for symbol 'textbackslash' on input line 371. -->

            <!-- The options for the command $latex2pdfCommand.
Leading and trailing blanks are ignored.
A sequence of at least one blank
separates the proper options.

The default value  comprises the following options: -interaction=nonstopmode
prevents latex from stopping at the first error. -synctex=1
makes latex create a pdf file
which synchronizes with an editor supporting synchtex.
-src-specials
includes source specials into the output. dvi only?
-recorder
makes latex create an fls-file
specifying all inputted files.
```

```
-shell-escape enables \write18
but this can also be done via
-shell-restricted        enable restricted \write18 and
-enable-write18
which is needed for some reason for driver dvipdfmx
which seems to be the sole one supporting
pdf-pictures in dvi-mode and pdf-pictures in pdf-mode.
In pdftex this must be specified explicitly as
Driver dvipdfmx is always used by xetex.
required: -interaction=STRING     set interaction mode (STRING=batchmode/
    nonstopmode/
                scrollmode/errorstopmode)

-ipc                     send DVI output to a socket as well as the usual
                output file

[-no]-shell-escape       disable/enable \write18{SHELL COMMAND}
-shell-restricted        enable restricted \write18 -translate-file=TCXNAME use the
     TCX file TCXNAME
-8bit                    make all characters printable by default


// maybe, this shall be mandatory
-recorder                enable filename recorder


allowed:
-file-line-error-style
-record-package-usages -include-directory=dir
-enable-write18
-enc                     enable encTeX extensions such as \mubyte
-etex                    enable e-TeX extensions
-halt-on-error           stop processing at the first error
-ipc-start               as -ipc, and also start the server at the other end -mktex
    =FMT               disable/enable mktexFMT generation (FMT=tex/tfm/pk)
-mltex                   enable MLTeX extensions such as \charsubdef
[-no]-parse-first-line   disable/enable parsing of first line of input file
-src-specials            insert source specials into the DVI file -src-specials=
    WHERE     insert source specials in certain places of
                the DVI file. WHERE is a comma-separated value
                list: cr display hbox math par parend vbox -synctex=NUMBER
                           generate SyncTeX data for previewers if nonzero
-ini                     be pdfinitex, for dumping formats;
                this is implicitly true
                if the program name is 'pdfinitex'
not allowed:
-draftmode               switch on draft mode (generates no output PDF) -output-
    directory=dir   to specify the output dir -aux-directory=dir      to specify
    the output aux-dir -job-name=name          effectively changes the output file
     name
-quiet                   makes the log quiet and
                so circumvents error and warning detection
-no-file-line-error      disable/enable file:line:error style messages -fmt=FMTNAME
              use FMTNAME instead of program name or a %& line -output-format=
    FORMAT   use FORMAT for job output; FORMAT is 'dvi' or 'pdf'
                pdf is the only allowed... -progname=STRING        set program (
                    and fmt) name to STRING
                only names also without -progname are possible
-help                    display this help and exit
-version                 output version information and exit

-->
```

```
            <latex2pdfOptions> -interaction=nonstopmode -synctex=1
 -src-specials
 -recorder
 -shell-escape
            </latex2pdfOptions>

            <!-- The pattern is applied linewise to the log-file
and matching indicates an error
emitted by the command $latex2pdfCommand.

The default value is choosen to match quite exactly
the latex errors in the log file, no more no less.
Since no official documentation was found,
the default pattern may be incomplete.
In fact it presupposes, that $latex2pdfOptions
does not contain '-file-line-error-style'.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency.
The default value is '(^! )' (note the space). -->
            <patternErrLatex>(^! )</patternErrLatex>

            <!-- The pattern is applied linewise to the log-file
and matching indicates a warning
emitted by the command $latex2pdfCommand,
disragarding warnings on bad boxes
provided $debugWarnings is set.

This pattern may never be ensured to be complete,
because any package may indicate a warning
with its own pattern any new package may break completeness.
Nevertheless, the default value aims completeness
while be restrictive enough
not to indicate a warning where none was emitted.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency.
The default value is given below. -->
            <patternWarnLatex>
^(LaTeX Warning: |
LaTeX Font Warning: |
(Package|Class) .+ Warning: |
pdfTeX warning( \((\d|\w)+\))?: |
\* fontspec warning: |
Missing character: There is no .* in font .*!$|
A space is missing\. (No warning)\.)
            </patternWarnLatex>

            <!-- Whether debugging of
overfull/underfull hboxes/vboxes is on:
If so, a bad box occurs in the last LaTeX run,
a warning is displayed.
For details, set $cleanUp to false,
rerun LaTeX and have a look at the log-file.
The default value is 'true'. -->
            <debugBadBoxes>true</debugBadBoxes>

            <!-- Whether debugging of warnings is on:
If so, a warning in the last LaTeX run is displayed.
Note that warnings on boxes
```

```
is controlled separately via $debugBadBoxes.
For details, set $cleanUp to false,
rerun LaTeX and have a look at the log-file.
The default value is 'true'. -->
            <debugWarnings>true</debugWarnings>

            <!-- Whether creation of pdf-files from latex-files
goes via dvi-files.

If $pdfViaDvi is set
and the latex processor needs repetitions,
these are all done creating dvi
and then pdf is created in a final step
invoking the command $dvi2pdfCommand.
If $pdfViaDvi is not set,
latex is directly converted into pdf.

Currently, not only conversion of latex-files is affected,
but also conversion of graphic files
into graphic formats which allow inclusion in the tex-file.
If it goes via latex,
then the formats are more based on (encapsulated) postscript;
else on pdf.

In the dvi-file for jpg, png and svg
only some space is visible and only in the final step
performed by $dvi2pdfCommand,
the pictures are included using the bounding boxes
given by the .bb or the .xbb-file.
These are both created by $ebbCommand

Of course, the target dvi is not affected:
This uses always the dvi-format.
What is also affected are the tasks
creating html, odt or docs:
Although these are based on htlatex which is always dvi-based,
the preprocessing is done in dvi or in pdf.
Also the task txt is affected.

Performance:
false: 17.692s-18.892s
true:  22.449s-24.500s

The default value is 'false'. -->
            <pdfViaDvi>false</pdfViaDvi>

            <!-- The driver to convert dvi into pdf-files.
Note that this must fit the options
of the packages 'xcolor' and 'graphicx'.
Sensible values are
'dvipdf', 'dvipdfm', 'dvipdfmx',
and 'dvipdft'
(which is 'dvipdfm' with option '-t').
The default value is 'dvipdfmx'. -->
            <dvi2pdfCommand>dvipdfmx</dvi2pdfCommand>

            <!-- The options for the command $dvi2pdfCommand.
The default value is the empty string. -->
            <dvi2pdfOptions />

            <!-- The pattern is applied linewise to the log-file
and matching triggers rerunning $latex2pdfCommand
```

```
if $maxNumReRunsLatex is not yet reached
to ensure termination.

This pattern may never be ensured to be complete,
because any package
may indicate the need to rerun $latex2pdfCommand
with its own pattern any new package may break completeness.
Nevertheless, the default value aims completeness
while be tight enough not to trigger a superfluous rerun.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency.
The default value is given below.
-->
            <patternReRunLatex>
^(LaTeX Warning: Label\(s\) may have changed\. Rerun to get cross-references right
    \.$|
Package \w+ Warning: .*Rerun( .*|\.)$|
Package \w+ Warning: .*$^\(\w+\) .*Rerun .*$|
LaTeX Warning: Etaremune labels have changed\.$|
\(rerunfilecheck\)                Rerun to get outlines right$)
            </patternReRunLatex>

            <!-- The maximal allowed number of reruns of $latex2pdfCommand.
This is to avoid endless repetitions.
The default value is 5.
This shall be non-negative
or -1 which signifies that there is no threshold. -->
            <maxNumReRunsLatex>-1</maxNumReRunsLatex>


            <!-- The BibTeX command to create a bbl-file
from an aux-file and a bib-file
(using a bst-style file).
The default value is 'bibtex'. -->
            <bibtexCommand>bibtex</bibtexCommand>

            <!-- The options for the command $bibtexCommand.
The default value is the empty string. -->
            <bibtexOptions />

            <!-- The pattern is applied linewise to the blg-file
and matching indicates that $bibtexCommand failed.
The default value is chosen
according to the 'bibtex' documentation. -->
            <patternErrBibtex>error message</patternErrBibtex>

            <!-- The pattern is applied linewise to the blg-file
and matching indicates a warning $bibtexCommand emitted.
The default value is chosen
according to the 'bibtex' documentation. -->
            <patternWarnBibtex>^Warning--</patternWarnBibtex>

            <!-- The MakeIndex command to create an ind-file
from an idx-file logging on an ilg-file.
The default value is 'makeindex'. -->
            <makeIndexCommand>makeindex</makeIndexCommand>

            <!-- The options for the command $makeIndexCommand.
The default value is the empty string. -->
            <makeIndexOptions>-c</makeIndexOptions>
```

```
            <!-- The pattern is applied linewise to the ilg-file
and matching indicates that $makeIndexCommand failed.
The default value '(!! Input index error )'
is chosen according to the 'makeindex' documentation. -->
            <patternErrMakeIndex>
(!! Input index error )
            </patternErrMakeIndex>

            <!-- The pattern is applied linewise to the ilg-file
and matching indicates a warning $makeIndexCommand emitted.
The default value '(## Warning )'
is chosen according to the 'makeindex' documentation. -->
            <patternWarnMakeIndex>(## Warning )</patternWarnMakeIndex>

            <!-- The pattern is applied linewise to the log-file
and matching triggers rerunning $makeIndexCommand
followed by $latex2pdfCommand.

This pattern only matches a warning
emitted by the package 'rerunfilecheck'
e.g. used with option 'index'.
The default value
is chosen according to the package documentation.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the bug. -->
            <patternReRunMakeIndex>
(^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get index right\.$)
            </patternReRunMakeIndex>

            <!-- The SplitIndex command to create ind-files
          from an idx-file logging on ilg-files.
          This command invokes $makeIndexCommand.
          The default value is 'splitindex'. -->
            <splitIndexCommand>splitindex</splitIndexCommand>

            <!-- The options for $splitIndexCommand.
          Here, one has to distinguish between the options
          processed by $splitIndexCommand
          and those passed to $makeIndexCommand.
          The second category cannot be specified here,
          it is already given by $makeIndexOptions.
          In the first category is the option '-m'
          to specify the $makeIndexCommand.
          This is used automatically and cannot be specified here.
          Since $splitIndexCommand is used
          in conjunction with package 'splitidx',
          which hardcodes various parameters
          which are the default values for $splitIndexCommand
          and because the option may not alter certain interfaces,
          the only option which may be given explicitly
          is '-V', the short cut for '--verbose'.
          Do not use '--verbose' either for sake of portability.
          The default value is '-V'; it could also be empty. -->
            <splitIndexOptions>-V</splitIndexOptions>

            <!-- The MakeGlossaries command to create a gls-file
from a glo-file (invoked without file ending)
also taking ist-file or xdy-file
into account logging on a glg-file.
```

```
The default value is 'makeglossaries'. -->
            <makeGlossariesCommand>makeglossaries</makeGlossariesCommand>

            <!-- The options for the command $makeGlossariesCommand.
These are the options for 'makeindex'
(not for $makeIndexCommand)
and for 'xindy' (also hardcoded).
The aux-file decides on whether program is executed
and consequently which options are used.

The default value is the empty option string.
Nevertheless, 'xindy' is invoked as
'xindy -L english -I xindy -M ...'.
With option '-L german', this is added.
Options '-M' for 'xindy' '-s' for 'makeindex' and '-t' and '-o' for both, 'xindy'
     and 'makeindex'. -->
            <makeGlossariesOptions />

            <!-- The pattern is applied linewise to the glg-file
and matching indicates that $makeGlossariesCommand failed.
The default value '(^\*\*\* unable to execute: )'
is chosen according to the 'makeindex' documentation.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the bug. -->
            <patternErrMakeGlossaries>
 (^\*\*\* unable to execute: )
            </patternErrMakeGlossaries>

            <!-- The pattern in the glg-file
indicating an error when running 'xindy'
via $makeGlossariesCommand .
The default value is '(^ERROR: )'
(note the space and the brackets).
If this is not appropriate, please modify
and notify the developer of this plugin. -->
            <!--patternErrXindy>(^ERROR: )</patternErrXindy-->

            <!-- The pattern is applied linewise to the glg-file
and matching indicates a warning when running 'xindy'
via $makeGlossariesCommand.

The default value '(^WARNING: )'
(note the space and the brackets)
is chosen according to the 'xindy' documentation.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the deficiency. -->
            <patternWarnXindy>(^WARNING: )</patternWarnXindy>


            <!-- The pattern is applied linewise to the log-file
and matching triggers rerunning $makeGlossariesCommand
followed by $latex2pdfCommand.

This pattern only matches a warning
emitted by the package 'rerunfilecheck'
e.g. used with option 'glossary'.
The default value
is chosen according to the package documentation.
```

```
If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the bug. -->
            <patternReRunMakeGlossaries>
(^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get glossary right\.$)
            </patternReRunMakeGlossaries>

            <!-- The tex4ht command.
Possible values are e.g. 'htlatex' and 'htxelatex'.
The default value (for which this software is also tested)
is 'htlatex'. -->
            <tex4htCommand>htlatex</tex4htCommand>

            <!-- The options for the 'tex4ht'-style
which creates a dvi-file or a pdf-file
with information to create sgml,
e.g. html or odt or something like that.
The default value is 'html,2'.

Format:
<Output format>, <index>, <depth>,
['info'], ['next'], ['fn-in'], ['frames'],
['pmathml'], ['pmathml-css'], ...

options in [] are optional
DEFAULT: html,2
Available formats are html, xhtml, mathml, ooffice index=2  index in 2 columns.
depth is the depth of sectioning
to which separate files are created.
fn-in specifies inline footnotes
frames specifies separate frames for contents and toc
mathml specifies mathml
uni-html4 is used for unicode
-->
            <!-- xhtml,uni-html4,0 mathml,
,uni-html4,2,svg-->
            <tex4htStyOptions>
xhtml,uni-html4,2,pic-tabular
            </tex4htStyOptions>

            <!-- options for tex4ht.c, default is empty -->
            <!-- '-cunihtf' forces unicode -->
            <tex4htOptions> -cunihtf -utf8</tex4htOptions>

            <!-- The options for 't4ht' which converts idv-file and lg-file
into css-files, tmp-file and,
by need and if configured accordingly into png files.
The value '-p' prevents creation of png-pictures.
The default value is the empty string. -->
            <t4htOptions>-cvalidate</t4htOptions>

            <!-- The pattern is applied to file names
and matching shall accept
exactly the target files of goal 'html'
for a given latex main file 'xxx.tex'.
Matching triggers copying those files to $outputDirectory.

The patterns for the other targets
are hardcoded and take the form
'^T$T\.yyy$', where 'yyy'
may be an ending or an alternative of endings.
```

```
This pattern is neither applied to directories
nor to 'xxx.tex' itself.

For an explanation of the pattern 'T$T',
see $patternCreatedFromLatexMain.
Spaces and newlines are removed
from that pattern before processing.

The pattern is designed to match quite exactly
the files to be copied to $targetSiteDirectory,
for the goal 'html',
not much more and at any case not less.
since $tex2htCommand is not well documented,
and still subject to development,
this pattern cannot be guaranteed to be final.

If the current default value is not appropriate,
please overwrite it in the configuration
and notify the developer of this plugin of the bug.
The default value is given below. -->
            <patternT4htOutputFiles>
 ^(T$T(((ch|se|su|ap|li)?\d+)?\.x?html?|
       \.css|
 \d+x\.x?bb|
 \d+x\.png|
 -\d+\.svg)|
      (cmsy)\d+(-c)?-\d+c?\.png)$
            </patternT4htOutputFiles>

            <!-- The latex2rtf command to create rtf from latex directly.
The default value is 'latex2rtf'. -->
            <latex2rtfCommand>latex2rtf</latex2rtfCommand>

            <!-- The options of the command $latex2rtfCommand.
The default value is the empty string. -->
            <latex2rtfOptions />

            <!-- The odt2doc command
to create MS word-formats from otd-files.
This command comes with package unoconv.
The default value is 'odt2doc'. -->
            <odt2docCommand>odt2doc</odt2docCommand>

            <!-- The options of the command $odt2docCommand.
Above all specification of output format
via the option '-f'.
Invocation is 'odt2doc -f<format> <file>.odt'.
All output formats are shown by 'odt2doc --show'
but the formats interesting in this context
are doc, doc6, doc95, docbook, docx, docx7, ooxml, rtf.
Interesting also the verbosity options
'-v', '-vv', '-vvv'
the timeout '-T=secs' and '--preserve'
to keep permissions and timestamp
of the original document.
The default value is '-fdocx'. -->
            <odt2docOptions>-fdocx</odt2docOptions>

            <!-- The pdf2txt-command for converting pdf-files
into plain text files.
  The default value is 'pdftotext'. -->
            <pdf2txtCommand>pdftotext</pdf2txtCommand>
```

```
            <!-- The options of the command $pdf2txtCommand.
The default value is the empty string. -->
            <pdf2txtOptions />

            <!-- The chktex-command for checking latex main files.
The default value is 'chktex'. -->
            <chkTexCommand>chktex</chkTexCommand>

            <!-- The options of the command $chkTexCommand,
except '-o output-file'
specifying the output file which is added automatically.

Here is a list of options useful in this context.
The first group of these are muting options:
- '-w', '-e', '-m',
Make the message number passed as parameter
a warning/an error/a message and turns it on.
Messages are not counted.
- '-n'
Turns the warning/error number passed as a parameter off.
- '-L'
Turns off suppression of messages on a per line basis.

The next group of interesting options are for output control:
- '-q'
Shuts up about copyright information.
- '-o output-file'
Specifies the output file. This is added automatically
and shall thus not be specified by the user.
- '-b[0|1]'
If you use the -o switch, and the named outputfile exists,
it will be renamed to 'filename.bak'.
- '-f format'
Specifies the format of the output
via a format similar to 'printf()'.
For details consult the manual.
- '-vd'
Verbosity level followed by a number 'd'
specifying the format of the output.
The verbosity number is resolved as a pattern
as if given by the option '-f format'.
Thus the option '-v' is ignored
if the option '-f format' is specified.

The default value is '-q -b0'
avoiding verbose output and backing up the output log-file. -->
            <chkTexOptions>-q -b0 -v1 -g0 -l ${basedir}/src/site/tex/chktexrc</
                chkTexOptions>


            <!-- The diff-command for diffing pdf-files strictly or just visually
      to check that the created pdf files are equivalent with prescribed ones.
      CAUTION: there are two philosophies:
      Either the latex source files are created in a way that they reproduce
          strictly.
      Then a strict diff command like 'diff' is appropriate.
      Else another diff command is required which checks for a kind of visual
          equality.
      The default value is a mere 'diff'.
      Alternatives are 'diff-pdf' and 'diff-pdf-visually'
      both implementing a visual diff.
```

```
      Note that unlike for other tools, no options can be passed in this case
          explicitly. -->

            <diffPdfCommand>diff</diffPdfCommand>

          </settings>
        </configuration>

        <executions>
          <execution>
            <id>process-latex-sources-pdf</id>
            <goals>
              <goal>pdf</goal>
            </goals>
          </execution>

          <!-- tied implicitly to phase site -->
          <execution>
            <id>process-latex-sources</id>
            <goals>
              <goal>cfg</goal>
            </goals>
          </execution>
          <!-- tied implicitly to phase clean -->
          <execution>
            <id>clear-latex-sources</id>
            <goals>
              <goal>clr</goal>
            </goals>
          </execution>
          <!-- tied implicitly to phase validate -->
          <execution>
            <id>validate-converters</id>
            <goals>
              <goal>vrs</goal>
            </goals>
            <configuration>
              <versionsWarnOnly>true</versionsWarnOnly>
            </configuration>
          </execution>

        </executions>
      </plugin>
```

Listing 8.2: The full configuration and executions of this maven plugin

It has to be placed in the build element where below dots are given.

```
<build>
  <plugins>
    ...
  </plugins>
</build>
```

The ant-task described here requires the target and task definitions in `build.xml` given by Listing 8.3.

```
<!-- CAUTION: currently, this has a problem with bootstrapping:
must be started with maven: mvn install first.
```

```
After mvn clean ant build does not work.
-->

<project name="Main" default="jar" basedir=".">
  <property name="targetSiteDir"    value="${basedir}/target/site" />
  <property name="targetDir"        value="${basedir}/target" />
  <property name="version"          value="1.5-SNAPSHOT" />
  <!-- the following must be adapted to the local installation. -->
  <property name="antJarDir"        value="/usr/share/ant/lib/" />
  <property name="createdJar"
            value="latex-maven-plugin-${version}-antTask.jar" />


  <target
      name="init"
      description="Initializes all properties.">
    <property name="clsDir"             value="target/classes" />
    <property name="tstClsDir"          value="target/test-classes/"/>
    <!--property name="targetDir"          value="target" /-->
    <property name="srcJavaDir"         value="src/main/java/" />
    <property name="compileClsPath"     value="${clsDir}:${tstClsDir}" />
    <property name="encoding"           value="ISO-8859-1"/>

    <!-- ensure that all required directories are present.
         these are removed by mvn only. -->
    <!--mkdir dir="${targetDir}"/-->
    <mkdir dir="${clsDir}"/>
  </target>


  <target name="clean"
          depends="init"
          description="Delete all generated files
                       including tests and site.">
    <delete includeemptydirs="true" failonerror="false">
      <fileset dir="${targetDir}/**">
      </fileset>
    </delete>
  </target>


  <target name="jar"
          depends="init"
          description="Creates a jar-file defining the ant-task
                       and copies it where ant finds it.">
     <javac srcdir="${srcJavaDir}"
            classpath="${antJarDir}ant.jar"
            destdir="${clsDir}"
            encoding="${encoding}"
            debug="${javac.debug}"
            debuglevel="none"
            excludes="eu/simuline/m2latex/mojo/**"
            includeAntRuntime="false"
            compiler="modern"
            fork="yes">
     </javac>
     <jar
         destfile="${targetDir}/${createdJar}"
         basedir="${clsDir}"
         includes="**/*.class" excludes="${createdJar}">
     </jar>
  </target>
```

```
  <target name="install"
          description="Copies␣the␣relevant␣jar-file␣defining␣the␣ant-task
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣to␣where␣ant␣finds␣it:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣in␣the␣installation,␣not␣in␣the␣local␣environment.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣This␣must␣be␣invoked␣as␣root.␣">
      <copy file="${targetDir}/${createdJar}"
            todir="${antJarDir}"/>
  </target>

  <target name="link"
          description="Copies␣the␣relevant␣jar-file␣defining␣the␣ant-task
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣to␣where␣ant␣finds␣it:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣in␣the␣installation,␣not␣in␣the␣local␣environment.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣This␣must␣be␣invoked␣as␣root.␣">
     <symlink link="${antJarDir}/${createdJar}"
              resource="${targetDir}/${createdJar}"/>
  </target>

  <target name="uninstall"
          depends="jar"
          description="Deletes␣the␣relevant␣jar-file␣defining␣the␣ant-task
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣where␣ant␣finds␣it:
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣in␣the␣installation,␣not␣in␣the␣local␣environment.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣This␣must␣be␣invoked␣as␣root.␣">
       <delete file="${antJarDir}${createdJar}" />
       <symlink action="delete" link="${antJarDir}${createdJar}"/>
  </target>

 <!-- deactivate the following unless the ant task is installed already -->

  <path id="latex.classpath">
    <fileset dir="${antJarDir}">
      <include name="${createdJar}"/>
    </fileset>
  </path>

  <taskdef name="latexCfg"
           classname="eu.simuline.m2latex.antTask.LatexCfgTask"
           classpathref="latex.classpath"/>

  <target name="latex:cfg"
          description="Create␣pdf,␣html␣and␣other␣formats␣from␣latex.␣">
    <latexCfg>
        <!--patternErrXindy="(^ERROR:␣)" -->

        <!--             texPath='' -->
      <settings texSrcDirectory="src/site/tex"
                texSrcProcDirectory='.'
                readTexSrcProcDirRec='true'
                outputDirectory="."
                targets="pdf,␣html"
                convertersExcluded=''
                cleanUp='true'
                fig2devCommand="fig2dev"
                fig2devGenOptions=""
                fig2devPtxOptions=""
                fig2devPdfEpsOptions=""
                gnuplotCommand="gnuplot"
                gnuplotOptions=""
                metapostCommand="mpost"
                svg2devCommand='inkscape'
```

```
                    svg2devOptions='-z -D --export-latex'
                    ebbCommand='ebb'
                    ebbOptions='-v'
                    latex2pdfCommand="pdflatex"
                    debugBadBoxes='true'
                    debugWarnings='true'
                    pdfViaDvi='false'
                    dvi2pdfCommand='dvipdfmx'
                    dvi2pdfOptions=''
                    maxNumReRunsLatex='-1'
                    bibtexCommand="bibtex"
                    bibtexOptions=""
                    makeIndexCommand="makeindex"
                    makeIndexOptions="-c"
                    splitIndexCommand='splitindex'
                    splitIndexOptions="-V"
                    makeGlossariesCommand="makeglossaries"
                    makeGlossariesOptions=""
                    patternErrMakeGlossaries="(^\*\*\* unable to execute: )"
                    patternWarnXindy="(^WARNING: )"
                    tex4htCommand="htlatex"
                    tex4htStyOptions="xhtml,uni-html4,2,svg,pic-tabular"
                    tex4htOptions=" -cunihtf -utf8"
                    t4htOptions="-cvalidate"
                    latex2rtfCommand="latex2rtf"
                    latex2rtfOptions=""
                    odt2docCommand="odt2doc"
                    odt2docOptions="-fdocx"
                    pdf2txtCommand="pdftotext"
                    pdf2txtOptions=""
                    chkTexCommand='chktex'
                    chkTexOptions='-q -b0'>
<!-- -v1 -g0 -l ${basedir}/src/site/tex/chktexrc nowhere used. -->
        <patternLatexMainFile>
^(\\RequirePackage\s*(\[(\s|\w|,)*\])?\s*\{\w+\}\s*(\[(\d|\.)+\])?|
\\PassOptionsToPackage\s*\{\w+\}\s*\{\w+\}|
%.*$|
\\input\{[^{}]*\}|
\s)*
\\(documentstyle|documentclass)
        </patternLatexMainFile>

        <patternCreatedFromLatexMain>
^(
T$T(                              \.([^.]*|synctex\.gz|out\.ps)|
      (-|ch|se|su|ap|li)?\d+\.x?html?|
                           \d+x\.x?bb|
                          \d+x?\.png|
                           -\d+\.svg|
                             -.+\.(idx|ind|ilg))|
zzT$T\.e?ps|
(cmsy)\d+(-c)?-\d+c?\.png|
(pdf)?latex\d+\.fls)$
        </patternCreatedFromLatexMain>

        <metapostOptions>
          -interaction=nonstopmode -recorder -s prologues=2
        </metapostOptions>

        <latex2pdfOptions>
          -interaction=nonstopmode
          -synctex=1
```

```
          -src-specials
          -recorder
          -shell-escape
        </latex2pdfOptions>
        <patternErrLatex>(^! )</patternErrLatex>
        <patternWarnLatex>
^(LaTeX Warning: |
LaTeX Font Warning: |
(Package|Class) .+ Warning: |
pdfTeX warning( \((\d|\w)+\))?: |
\* fontspec warning: |
Missing character: There is no .* in font .*!$|
A space is missing\. (No warning)\.)
        </patternWarnLatex>
        <patternReRunLatex>
^(LaTeX Warning: Label\(s\) may have changed\. Rerun to get cross-references right
    \.$|
Package \w+ Warning: .*Rerun( .*|\.)$|
Package \w+ Warning: .*$^\(\w+\) .*Rerun .*$|
LaTeX Warning: Etaremune labels have changed\.$|
\(rerunfilecheck\)                Rerun to get outlines right$)
        </patternReRunLatex>

        <patternErrBibtex>error message</patternErrBibtex>
        <patternWarnBibtex>^Warning--</patternWarnBibtex>

        <patternErrMakeIndex>(!! Input index error )</patternErrMakeIndex>
        <patternWarnMakeIndex>(## Warning )</patternWarnMakeIndex>
        <patternReRunMakeIndex>
          (^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get index right\.$)
        </patternReRunMakeIndex>

        <patternReRunMakeGlossaries>
          (^\(rerunfilecheck\) +Rerun LaTeX/makeindex to get glossary right\.$)
        </patternReRunMakeGlossaries>

        <patternT4htOutputFiles>
          ^(T$T(((ch|se|su|ap|li)?\d+)?\.x?html?|
                \.css|
                \d+x\.x?bb|
                \d+x\.png|
                -\d+\.svg)|
          (cmsy)\d+(-c)?-\d+c?\.png)$
        </patternT4htOutputFiles>
      </settings>
    </latexCfg>
  </target>

  <taskdef name="latexClr"
           classname="eu.simuline.m2latex.antTask.LatexClrTask"
           classpathref="latex.classpath"/>

  <!-- very bad: copied parameters -->
  <target name="latex:clr"
          description="Delete files created in latex source directory. ">
    <latexClr>

        <!--patternErrXindy="(^ERROR: )" -->

      <settings texSrcDirectory="src/site/tex">
        <patternLatexMainFile>
          \s*\\(documentstyle|documentclass).*
```

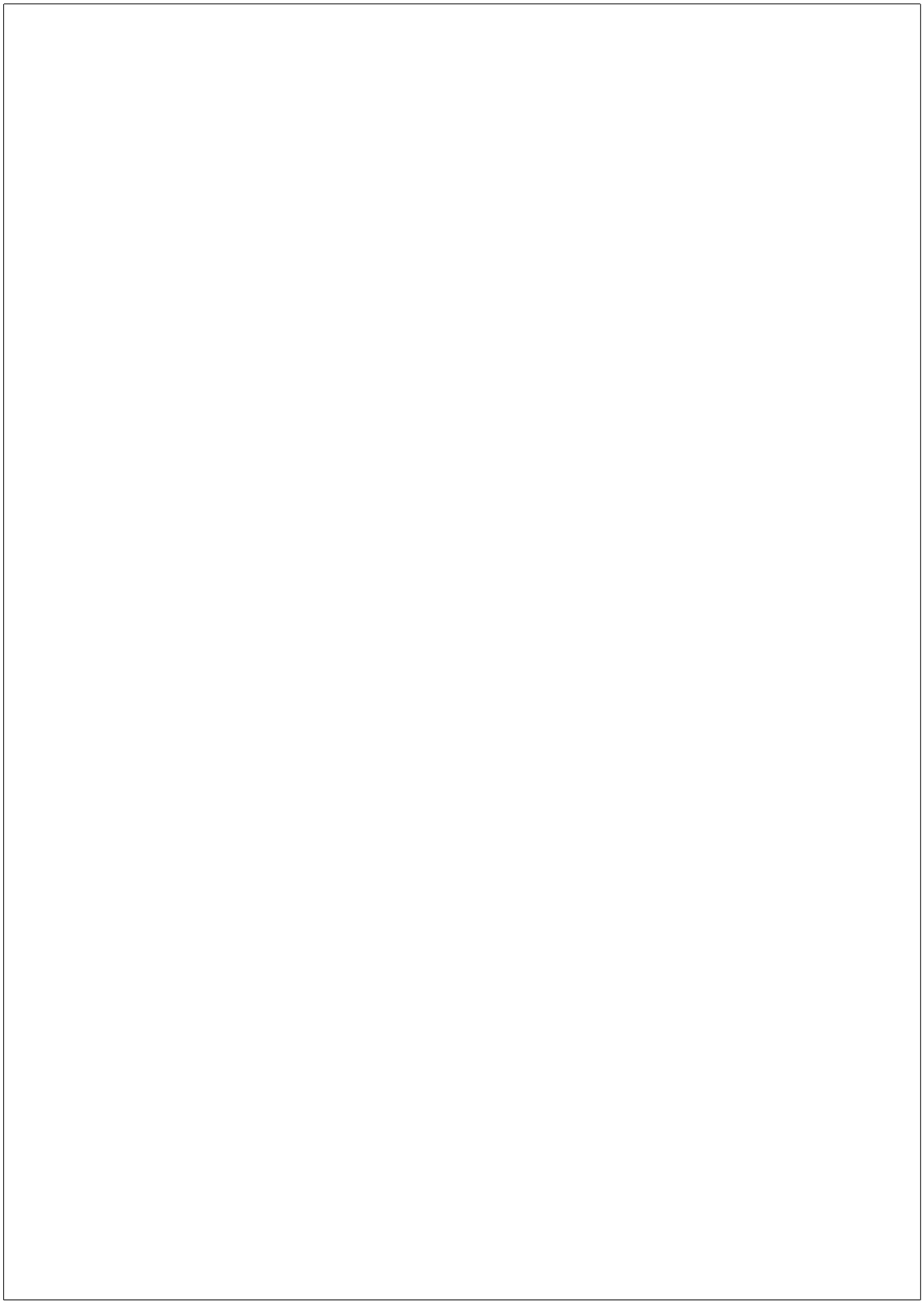
```
        </patternLatexMainFile>

        <patternCreatedFromLatexMain>
^(
T$T(                                \.([^.]*|synctex\.gz|out\.ps)|
      (-|ch|se|su|ap|li)?\d+\.x?html?|
                           \d+x\.x?bb|
                          \d+x?\.png|
                           -\d+\.svg|
                             -.+\.(idx|ind|ilg))|
zzT$T\.e?ps|
(cmsy)\d+(-c)?-\d+c?\.png|
(pdf)?latex\d+\.fls)$
        </patternCreatedFromLatexMain>

      </settings>
    </latexClr>
  </target>

</project>
```

Listing 8.3: The definition of this ant task and target



Chapter 9

Gaps

Only figures created with xfig and stored as files pdf and ptx may be integrated into a LaTeX document. This could be extended to a broader variety of export file formats. The problem is, that fig-files to not contain information on the export format. This has to be either given elsewhere in a config file or determined by pre-parsing the tex-files.

There is no support for pictures in gif-format but maybe a converter to png is all needed.

There is no proper make-mechanism taking dependencies into account. Thus all documents in all formats specified are remade, whether they changed or not.

Also, if more than one target is created from one LaTeX source, common steps are redone for each target. E.g. if pdf and html are created, pdf creation is done twice and if pdf, html, odt and docx are created, odt is done twice (once for odt second for docx) and pdf is done even trice: once for pdf itself, once for odt and once for docx.

Creating more than one index is supported only via package `splitidx` in conjunction with `SplitIndex`. There are the following packages also supporting multiple indices but not supported officially: `index` described in [Jon95], `amsmidx` described in [Bee07] and `imakeidx` described in [Gre16]. Note that the package `multind` is obsolete.

According to [Tal16b], Section 1, there are three options to create a glossary, whereas this software supports option two only, which uses `makeindex`. Also, although the package `glossaries` itself supports multiple glossaries via the command

```
\newglossary[log-ext]{name}{in-ext}{out-ext}{title}[counter]
```

described in [Tal16b], Section 12, this software only supports creating a single glossary.

Reading [Tal16b], Section 15.1, the glossarystyle `index` seems to allow creating indices through the `glossaries` package making any index-package obsolete.

For development given the LaTeX main file `xxx.tex`, the files `xxx.pdf`, `xxx.pdf`, `xxx.synctex.gz` and `xxx.log` are vital. Thus it would be fine to have a goal which touches these files or to have a parameter to touch these prior to creation to avoid that these are cleaned up after the run. This is an alternative to setting parameter `cleanup` to `false`. On the other hand, goal `grp` creating graphics in conjunction with a development tool like `auctex` for `emacs`, allows to compile a latex main file in that tool and thus to access `xxx.log` and `xxx.pdf`.

There are lots of possible improvements to be done on the goal *check*.

The ant-task does not allow to create single formats, e.g. pdf selectively.

The ant-build is not completed: tests are not run and test runs are no prerequisite for installation.

This manual is not finished. To test the overall functionality of the maven-plugin and of the ant-task described here, this manual is created through plugin and task.

Support for djvu via pdf2djvu: pdf2djvu -o output_file input_file

pdf2dsc (ps with document structuring convention)

pdf2svg is not so useful.

pdftohtml -c is also not bad,

consider also pdftocairo for creation of tiff and ps and many others.

Chapter 10

Bugs

Seemingly, indices and glossaries based on page numbers (there seems to be an alternative to this), may be out of date with the current algorithm: First `pdflatex` is run to create the raw index. Then a sorting program like `makeindex` is called which creates the sorted, collected and formatted index. Then one `pdflatex` run is required to include this index into the created pdf-file. A second `pdflatex` run is required to write the index to the table of contents, as typically required. The problem with this procedure is, that the subsequent runs of `pdflatex` change the raw index which requires rerunning `makeindex` and after that again `pdflatex`.

One way to solve that problem is to use the package `imakeidx` (improved makeidx) instead of the traditional package `makeidx`. This offers also multiple indices, which is another gap to be filled. Seemingly, `imakeidx` does not support glossaries and so for these, another solution is required, although the problem is the same.

Packages `robustindex` and `robustglossaries` offer another solution. The advantage would be to have handled both index and glossary. Also support of hyperrefs within indices and glossaries seem to be expanded. On the other hand, the two packages seem experimental and seem to to play with package `hyperref`.

The current implementation is based on package `rerunfilecheck` which works for index but not for glossary.

Check whether `glossaries` option `autorun` makes sense. Seems to run the command `makeglossaries` after each latex run. But how to find out whether to rerun latex???

Pattern to identify latex main file: Documentation: shall not include the environment `documentclass/documentstyle` in an input. Also check `RequiresPackage` and check whether `(re)newcommand` is possible or makes sense.

Maybe there is a bug in the number of reruns: I think, makeglossaries is like bibtex needing two latex reruns and not like makeindex, which requires a single rerun.

Since this software heavily relies on `rerunfilecheck`, maybe a warning if not used is a good idea.

Figures are missing in html output Formulae are missing in html output. Index is s missing in html output. Glossary occurs in the toc but is not numbered.

Did not find a way to add a numbered entry for the glossary into the table of contents.

The pattern `(! )` detects an error only `-no-file-line-error` (which is the default) is set but does not work with option `-file-line-error`. This yields

```
./manualLatexMavenPlugin.tex:2500: Undefined control sequence.
l.2500 \bla
```

instead of

```
! Undefined control sequence.
l.2500 \bla
```

I ask myself how to detect this error in file line error mode!

Pattern matching is line-wise. This is inappropriate for patternLatexMainFile but also for further patterns like multiline-warnings.

Also there seems to be a bug in java's regex package, which leads to non-termination: pattern `(\s*)*xx` seems not to terminate.

A problem is also that the ending "`.svg`" may occur as a source and as a target file of htlatex. Thus `mvn latex:clr` tries to delete the targets of the svg-files, although these are not sources but themselves targets.

A way to solve this problem is, to apply the delete pattern to graphic source files and the files created. CAUTION: for svg, the files created by the latex run shall be taken into account. A warning shall be issued for each matching.

Target html: references to figures are missing. jpg and png-pictures oddly represented. With option svg: problem. Leave away, then at least the formula occurs. But then, from the mixed pictures only the text occur, whereas the pdf is still missing. Maybe htlatex still relies on eps-format. Table is very wide. Umlauts and sz maybe also not properly represented.

Still for target html: currently all aspects making problems are deactivated: Figures, index and glossary. For the index have a look at the log-file. These aspects must be re-integrated as soon as possible.

For html: run package tex4ht with option `info` to obtain further options and their descriptions. Also add a proper description into this manual.

For files `.directory` ("`.`" first), the separation of root and suffix does not work. Maybe the best to ignore files like that.

Target `txt`: seems as if index and glossary not up to date.

target `pdf`: Idea to run `makeglossaries` always prior to pdflatex.

Maybe this is more a gap than a bug: support for dvi-creation should be provided separately.

For target dvi, neither png nor jpg-pictures are included. The other formats work with `$pdfViaDvi` set. Note that the postscript-files must be in the same directory as the dvi, probably because it includes them only by link.

For the other case, `$pdfViaDvi` unset, this requires some research.

Also for creation of the txt-format, `$pdfViaDvi` must be set.

FIXME: on bibliography, index and glossary

The application `chktex` does not necessarily return an error code if something goes wrong, e.g. reading `-l chktexrc`. Thus only in debug mode one can recognize the misbehavior. This knocks out detection of build failures.

Also I would like to replace the global `chktexrc` by a local version, via '`-g0 -l chktexrc.my`'. The problem is, that the file is interpreted relative to the working directory.

The application chtex has an option `-I` to specify, whether input files shall be read. If not, creation of graphics is immaterial. I can also imagine, that one wants to configure, whether graphics shall be created or not.

It may make sense to define in `chktexrc` another verbosity level with format allowing to decide whether there is a warning/error/message. Now I modified the levels that all but the headlines start with blank. This makes it easy in -v1 and in -v2 to detect warning/error/message at the beginning of a line, without the risk of false error because a message is logged on a text starting with the word "error".

Maybe this is not a bug but an inconsistency between auctex and local config; Running with the plugin, we obtain

```
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014) (preloaded format=pdflatex 2014.8.9)  30 JAN 2017 10:58
entering extended mode
 \write18 enabled.
 Source specials enabled.
 %&-line parsing enabled.
**test.tex
(./test.tex
```

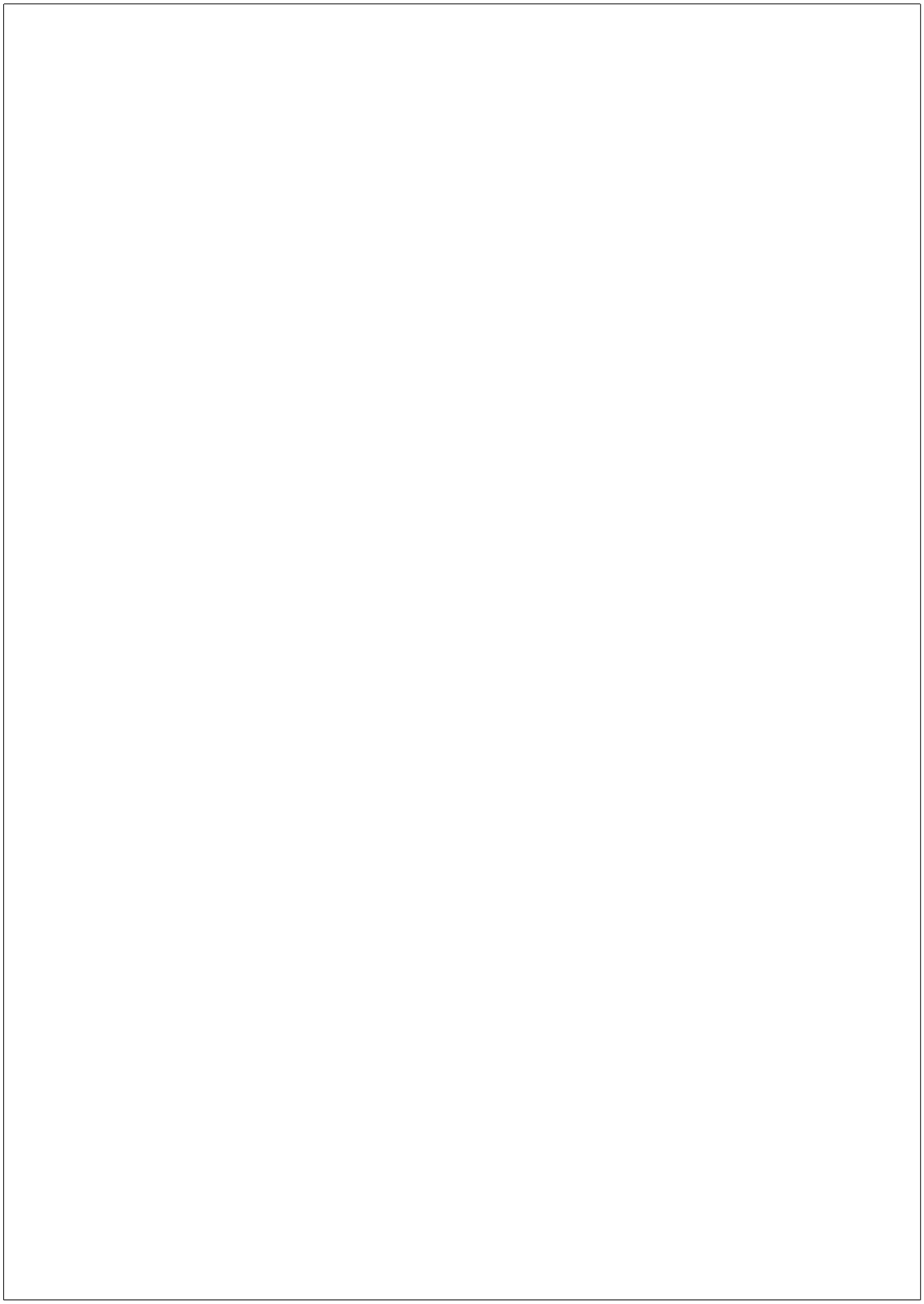
whereas running from within Emacs with auctex we obtain

```
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014) (preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
```

and also the behavior is slightly different, e.g. on file

```
\documentclass{article}
\begin{document}
 äö¿
\end{document}
```

The parameter `patternReRunLatex` treated in Section 6.3.3 needs more careful investigation. This is done a little bit in class `org.m2latex.core.Settings`.



Chapter 11

Tests

FIXME: this chapter describes the tests to be performed.

Missing are tests on logging, tests on various input formats, output formats, tests including several paths defined by invocation of auxiliary applications for index, glossary, ...

Chapter 12

Bibliography

[Aa08] Ola Andersson and al. Scalable Vector Graphics (SVG) Tiny 1.2 Specification. Technical report, W3C, `https://www.w3.org/TR/SVG/`, 12. 2008.

[Ars09] Donald Arseneau. *The import package*. asnd@triumf.ca, 3. 2009. This manual corresponds to import v5.1, dated 23–Mar–2009.

[BB16] Javier Bezos and Johannes L. Braams. *Babel*, version 3.9r edition, 4. 2016.

[Bee07] B. Beeton. *The amsmidx package*. American Mathematical Society, `https://www.ctan.org/pkg/amsmidx`, version 2.02 edition, 9. 2007.

[BLC+14] J. Braams, L. Lamport, D. Carlisle, F. Mittelbach, R. Schöpf, A. Jeffrey, and C. Rowley. *Standard LaTeX 2ε packages makeidx and showidx*. LaTeX Project, `http://latex-project.org/bugs.html`, 9. 2014.

[Car98] David Carlisle. *The longtable package*, v4.09 edition, 5. 1998.

[Car14] David Carlisle. *The ifthen package*, v1.1c edition, 9. 2014.

[Car16] D. P. Carlisle. *Packages in the 'graphics' bundle*. `https://www.ctan.org/pkg/graphicx`, 5. 2016. The LaTeX3 Project.

[Da11] Erik Dahlström and al. Scalable Vector Graphics (SVG) 1.1 Specification. Technical report, W3C, `https://www.w3.org/TR/SVG/`, 8. 2011.

[DHH02] David Duce, Ivan Herman, and Bob Hopgood. Svg tutorial. Technical report, Oxford Brookes University, W2C, 2002.

[Fea16] Simon Fear. *Publication quality tables in LaTeX*. 300A route de Meyrin, Meyrin, Switzerland, v1.618033 edition, 4. 2016.

[Grä96] George Grätzer. *Math into LaTeX*. Springer Science, New York, 1996.

[Gre16] E. Gregorio. *The package imakeidx*. `https://www.ctan.org/pkg/imakeidx`, v1.3e edition, 10. 2016. Enrico.Gregorio@univr.it.

[Hei16] The LaTeX3 project, `https://github.com/ho-tex/oberdiek/issues`. *The ifpdf Package*, v3.1 edition, 5. 2016. A re-implementation of Heiko Oberdiek's ifpdf package.

[HMH15] Jobst Hoffmann, Brooks Moses, and Carsten Heinz. *The Listings Package*. j.hoffmann(at)fh-aachen.de, v1.6 edition, 6. 2015.

[Hob14] John D. Hobby. *METAPOST, a user's manual*, 5. 2014. for version 1.999.

[Ilt12] Philip Ilten. *The svg Package*, v1.0 edition, 9. 2012. philten@cern.ch.

[ISOa] International Organization for Standardization (ISO). *Document management – Electronic document file format for long-term preservation – Part 1: Use of PDF 1.4 (PDF/A-1)*.

[ISOb] International Organization for Standardization (ISO). *Document management – Electronic document file format for long-term preservation – Part 2: Use of ISO 32000-1 (PDF/A-2)*.

[ISO12] International Organization for Standardization (ISO). *Document management — Electronic document file format for long-term preservation — Part 3: Use of ISO 32000-1 with support for embedded files (PDF/A-3)*, 2012.

[JM15] Alan Jeffrey and Frank Mittelbach. *inputenc.sty*. The LaTeX project, `http://latex-project.org/`, v1.2c edition, 3. 2015.

[Jon95] David M. Jones. *A new implementation of LaTeX's indexing commands*. `https://www.ctan.org/tex-archive/macros/latex/contrib/index?lang=en`, v4.1beta edition, 9. 1995.

[Ker16] Uwe Kern. *Extending LaTeX's color facilities: the xcolor package*. `www.ukern.de/tex/xcolor.html`, xcolor@ukern.de, v2.12 edition, 5. 2016.

[Koh16] M. Kohm. *Creating More Than One Index Using splitidx and SplitIndex*. `https://www.ctan.org/pkg/splitindex?lang=en`, v1.2c edition, 2. 2016. komascript@gmx.info.

[Kwo88] C. Kwok. *EEPIC Extensions to epic and LaTeX Picture Environment Version 1.1*. Department of Electrical Engineering and Computer Science, University of California, Davis, California, 2. 1988. `https://www.ctan.org/pkg/eepic?lang=de`.

[Lam87] Leslie Lamport. MakeIndex : An Index Processor For LaTeX, 2. 1987.

[LRZ] MakeIndex - ein Indexprozessor fuer LaTeX. `https://www.lrz.de` Menues: services,software,textverarbeitung,makeindex.

[Mar09] Nicolas Markey. Tame the beast - the b to x of bibtex. manuscript, 10. 2009. markey@lsv.ens-cachan.fr.

[MFL16] Frank Mittelbach, Robin Fairbairns, and Werner Lemberg. *LaTeX font encodings*. The LaTeX3Project Team, 2. 2016.

[Mös98] Peter Mösgen. Makeindex Sachregister erstellen mit LaTeX. Katholische Universität Eichstätt Universitätsrechenzentrum, 5. 1998.

[ner16] Ernst Reißner. The xfig file format for xfig 3.2. `http://www.simuline.eu/LatexMavenPlugin/xfig/xfigFormat.pdf`, 12. 2016.

[ner17] Ernst Reißner. The dvi format and the program dvitype. `http://www.simuline.eu/LatexMavenPlugin/dvi/dviFormat.pdf`, 1. 2017.

[Obe16a] Heiko Oberdiek. *The bmpsize package*. heiko.oberdiek@googlemail.com, v1.7 edition, 5. 2016.

[Obe16b] Heiko Oberdiek. *The ifluatex package*. heiko.oberdiek@googlemail.com, v1.4 edition, 5. 2016.

[Obe16c] Heiko Oberdiek. *The rerunfilecheck package*, v1.8 edition, 5. 2016.

[Obe16d] Heiko Oberdiek. *The transparent package*, v1.1 edition, 5. 2016.

[Pat88] Oren Patashnik. Bibtexing. manuscript, 2. 1988.

[PDF] Adobe Systems Incorporated 2008. *Document management — Portable document format — Part 1: PDF 1.7*.

[RO22] Sebastian Rahtz and Heiko Oberdiek. *Hypertext marks in LaTeX: a manual for hyperref*, 2. 2022.

[Rob10] Will Robertson. *The ifxetex package*. wspr81@gmail.com, v0.6 edition, 9. 2010.

[Sch11] Ulrich Michael Schwarz. *The nag package*. absatzen, `http://absatzen.de/`, ulmi@absatzen.de, 11. 2011. corresponds to nag 0.7, dated 2011/11/19.

[Sch16] R. Schlicht. *The microtype package*. w.m.l@gmx.net, v2.6a edition, 5. 2016.

[Sio17] Laurens Sion. *The pdfprivacy package*. laurens@sion.info, v1.0 edition, 12. 2017.

[SMCR15] Walter Schmidt, Frank Mittelbach, David Carlisle, and Chris Rowley. *The fix-cm package*. The LaTeX Project Team, v1.1t edition, 1. 2015.

[SU06] A. Simonic and S. Ulrich. *srcltx.sty · srctex.sty*. stefanulrich@users.sourceforge.net, v1.6 edition, 11. 2006.

[Sza07] Péter Szabó. *The anyfontsize package*. pts@fazekas.hu, 2. 2007.

[TAK+14] Kresten Krab Thorup, Per Abrahamsen, David Kastrup, et al. *AUCTEX A sophisticated TEX environment for Emacs*. Free Software Foundation, Inc., version 11.88 edition, 10 2014.

[Tal16a] N. L.C. Talbot. The glossaries package v4.26: a guide for beginners. `https://www.ctan.org/pkg/glossaries?lang=de`, 10. 2016.

[Tal16b] N. L.C. Talbot. *User Manual for glossaries.sty v4.26*. dickimaw-books, `http://www.dickimaw-books.com/`, 10. 2016.

[Tan15] T. Tantau. *TikZ and PGF Manual for Version 3.0.1a*. Institut für Theoretische Informatik, Universität zu Lübeck, Lübeck, Germany, 8. 2015. `http://sourceforge.net/projects/pgf`.

[Tea00] The LaTeX3Project Team. *LaTeX2ε font selection*, 9. 2000.

[Thi16] Jens T. Berger Thielemann. *ChkTEX v1.7.6*. Jens Berger, Spektrumvn. 4, N-0666 Oslo, jensthi@ifi.uio.no, 9. 2016.

[Ume10] Hideo Umeki. *The geometry package*. latexgeometry@gmail.com, v5.6 edition, 9. 2010.

[Wic99] Mark A. Wicks. *Dvipdfm User's Manual*, 9. 1999. Version 0.12.4b.

[WK20] Thomas Williams and Colin Kelley. *gnuplot 5.4 – An Interactive Plotting Program*. `http://sourceforge.net/projects/gnuplot`, 7. 2020. Version 5.4.

[WP10] P. Wilson and H. Press. *The tocbibind package*. latex-project, `https://www.ctan.org/pkg/tocbibind?lang=de`, v1.5k edition, 10. 2010.

[Zan10] Timothy Van Zandt. *The 'fancyvrb' package Fancy Verbatims in LaTeX*. Princeton University, tvz@Princeton.EDU, v2.8 edition, 5. 2010.

Chapter 13

General Index

Chapter 14

LaTeX Packages

Glossary

aux auxiliary file. 48

doc outdated document format for MS word. 25

docx current document format for MS word. 11, 12, 25, 47

dvi device independent file format. 11, 34, 47, 60

eps encapsulated postscript. 14, 34, 35, 42

fig native file format for xfig. 14, 24, 27, 33

gif Graphics Interchange Format, allows also animations. 45, 123

glo glossary file containing unsorted and multiple glossary entries. 48

gls glossary file containing sorted, unified and formatted glossary entries. 49

gp Gnuplot file format. 33

html hypertext markup language. 11, 12, 15, 34, 47

idx index file containing unsorted and multiple index entries. 48, 51

ind index file containing sorted, unified and formatted index entries. 48, 51

jpg Graphics format developed by the Joint Photographic Experts Group. 24, 33, 34, 45

mp metapost. 14, 24, 41

mps metapost's postscript like output including text. 24, 41

mpx metapost tex output: texts. 41

odt open document text. 11, 12, 47

pdf portable document format. 11, 14, 24, 47

png Portable Network Graphics. 24, 33, 34, 42, 45, 60, 82, 123

sgml Standard generalized markup language. 15

svg Scalable Vector Graphics. 14, 24, 33, 34, 42, 43

xhtml extensible hypertext markup language. 12

xml extensible markup language. 15