Arakhnê.ORG

LaTeX Packages for the Unified Process Methodology

LaTeX Packages for the Unified Process Methodology

Official Documentation

Reference:	UPM-2007-01
Version:	7.0
Updated:	2007/07/07
Status:	Public

Authors: Stéphane Galland

Document Summary

Project	Document	Reference	Version	Last Update
LaTeX Packages for the Unified Process Methodology	Official Documentation	UPM-2007-01	7.0	2007/07/07

Authors

Names	Emails
Stéphane Galland	galland@arakhne.org

Validators

Names	Emails	Initials
Stéphane Galland	galland@arakhne.org	

Version History

Version	Date	Updates
1.0	2006/04/11	First release of this example
2.0	2006/04/19	Add the mtabular and mtable environments
3.0	2006/04/27	Add the package upmethodology-code
4.0	2007/04/13	Update the upmethodology-fmt package Adding commands for informed people
5.0	2007/07/02	Update the upmethodology-fmt package Adding symbols and text formatting functions Updating footnote functions
6.0	2007/07/04	Add front page selection macro
7.0	2007/07/07	Add bibliography format macros

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This set of package permits to write documents according to the Unifess Process Methodology. It was initially wittren by Stéphane Galland from the Systems and Transport laboratory[1].
The packages are:

- `upmethodology-version.sty`: permits to set the version and the status of the document. It also permits to manage the document history;
- `upmethodology-fmt.sty`: provides some usefull functions to format the UP documents;
- `upmethodology-document.sty`: provides functions to manage the project, the subproject and the status of the document;
- `upmethodology-frontpage.sty`: formats and provides a front page for the document;
- `upmethodology-task.sty`: is the LaTeX 2_ε package that provides commands to manage project's tasks.
- `upmethodology-document.cls`: is the LaTeX 2_ε class that provides the whole document specification. It is based on `book` and on the previous packages;
- `upmethodology-code.sty`: provides macros for source code formatting.

[1]University of Technology of Belfort-Montbéliard, France, http://set.utbm.fr/

Chapter 2

Package upmethodology-version

`Version: 2007/07/03`

The package `upmethodology-version` permits to set the version and the status of the document. It also provides functions to manage the document history;

2.1/ Constants for the Document Status

Some LaTeX 2_ε variables provides strings that describe the status of the document. They can be used in functions such as `\updateversion`.

- `\upmrestricted`: the document is under a restricted access, generally corresponding to the list of authors;
- `\upmvalidable`: authors indicates with this tathat the document could be sent to validators;
- `\upmvalidated`: the document was validated, but not published and published by all people;
- `\upmpublic`: the document published and accessible by all people;

2.1.1/ Information about the Document

The following functions permit to access to the informations about the document:

- `\theupmversion`: replies the last version number for the document;
- `\upmdate{version}`: replies the updating date of the document corresponding to the given version number;
- `\upmdescription{version}`: replies the updating comment of the document corresponding to the given version number;
- `\upmstatus{version}`: replies the status of the document corresponding to the given version number.
- `\theupmdate`: replies the last updating date for the document. It is equivalent to `\upmdate{\theupmversion}`;
- `\theupmlastmodif`: replies the last updating comment for the document. It is equivalent to `\upmdescription{\theupmversion}`;
- `\theupmstatus`: replies the last status for the document. It is equivalent to `\upmstatus{\theupmversion}`;

2.2/ Register Revisions

The package `upmethodology-version` permits to register revisions for building an history. The available functions are:

- `\updateversion{version}{date}{description}{status}`: registers an revision for the document. The revision indicates that the given version was produced at the given date. A small description of the changes and the resulting document's status must be also provided. The function `\updateversion` is a generalization of the following functions;
- `\initialversion[version]{date}{description}{status}`: registers the initial version of the document. If not given, the version is assumed to be `0.1`;
- `\incversion{date}{description}{status}`: regiters a revision corresponding to the next major version. For example, if the version number was `2.67` before `\incversion`, this function add the version `3.67` with the given informations (incrementation of the major part of the version number);
- `\incsubversion{date}{description}{status}`: regiters a revision corresponding to the next minor version. For example, if the version number was `2.67` before `\incsubversion`, this function add the version `2.68` with the given informations (incrementation of the minor part of the version number);

2.3/ Formatted List of Versions

To obtain a formatted list of versions, you could use the command `\upmhistory[width]` which produces:

Version History

Version	*Date*	*Updates*
1.0	2006/04/11	First release of this example
2.0	2006/04/19	Add the mtabular and mtable environments
3.0	2006/04/27	Add the package upmethodology-code
4.0	2007/04/13	Update the upmethodology-fmt package
		Adding commands for informed people
5.0	2007/07/02	Update the upmethodology-fmt package
		Adding symbols and text formatting functions
		Updating footnote functions
6.0	2007/07/04	Add front page selection macro
7.0	2007/07/07	Add bibliography format macros

2.4/ Localization

The following commands defines some localized strings used by `upmethodology-version`:

- `\upm@lang@date`: Date;
- `\upm@lang@updates`: Updates;
- `\upm@lang@version`: Version;
- `\upm@lang@version@history`: Version History;

Chapter 3

Package upmethodology-fmt

`Version: 2007/07/06`

The package `upmethodology-fmt` provides some usefull facilities to format an UP document.

3.1/ Figures

You could include afigure inside your document with the following commands:

```
\mfigure[position]{options}{filename}{caption}{label}
\mfigure*[position]{options}{filename}{caption}{label}
```

These two commands permits to include an image in your document.The parameters are:

- `position`: is the desired position of the figure (see \beginfigure[position]). It could be `t` (top of the page), `b` (bottom of the page), `h` (at the command location if possible) or `H` (at command location);
- `options`: are the options passed to `\includegraphics`;
- `filename`: is the filename passed to `\includegraphics`;
- `caption`: is the caption of the figure (see \caption{caption});
- `label`: is the label used to reference the figure (see \label{fig:label}).

The difference between `\mfigure` and `\mfigure*` is the same as the difference between `\begin{figure}` and `\begin{figure*}`: the star-version fits to the entire paper width event if the document has two or more columns.
Because the two commands abose register a label with string starting with `fig:`, we propose the following function to easily access to the figure's references:

- `\figref{label}`: is equivalent to `\ref{fig:label}`;
- `\figpageref{label}`: is equivalent to `\pageref{fig:label}`.

The figure 3.1 page 10 is obtained with the command: `\mfigure[ht]{width=.6\linewidth}{slogo}{Example of figure inclusion with \texttt{{\textbackslash}mfigure}}{example:mfigure}`. The reference and page reference are obtained with `\figref{example:mfigure}` and `\figpageref{example:mfigure}`.

Figure 3.1: Example of figure inclusion with `\mfigure`

3.2/ Sub-figures

In some case, it is usefull to put several images inside the same picture, but without lousing the possibility to reference each subfigure. This feature was proposed by the package `subfigure`. The following environments provides helper functions for `subfigure`:

```
\begin{mfigures}[position]{caption}{label}
...
\end{mfigures}
\begin{mfigures*}[position]{caption}{label}
...
\end{mfigures*}
```

These two commands permits to include an image in your document.The parameters are:

- `position`: is the desired position of the figure (see \beginfigure[position]). It could be `t` (top of the page), `b` (bottom of the page), `h` (at the command location if possible) or `H` (at command location);
- `caption`: is the caption of the figure (see \caption{caption});
- `label`: is the label used to reference the figure (see \label{fig:label}).

Inside the environment `\mfigures[*]`, you could use the command `\mfigure` to properly include a subfigure (the first optional parameter is ignored) or you could use the command `\msubfigure{options}{file}{caption}`.
The figure 3.2 page 10 is obtained with the environment:

```
\begin{mfigures}{Example of subfigures with \texttt{mfigures}}{example:msubfigure}
\mfigure{width=.4\linewidth}{logo}{First subfigure}{example:firstsubfigure}
\hfill
\msubfigure{width=.4\linewidth}{smalllogo}{Second subfigure}
\end{mfigures}
```

The reference and page reference are obtained with `\figref{example:msubfigure}` and `\figpageref{example:msubfigure}`.

(a) First subfigure

(b) Second subfigure

Figure 3.2: Example of subfigures with `mfigures`

The references to the subfigures could be obtained in two way:

- using the label given as the last parameter of `\mfigure`, eg. the label `example:firstsubfigure` corresponds to 3.2(a);
- using the label of the enclosing figure to which the index of the subfigure could be appended (in its roman representation and prefixed by the character "`:`"), eg. the label `example:msubfigure:b` corresponds to 3.2(b);

3.3/ Figures with embedded TeX commands

In several case it is usefull to include TeX commands inside a figure. It is possible with the files `.pstex` exporting from a software such as `xfig`.
To put a TeX command inside your figure, follows the steps:

1. in `xfig` create a text label with the *special* property set. In this label types the string `\FIG`δ where δ must be replaced by an identifier of your choice but only composed of letters (example: `\FIGmyid`);
2. in `xfig` saves your figure as `.pstex` files;
3. in LaTeX, just before including the figure with the embedded TeX commands, define the expressions to put in the figure. This action must be done with one of the commands:
   - `\figmath{id}{expr}` will associate to the given identifier the given mathematical expression,
   - `\figtext{id}{expr}` will associate to the given identifier the given text expression;
4. in LaTeX, include the figure with one of the commands:
   `\mfigurewtex[position]{width}{filename}{caption}{label}`
   `\mfigurewtex*[position]{width}{filename}{caption}{label}`

3.4/ Tabulars

You could include a tabular inside your document with the following environment:
`\begin{mtabular}[width]{ncolumns}{columns}...\end{mtabular}`

This tabular is an extension of the `tabularx` environment which provides dynamic columns with the specifier `X`. The parameters are:

- `width`: is the desired width of the tabular;
- `ncolumns`: is the count of columns in the tabular. It must be consistent with the column description;
- `columns`: is the description of the columns according to the `tabular` and `tabularx` packages.

The `mtabular` environment provides:

- `\tabulartitle{title}`
  this command permits to define the title of the tabular. It uses the colors `backtableheader` and `fronttableheader` for the background and the foreground respectively;

- `\tabularheader{`*header*$_1$`}...{`*header*$_n$`}`
  this command permits to define the titles of the columns. It uses the colors `backtableheader` and `fronttableheader` for the background and the foreground respectively. Because the count of columns was given to the environment this function takes the same count of parameters as the count of columns.

The following example of table is obtained by:

```
\begin{mtabular}[\linewidth]{4}{lXrX}
\tabulartitle{Example of \texttt{mtabular}}
\tabularheader{Col1}{Col2}{Col3}{Col4}
a & b & c & d \\
\hline
e & f & g & h \\
\end{mtabular}
```

Example of `mtabular`			
Col1	*Col2*	*Col3*	*Col4*
a	b	c	d
e	f	g	h

3.5/ Tables

You could include a table inside your document with the following environment:
`\begin{mtable}[position]{width}{ncolumns}{columns}{caption}{label}...\end{mtable}`

This environment is based on the `mtabular` environment. The parameters are:

- `position`: is the desired position of the table according to the LaTeX's `table` definition;
- `width`: is the desired width of the table (ie., the tabular inside the table);
- `ncolumns`: is the count of columns in the table (ie., the tabular inside the table). It must be consistent with the column description;
- `columns`: is the description of the columns according to the `tabular` and `tabularx` packages;
- `caption`: is the caption of the table;
- `label`: is the label referencing the table.

Because the `mtable` environment registers a label with a string starting with `tab:`, the following functions are proposed to easily access to the table's references:

- `\tabref{label}`: is equivalent to `\ref{tab:label}`;
- `\tabpageref{label}`: is equivalent to `\pageref{tab:label}`.

The table 3.1 page 13 is an illustration of the following LaTeX code:

```
\begin{mtable}{\linewidth}{4}{lXrX}{Example of \texttt{mtable}}{example:mtable}
\captionastitle
\tabularheader{Col1}{Col2}{Col3}{Col4}
a & b & c & d \\
\hline
e & f & g & h \\
\end{mtable}
```

Example of mtable			
Col1	*Col2*	*Col3*	*Col4*
a	b	c	d
e	f	g	h

Table 3.1: Example of `mtable`

The command `\captionastitle` is equivalent to a call to the macro `\tabulartitle` with the caption in parameter.

3.6/ Enumerations

The package `upmethodology-fmt` provides a set of commands dedicated to enumeration lists.

3.6.1/ Enumeration Counters

Sometimes it is usefull to start an enumeration list from a specifical given number. This package provides several commands for saving and restoring the counter use by the enumeration lists.
Caution: only once counter could be saved at a given time.

- `\savecounter{name}`
  save the counter identifier by the given name;
- `\restorecounter{name}`
  put the previously saved value into the given counter;
- `\setenumcounter{value}`
  force the value of the enumeration counter;
- `\getenumcounter`
  replies the value of the enumeration counter;
- `\saveenumcounter`
  save the enumeration counter;
- `\restoreenumcounter`
  force the enumeration to use the saved counter's value;

3.6.2/ Inline Enumeration

In several document, an enumeration of things is written inside a paragraph instead of inside a list of points. A simple example is: (i) first thing; (ii) second thing; (iii) etc. And it is produced by the LaTeX code:

```
\begin{inlineenumeration}
\item first thing;
\item second thing;
\item etc.
\end{inlineenumeration}
```

3.7/ Footnotes

The package `upmethodology-fmt` provides a set of commands allowing to save the reference number of a footnote and to recall this reference many time as required.

- `\savefootnote{footnote text}{footnote id}`
  put a footnote and mark it with the corresponding label.
  Example: `\savefootnote{This is an example of a recallable footnote}{footrecalla}`[1];

- `\savefootnote*{footnote text}{footnote id}`
  mark a footnote with the corresponding label but do not put in the current page.
  Example: `\savefootnote*{This is a second example of a recallable footnote}{footrecallb}`;

- `\reffootnote{footnote id}`
  recall the footnote reference without page number.
  Example: `\reffootnote{footrecalla}`[1].

- `\reffootnote*{footnote id}`
  recall the footnote reference with the page number if different of the current page.
  Example: `\reffootnote*{footrecallb}`[2].

3.8/ UML diagrams on the side of paragraphs

The package `upmethodology-fmt` provides an environment which permits to put an UML diagram (or any other picture) on the side of a paragraph.

- `\begin{umlinpar}[width]{picture_path}`
  `text`
  `\end{umlinpat}`
  put the specified picture on the side of the given text. The optional parameter `width` corresponds to the desired width ofthe picture. By default it is `.5\linewidth`.

[1]This is an example of a recallable footnote
[2]This is a second example of a recallable footnote

This paragraph is an typical example of the usage of the environment `umlinpar`. To obtain it, the following LaTeX code was typed:

```
\begin{umlinpar}{smalllogo}
This paragraph is an typical example
of the usage of the environment
\texttt{umlinpar}.
\end{umlinpar}
```

3.9/ Date formatting

Because the concept of date was important and unfortunately localized, this package provides a set of functions to define and extract information from dates (the supported date formats are described in table 3.2):

- `\makedate{day}{month}{year}`
  permits to create the text corresponding to the given date according to the current localized date format.
- `\extractyear{formatted_date}`
  extract the year field from a date respecting the localized date format.
- `\extractmonth{formatted_date}`
  extract the month field from a date respecting the localized date format.
- `\extractday{formatted_date}`
  extract the day field from a date respecting the localized date format.

`yyyy/mm/dd`	default format
`dd/mm/yyyy`	french format

Table 3.2: List of supported date formats

3.10/ Text formatting

The package `upmethodology-fmt` provides a set of commands to format the text.

- `\textsup{text}`
  put a text as exponent in text mode instead of the basic LaTeXexponent in math mode.
  Example: `\textsup{this is an exponent}`$^{\text{this is an exponent}}$;
- `\textsub{text}`
  put a text as indice in text mode instead of the basic LaTeXindice in math mode.
  Example: `\textsub{this is an indice}`$_{\text{this is an indice}}$;
- `\makename[von]{first name}{last name}`
  format the specified peoople name components according to the document standards. By default, the format `first von last` is used.
  Example: `\makename{Stéphane}{Galland}`, Stéphane GALLAND ;

3.11/ Symbols

The package `upmethodology-fmt` provides several symbols described inside the table 3.3.

`\copyright`	©
`\trademark`	™
`\regmark`	®
`\smalltrade`	™
`\smallreg`	®
`\smallcopy`	©
`\st`	st
`\nd`	nd
`\rd`	rd
`\th`	th

Table 3.3: List of symbols

3.12/ Bibliography

The package `upmethodology-fmt` provides a set of commands allowing to manage the bibliography. The default bibliography style is `abbr`.

- `\bibliographystyle{style}`
  set the bibliography style to use.
  Example: `\bibliographystyle{alpha}`;
- `\bibliography{file}`
  set the BibTeX file to use.
  Example: `\bibliography{mybib}`;
- `\bibsize{size}`
  set the font size used for the bibliography section.
  Example: `\bibsize{}`;

Chapter 4

Package upmethodology-document

`Version: 2007/07/03`

The package `upmethodology-document` provides base function to manage document information (project, subproject, authors...).

4.1/ Document Information and Declaration

The informations associated to an UP document are:

- `\theupmproject` is the name of the project for which the document was produced;
- `\theupmsubproject` is the name of the sub-project for which the document was produced;
- `\theupmdocname` is the name of the document;
- `\theupmdocref` is the reference number of the document;
- `\theupmfulldocname` is the complete name of the document (composing by the project, subp-project and name of the document).

You could declare the information about your document with one of the following functions:
`\declaredocument{project}{name}{ref}`
`\declaredocumentex{project}{subproject}{name}{ref}`
where the parameters are:

- `project` is the name of the project for which the document is for;
- `subproject` is the name of the sub-project for which the document is for;
- `name` is the name of the document;
- `ref` is the reference number of the document.

4.2/ Document Summary

You can obtain a documet summary with the command `\upmdocumentsummary[width]` which produces:

Document Summary				
Project	*Document*	*Reference*	*Version*	*Last Update*
LaTeX Packages for the Unified Process Methodology	Official Documentation	UPM-2007-01	7.0	2007/07/07

4.3/ Change Icons

By default, this package uses the logo of the laboratory Systems and Transport as icons. You could change them with the commands:

- `\defupmsmalllogo{filename}` defines the small logo used in the headers for instance;
- `\defupmlogo{filename}` defines the logo used on the front page for instance.

The logos' filenames are accessible with the functions `\theupmsmalldoclogo` and `\theupmdoclogo`.

4.4/ Document Authors

An author is someone who participates to the writing of the document. You could register author identities with:
`\addauthor[email]{firstname}{name}`
`\addauthorvalidator[email]{firstname}{name}`
The list of the authors is accessible by two means:

- `\theauthorlist` is a coma-separated list of the authors' names;
- `\upmdocumentauthors` procudes an array of all the authors (see below for an example).

Authors	
Names	*Emails*
Stéphane Galland	galland@arakhne.org

4.5/ Document Validators

A validator is someone who participates to the validation of the document. You could register validator identities with:
`\addvalidator[email]{firstname}{name}`
`\addauthorvalidator[email]{firstname}{name}`
The list of the validators is accessible by two means:

- `\thevalidatorlist` is a coma-separated list of the validator' names;
- `\upmdocumentvalidators` procudes an array of all the validators (see below for an example).

Validators		
Names	*Emails*	*Initials*
Stéphane Galland	galland@arakhne.org	

4.6/ Informed People

An informed people is someone who receives the document to be informed about its content. You could register informed people identities with:
`\addinformed[email]{firstname}{name}`
The list of the informed people is accessible by two means:

- `\theinformedlist` is a coma-separated list of the informed people' names;
- `\upmdocumentinformedpeople` procudes an array of all the informed people (see below for an example).

4.7/ Document Sectionning

The package `upmethodology-document` provides several commands that permit to create special sections:

4.7.1/ Part Sectionning

If you want to add a document part that has no a part number but appearing inside the table of content, the classical LaTeX commands `\part` and `\part*` are inefficient. Indeed, `\part` add a numbered part inside the table of content, and `\part*` adds an unnumbered part but not inside the table of content.
To add a unnumbered part inside the table of content, you could use one of the commands:
`\parttoc[toctitle]{title}`
`\parttoc*[toctitle]{title}`
The commands `\parttoc` and `\parttoc*` have the same effect except that `\parttoc*` aligns the part's title to the other numbered parts' titles; and `\parttoc` not.

4.7.2/ Chapter Sectionning

If you want to add a document chapter that has no a chapter number but appearing inside the table of content, the classical LaTeX commands `\chapter` and `\chapter*` are inefficient. Indeed, `\chapter` add a numbered chapter inside the table of content, and `\chapter*` adds an unnumbered chapter but not inside the table of content.
To add a unnumbered chapter inside the table of content, you could use one of the commands:
`\chaptertoc[toctitle]{title}`
`\chaptertoc*[toctitle]{title}`
The commands `\chaptertoc` and `\chaptertoc*` have the same effect except that `\chaptertoc*` aligns the chapter's title to the other numbered chapters' titles; and `\chaptertoc` not.

4.7.3/ Section Sectionning

If you want to add a document section that has no a section number but appearing inside the table of content, the classical LaTeX commands `\section` and `\section*` are inefficient. Indeed, `\section` add a numbered section inside the table of content, and `\section*` adds an unnumbered section but not inside the table of content.
To add a unnumbered section inside the table of content, you could use one of the commands:
`\sectiontoc[toctitle]{title}`
`\sectiontoc*[toctitle]{title}`
The commands `\sectiontoc` and `\sectiontoc*` have the same effect except that `\sectiontoc*` aligns the section's title to the other numbered sections' titles; and `\sectiontoc` not.

4.7.4/ Subsection Sectionning

If you want to add a document subsection that has no a subsection number but appearing inside the table of content, the classical LaTeX commands `\subsection` and `\subsection*` are inefficient. Indeed, `\subsection` add a numbered subsection inside the table of content, and `\subsection*` adds an unnumbered subsection but not inside the table of content.
To add a unnumbered subsection inside the table of content, you could use one of the commands:
`\subsectiontoc[toctitle]{title}`
`\subsectiontoc*[toctitle]{title}`
The commands `\subsectiontoc` and `\subsectiontoc*` have the same effect except that `\subsectiontoc*` aligns the subsection's title to the other numbered subsections' titles; and `\subsectiontoc` not.

4.7.5/ Subsubsection Sectionning

If you want to add a document subsubsection that has no a subsubsection number but appearing inside the table of content, the classical LaTeX commands `\subsubsection` and `\subsubsection*` are inefficient. Indeed, `\subsubsection` add a numbered subsubsection inside the table of content, and `\subsubsection*` adds an unnumbered subsubsection but not inside the table of content.
To add a unnumbered subsubsection inside the table of content, you could use one of the commands:
`\subsubsectiontoc[toctitle]{title}`
`\subsubsectiontoc*[toctitle]{title}`
The commands `\subsubsectiontoc` and `\subsubsectiontoc*` have the same effect except that `\subsubsectiontoc*` aligns the subsubsection's title to the other numbered subsubsections' titles; and `\subsubsectiontoc` not.

4.8/ Localization

The following commands defines some localized strings used by `upmethodology-document`:

- `\upm@lang@project`: Project;
- `\upm@lang@document`: Document;

- `\upm@lang@docref`: Reference;
- `\upm@lang@lastupdate`: Last Update;
- `\upm@lang@document@summary`: Document Summary;
- `\upm@lang@document@authors`: Authors;
- `\upm@lang@document@validators`: Validators;
- `\upm@lang@document@names`: Names;
- `\upm@lang@document@emails`: Emails;
- `\upm@lang@document@initials`: Initials.

Chapter 5

Package upmethodology-frontpage

Version: 2007/07/04

The package `upmethodology-frontpage` provides an front page for the UP documents. This package does not provides any public function. It is based on all the previous packages.

5.1/ Change Front Page Layout

It is possible to change the layout of the front page with the command:
`\setfrontlayout{layout_name}`
where `layout_name` must be one of:

- `classic`: classic front page layout with title and logo;
- `modern`: front page layout with title and logo and background picture.

The figure 5.1 illustrates the differents layouts.

5.2/ Change Illustration Picture

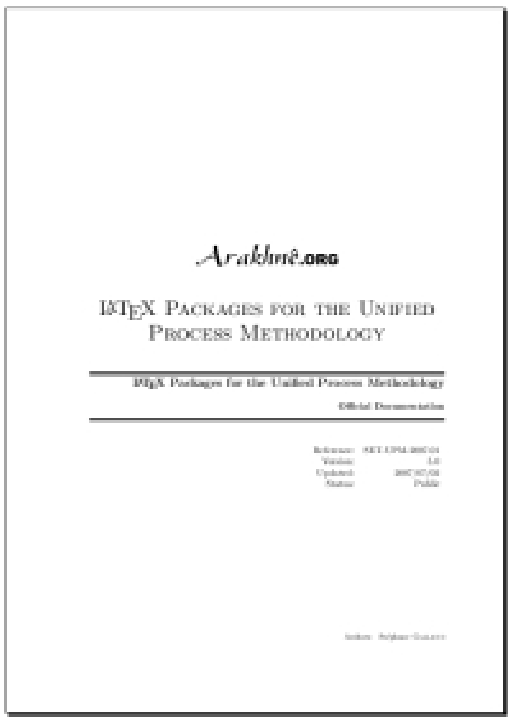
It is possible to insert an illustration picture on the front page. You could specify the image with the command:
`\setfrontillustration[width_factor]{filename}`
where:

- `width_actor` is the scaling factor of the picture according to the line width. If you specifies `1` the image will not be scaled, for `.5` the image will be the half of its original width...
- `filename` is the name of picture to use as the illustration.

5.3/ Localization

The following commands defines some localized strings used by `upmethodology-frontpage`:

- `\upm@lang@front@authors`: Authors;



Arakhnê.org

LaTeX Packages for the Unified Process Methodology

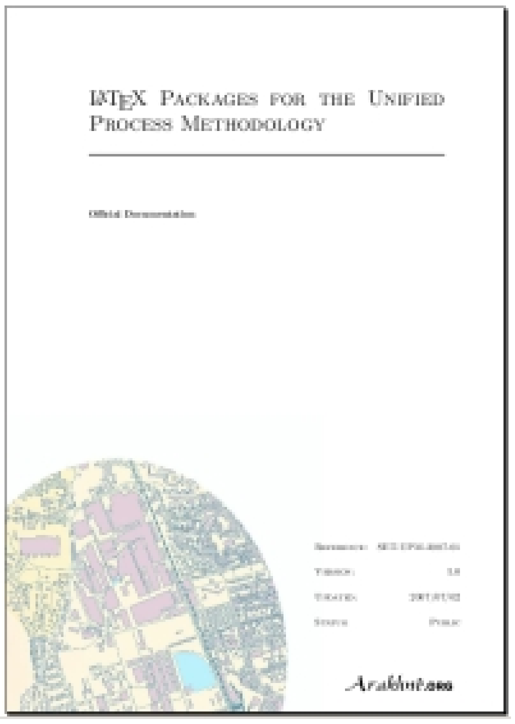
LaTeX Packages for the Unified Process Methodology

Official Documentation

Release:	SET-UPM-2007-01
Version:	1.0
Updated:	2007/07/02
Status:	Public

Author: Stéphane Galland

(a) `classic`



LaTeX Packages for the Unified Process Methodology

Official Documentation

Reference:	SET-UPM-2007-01
Version:	1.0
Updated:	2007/07/02
Status:	Public

Arakhnê.org

(b) `modern`

Figure 5.1: Front Page Layouts

Chapter 6

Package upmethodology-task

Version: 2007/03/19

The LaTeX package `upmethodology-task` provides a set of commands to define project's tasks.
This package could log the message "`Project Task(s) may have changed. Rerun to get cross-refe`
when some task information was not found or due to cross-references on them.

6.1/ Task Definition

The definition of a task could be made only inside one of the following environments:
`\begin{taskdescription}{id}...\end{taskdescription}`
`\begin{taskdescription*}{id}...\end{taskdescription*}`
where `id` is the identifier of the task.
The environment `taskdefinion` displays the task's description with a call to `\thetaskdescription{id}`. In the opposite `taskdefinition*` never displays the ta's description.
Inside one of the task's definition environment above, you could use one of the following commands to define the task's attributes:

- `\taskname{name}`
  permits to defines the name of the task;
- `\tasksuper{id}`
  indicates that the current task is a sub-task of the task identified by the given identifier;
- `\taskcomment{text}`
  permits to describe the task's purposes and goals (will be shown in the description box of the task's description);
- `\taskprogress{percent}`
  defines the percent for thtask's archieving;
- `\taskstart{date}`
  permits to set the starting date of the task (real or predicted);
- `\taskend{date}`
  permits to set the finished date of the task (real or predicted);

- `\taskmanager{name}`
  adds a task's manager into the list of the managers;
- `\taskmember{name}`
  adds a task's member into the list of the members;
- `\taskmilestone{date}{comment}`
  add a milestone into the task for the given date and described by the given comment.

6.2/ Task Reference

You could reference any information about the defined tasks in your document. In case you used cross-references this package could log the message "`Project Task(s) may have changed. Rerun to get cross-references right`" to complain about rebuilding of our document.
The following commands are available:

- `\thetasksuper{id}`
  replies the identifier of the parent task corresponding to the task identified by `id`;
- `\thetaskname{id}`
  replies the name of the the task identified by `id`;
- `\thetaskcomment{id}`
  replies the description for the the task identified by `id`;
- `\thetaskprogress{id}`
  replies the archieving percent for the the task identified by `id`;
- `\thetaskstart{id}`
  replies the starting date for the the task identified by `id`;
- `\thetaskend{id}`
  replies the ending date for the the task identified by `id`;
- `\thetaskmanagers{id}`
  replies the managers' list for the the task identified by `id`;
- `\thetaskmembers{id}`
  replies the members' list for the the task identified by `id`;
- `\thetaskmilestones{id}`
  replies the list of milestone's dates for the the task identified by `id`;
- `\thetaskmilestonecomment{id}{date}`
  replies the comment of the given milestone for the the task identified by `id`;
- `\thetaskdescription[width]{id}`
  replies the complete description of the the task identified by `id`.

6.3/ Localization

The following commands defines some localized strings used by `upmethodology-task`:

- `\upm@task@lang@task`: Task;
- `\upm@task@lang@escription`: Description;
- `\upm@task@lang@startat`: Start at;
- `\upm@task@lang@endat`: End at;
- `\upm@task@lang@archieved`: Archieved;
- `\upm@task@lang@managers`: Managers;
- `\upm@task@lang@members`: Members;
- `\upm@task@lang@Milestones`: Milestones;
- `\upm@task@lang@subtask`: Sub-task of.

Chapter 7

Class upmethodology-document

`Version: 2007/07/03`

The LaTeX class `upmethodology-document` provides an extension of the class `book` dedicated to the UP documents. This class automatically add a front page, a document summary, an author's list, a validator's list and a version history.
It also provides a default formatting of the headers of the pages.

Chapter 8

Package upmethodology-code

Version: 2006/04/27

The LaTeX package `upmethodology-code` provides a set of commands for source code formatting. The supported source codes are UML, Java and C++.
You could load the package with the following options:

`uml`	use the UML notation (default value)
`java`	use the Java notation
`cpp`	use the C++ notation

You could also change the notation language with the command:
`\upmcodelang{upm|java|cpp}`
The provided commands are listed in the following table:

Command	UML	Java	C++
Prototypes			
`\jclass{TheClass}`	TheClass	TheClass	TheClass
`\jinterface{TheInterface}`	*TheInterface*	*TheInterface*	*TheInterface*
`\jpackage{ThePackage}`	ThePackage	ThePackage	ThePackage
`\jfunc{FunctionName}`	`FunctionName`	`FunctionName`	`FunctionName`
Types			
`\jclazz`	**class**	**Class**	**class**
`\jvoid`	**void**	**void**	**void**
`\jboolean`	**boolean**	**boolean**	**bool**
`\jint`	**integer**	**int**	**int**
`\jlong`	**long integer**	**long**	**long**
`\jfloat`	**float**	**float**	**float**
`\jdouble`	**double**	**double**	**double**
`\jchar`	**character**	**char**	**char**
`\jstring`	**string**	String	std::string
`\jarray{T}`	**array of** Ts	T[]	T[]
`\jcollection{T}`	**collection of** Ts	Collection <T>	std::vector <T>
`\jset{T}`	**set of** Ts	Set <T>	std::set <T>

Command	UML	Java	C++
Constants			
`\jtrue`	TRUE	TRUE	TRUE
`\jfalse`	FALSE	FALSE	FALSE
Operations			
`\jcode{source code}`	`source code`	`source code`	`source code`
`\jcall{fct}{params}`	`fct(params)`	`fct(params)`	`fct(params)`
`\jop{operator}`	`operator`	`operator`	`operator`

Chapter 9

Authors and License