PropGen Documentation

Release 0.1

Holger Karl

March 02, 2012

CONTENTS

Contents:

CHAPTER

ONE

WHY THIS TOOL?

Writing an application for a research project is a challenging task: good ideas are needed, background research checks, a research hypothesis and a research program have to be formulated. When applying for a larger project, this is typically done by a group of people, coming from different organizations. A lot of work goes into the creative process - the mere act of writing the proposal, collecting information about the program structure, putting it in Gantt charts and tables of deliverables etc. should get out of the way!

Anybody who has tried to write a proposal for one of the European Union's Framework Programs knows that it can be a cumbersome process. The EU provides a relatively strict template which information to provide: information about work packages, tasks, deliverables, milestones, Gantt charts, etc. Much of this information is repeated at several places in the document, in various forms of presentation (tables, charts, free text). Merely keeping this information in synch can be a formidable challenge, in particular, when several people work on a proposal. To make matters worse, the EU only provides an MS Word template (and not a particularly well done one, either). There is no support to get all the administrative work out of the way.

This was the very situation we were in when we developed a proposal for a reasonably large EU proposal (an integrated project with about 15 partners). Instead of going down the Word-road, we decided to put all the information on a Wiki and to generate the actual proposal from there, using LaTeX to typeset the actual document and generating all the administrative information automatically. This has three main advantages:

- Wikis are easy to use even for novel users who are not used to using version control systems for collaborative work (let alone trying to distribute these files via email). Wikis naturally split up text in separate sections, circumventing the often problematic features of word processors to split up a document in smaller files.
- All the administrative information only needs to be entered ONCE. All possible presentations are automatically generated. They are guaranteed to stay synchronized. There is no time wasted for such work. Even non-trivial operations can be done until late in the proposal preparation without any risk (e.g., we decided to move a deliverable around a few hours before proposal submission - that would have been impossible with conventional tools).
- Wikis allow us to concentrate on the content, on our research ideas. We do not have to waste time fighting with a word processor.

To give one example: a task description for a workpackage looks as described in the figure in a moinmoin wiki. This then gets translated automatically into a Gantt chart for the workpackage (and into a Gantt chart for the project as a whole, and in tables, and in ...).

Tasks

Label	Start	Duration	Name	Lead partner
architectureDesign	1	3	Architecture Design	UE
archImprovement	4	2	Improvement of the Architecture	ABC
archImprovement	8	2	Improvement of the Architecture	ABC
archImprovement	12	2	Improvement of the Architecture	ABC
archFalsify	10	10	Falsification of the proposed Architecture	ISC

Figure 1.1: Example of a Wiki page, specifying several tasks for a workpackage (the architecture improvement task even has three phases).

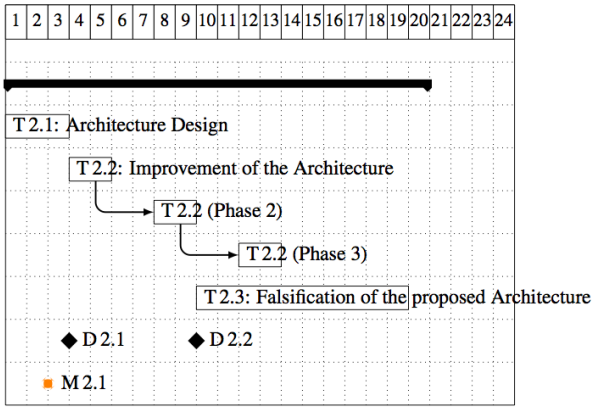


Figure 1.2: Resulting Gantt chart from the example task table (deliverable and milestones shown in this Gantt chart are defined in other Wiki tables)

Hence, the approach to go from a wiki to latex to PDF, and to submit this PDF file, has worked out nicely. It has produced a workflow that was reasonably easy for everybody, with full version control support without less IT-savy users needing to worry about it.

We felt that such a tool might be beneficial for a wider audience. So here it is - feel free to use it, to modify it, and to write interesting proposals using it. Our hope is that it will free up time from the mundane and boring tasks and enable all of us to concentrate more on the creative aspects of research.

Holger Karl

PS: When I write “we”, I refer to the team of colleagues engaged in the writing of said initial integrated research proposal. In particular, Bengt Ahlgren, Dirk Kutscher and Börje Ohlman deserve my thanks and gratitude for bearing through the rough-shot development of the inital version of this tool. I am indebted to them for constructive criticisms, ideas, and encouragement.

CHAPTER

TWO

INSTALLATION

2.1 Quick and dirty

The fastest possible way to get everything set up and produce a proposal PDF, assuming you have latex and python set up:

```
$ cd ~/tmp
$ wget --no-check-certificate https://github.com/hkarl/propgen/zipball/master --output-document=p
$ unzip propgen.zip
$ cd hkarl-propgen-7d7fd2d/
$ cd moin
$ python wikiserver.py &
$ cd ..
$ make
```

Note: the directory name of hkarl-propgen-XXXX will depend on the fingerprint of the current version in github. It will vary.

That will leave TestProject.pdf in the current directory. Use your webbrowser to go to http://127.0.0.1:8080/TestProject, make some changes to this page or to the Wiki pages linked from there, save the changes. Type make again in the shell. Gives an updated PDF file.

2.2 Usage scenarios

2.2.1 External Wiki

A typical usage scenario of this PropGen tool is the following:

- A proposal is to be prepared by a group of people.
- One of them runs the wiki, or an external Wiki provider is used
- Several people install the PropGen tool (ignoring the built-in Wiki) and can then built the PDF file for the proposal.
- Not everybody needs to install PropGen. Ideally, a version control system like SVN is integrated and whoever generates a new PDF file commits it to this version control system. Then, everybody has access to reasonably up-to-date versions.

This scenario is fairly straightforward to set up. I assume here that you have your external Wiki set up and know how to administer it.

2.2.2 Built-in Wiki

An alternative is to use the MoinMoin Wiki included in the PropGen distribution. Then, one partner has to run this wiki. Ideally and typically, the same machine is then able to run PropGen and to generate the proposal PDF.

This can conveniently be triggered via a crontab (e.g., do an hourly build) and the result can be put into a version control system similar to above.

If other partners want to setup PropGen as well to locally generate the PDF, that is no problem at all.

Advantage of this approach is that the generation scripts can directly talk to the wiki and there is no need to go over the network to pull the wiki files. This is substantially more reliable, faster, and easier to setup (in particular, if there are firewalls or proxies in place, which can be real trouble). The disadvantage is that often, a Wiki is already in place, people have their accounts on it, are accustomed to its syntax and quirks, it can have powerful features not present in the provided MoinMoin installation (e.g, Twiki has a very useful butracker that can be very beneficial during proposal writing). The choice is yours!

2.2.3 Integrating a version control system

As outlined above, it can be extremely useful to integrate a version control system like SVN. I would recommend to limit this to the latex directory, only committing files in this directory.

Nothing is done automatically here since the variety of VCS systems is large. But it should be a simple exercise to integrate corresponding commit commands in the Makefile.

However, some care has to be taken in that symbolic links from the LaTeX directory to the "generated" directory are used. The reason is that it can be convenient, towards the end of a proposal preparation process, to stop pulling some parts of a proposal from the Wiki and to rather work on the LaTeX files directly. This allows better fine-tuning then working via the Wiki. The process is simple: replace the symbolic link by the actual file. Then, this file is used and it is not touched by the generation process.

2.3 Actual installation

2.3.1 Prerequisites

You need the following software installed:

Python You will need Python version 2.7.2 or later. Python 3 is known not to work at this time, Python 2.6 is too old.

Mechanize If you want to pull in Wiki files over the network (e.g., from a remote Twiki), then you need the python mechanize module installed. Details can be found on the Mechanize webpage (http://wwwsearch.sourceforge.net/mechanize/).

Tex An up-to-date LaTeX installation. TexLive 2011 was used for development. Non-standard packages or packages which needed patches are provided in the distribution.

Make There is a simple makefile in place. It is not absolutely needed and could quite easily be replaced by shell scripts or batch files.

Bash The makefile uses some simple loop and test constructs of bash. (See the clean target, e.g.) It should not be difficult to do without or provide a version for another shell.

Operation system Development and testing took place on Mac OSX Snow Leopard. Normal Linux distributions should pose no problems at all. Installation on Window is likely to be problematic because of symbolic links, and makefiles, bash etc. is likely to require at least cygwin - but I have very little clue of Windows and dare not make any statements here. Your mileage might vary.

Sphinx If you should want to generate the documentation for the reStructuredText markup (I have no idea why you would want to do that), you will also need Sphinx (http://sphinx.pocoo.org/) , at least version 1.1.2. In particular, you will also need the aafigure extension (see http://packages.python.org/sphinxcontrib-aafig/) to be installed and added to conf.py; affigure also needs a dependency repotlab.

2.3.2 Installation

- Download the PropGen package and unpack to a folder of your choice.
  - From github:
    - Main page: https://github.com/hkarl/propgen
    - GIT Read-Only: git://github.com/hkarl/propgen.git
    - ZIP file: https://github.com/hkarl/propgen/zipball/master
  - Other sources still to come (possibly even a virtual machine)
- Decide which Wiki to use and set it up correctly.
  - Internal wiki: See *Setting up the MoinMoin Wiki included in distribution*
  - For both internal and external wiki: simply add information to settings.cfg (see *Setting up wiki in settings.cfg*)
- Add the templates to an external Wiki
  - Example templates for MoinMoin and Twiki are included in the templates folder. Ignore the directories; they are just to group the wiki files a bit. Each file becomes one Wiki page with the corresponding filename.
  - This step is not necessary when using the internal Wiki; it is pre-populated with an example pseudo project which should be easy to modify.
  - It might make sense to rename the "TestProject" page to some more specific for your project. (Then, remember to also rename the corresponding entry in settings.cfg.)
- Once you have setup Wiki access and the Wiki is running, try to generate a PDF file. cd into the main propgen directory and type make.

2.4 Setting up the MoinMoin Wiki included in distribution

If you want to use an external wiki (e.g., an existing Twiki), you can skip this section.

For more details, check the documentation of the MoinMoin Wiki.

2.4.1 Preconfigured account

The distributed version of MoinMoin is setup to support accounts, require login to edit or download material, and to deny anonymous access.

It has a preconfigured account ProjectMaster with password 123abc. This account ProjectMaster is configured as a superuser in MoinMoin. Check the lines

```
acl_rights_default = u"ProjectMaster:read,write,delete,revert,admin Known:read,write,revert,delete
```

and

```
superuser = [u"ProjectMaster"]
```

in the file moin/wikiconfig.py. It gives admin rights to the ProjectMaster account, and usual read, write, revert, delete rights to all other known accounts.

2.4.2 Change password

Obviously, you really, really want to change the password of this superuser. Log in as the user for which you want to change password, go to "Settings" (link on the very top of the page, to the left), then click "Change password".

Or go directly to http://127.0.0.1:8080/ProjectMaster?action=userprefs&sub=changepass (and replace "ProjectMaster" by the account name for which you want to change the password, of course).

2.4.3 Adding accounts

You could stick to the preconfigured account and distribute this account name and password to all members of your team. However, then it will not be possible to track who did which changes.

Hence, it is usually preferable to assign a dedicated username/password to each team member.

To add a user, you need to login as the superuser ProjectMaster. Go to the Wiki page NewUser (e.g., if you run it locally on the default port, goto http://127.0.0.1:8080/NewUser), and create as many users as you like.

2.4.4 Rename the main project page

In case you want a different main page name, simply use the "Rename page" action of the Wiki. Remember to rename the corresponding setting (projectName) in settings.cfg as well!

2.4.5 Run the MoinMoin Wiki

Simple:

```
$ cd moin
$ python wikiserver.py &
```

You might want to start this as a daemon, possibly start automatically after reboot. Consult your own operating system how to do that.

2.5 Setting up wiki in settings.cfg

The file settings.cfg contains both basic configurations to ensure that the download script talks to the right wiki server as well as basic configuration options about what kind of information to generate. The latter content-customization options are described elsewhere (Section *settings.cfg*). Here, we concentrate on basic connectivity settings.

2.5.1 Wiki

See LICENCE file for licencing information! Set up the necessary information to access the wiki: which type, where can it be found, what is the start page, which account and password to use to log in.

`projectName`
: This option specificies the root Wiki page where the main project information can be found. It also serves as the filename of the final pdf file.

  Default: TestProject

wikitype
The wikitype setting selects which type of wiki access is to be used. Wikitypes currently supported are: twiki, moinmoin and moinmoin-local

Option 1: moinmoin-local This option specifies that the a moinmoin wiki can be accessed in the same file system where the generation system executes. This option requires to sepcify the moinmoinpath option as well. No user and password need to be given, but the files must be accessible for reading.

Option 2: moinmoin

A moinmoin wiki is used and accessed remotely, using the mechanize library. To this end, both the wikiuser and wikipassword option need to be specified; they are used for login. Also, the baseURL option needs to be specified: it provides a URL where the wiki can be accessed, without any concrete page name. Example:

baseURL = http://hk-vm.cs.uni-paderborn.de:8080/

Option 3: twiki

It needs the same options as moinmoin as well as the loginURL setting. The difference is that the obtained files are parsed assuming the twiki syntax and the login process to a Twiki is slightly different from a moinmoin wiki. The distribution's configuration file example assumes a moinmoin-local

Default: moinmoin-local

moinmoinpath
Specify the moinmoin path in the standard PropGen distribution. This option is only relevant if wikitype = moinmoin-local and can be left out for other wiki types.

Default: ../moin/

baseURL
For remote access to a wiki. It specifies the "root" URL where the wiki can be accessed, without any particular page name. It is ignored for the moinmoin-local wikitype and only used for other wikitypes.

Default: http://hk-vm.cs.uni-paderborn.de:8080/

wikiuser
The wikiuser account name to use to log in. Ignored in the moinmoin-local wikitype. The default corresponds to the account name preset in the distribution's moinmoin wiki. Change this option to reflect your own user name.

Default: ProjectMaster

wikipassword
Password used to log in. The default is the one used in the distribution's example moinmoin wiki. You definitely want to change the password on the wiki and then reflect this change here. It is ignored in the moinmoin-local wikitype and only needed for remote access.

Default: 123abc

loginURL
Some remote wikis usually need a special login URL, e.g., Twiki wikis. Specify here. This setting is ignored in both the moinmoin-local wikitype (where no login is needed at all) and in the moinmoin wikitype (were the login URL is constructed directly from the baseURL and no separate URL is needed).

Default: https://twiki.sics.se/bin/login

httpProxyIP
For remote Wiki access through a proxy: These variables are used by the mechanize module, for access via an HTTP or HTTPS proxy. You can specify the proxy's IP, port number, and user and password, if needed. You can do that separately for HTTP and HTTPS access. But this does not always works out well. Also, this functionality is not well tested. Your mileage WILL vary! All the defaults are just empty.

Default:

httpProxyport
Default:

httpProxyuser
Default:

httpProxypassword
Default:

httpsProxyIP
Default:

httpsProxyport
Default:

httpsProxyuser
Default:

httpsProxypassword
Default:

CHAPTER

THREE

HOW TO USE PROPGEN, WHAT TO PUT ON WIKI, BUILDING A PDF FILE

This section describes how to work with the Wiki to enter actual project information. There are three main types of pages: the main project page, a generic text page to hold free text, and the structured work package pages that contain specific information about tasks in a work package, effort distribution, deliverables and milestones, as well as text describing the actual work to be done in a work package.

3.1 The main project page

The main project page (the Wiki page which you configured under projectName in settings.cfg) collects are the general information about the project and it links to further details providing detailed information. Only pages linked from this page are considered in building the proposal text.

It has four sections: Proposal structure, main data, workpackages, and partner data, described next. It is **important** to keep the structure of this page fixed, else it will not be possible to find relevant information. Sections are recognized by their headings; feel free to move around such sections on the page but do **not** change heading titles or the structure of the individual sections.

3.1.1 Proposal structure

This section has a bullet list, each just having the name of a Wiki page. Each of these Wiki pages will get downloaded from the Wiki and turned into LaTeX code; each such file is treated as a generic text page (see below for details). The file name of these LaTeX files will be the same as the Wiki name. The main.tex file can then include these files.

The order of the bullets in this list does not matter. The example setup has all the usual sections of a typical EU FP7 proposal and the example main.tex file is already set up to include these files in the appropiate order.

To add further pages, just add a bullet and put the Wiki name in the line. Add an include command in the right place in main.tex. to get rid of a file, simply remove it from the bullet list and remove the corresponding input command from main.tex.

There is one special case here, though: if a file includes the phrase bibtex (in whatever capitalization), it geets turned into a .bib file, not a .tex file. Usually, a single such line is suitable to point to the project's bib-file for references, but more are of course possible. The example setup includes an example with BibtexReferences, and the main.tex file is setup to use this as a BiBTeX file.

3.1.2 Main data

The main data section specifies basic data about the project as a whole.

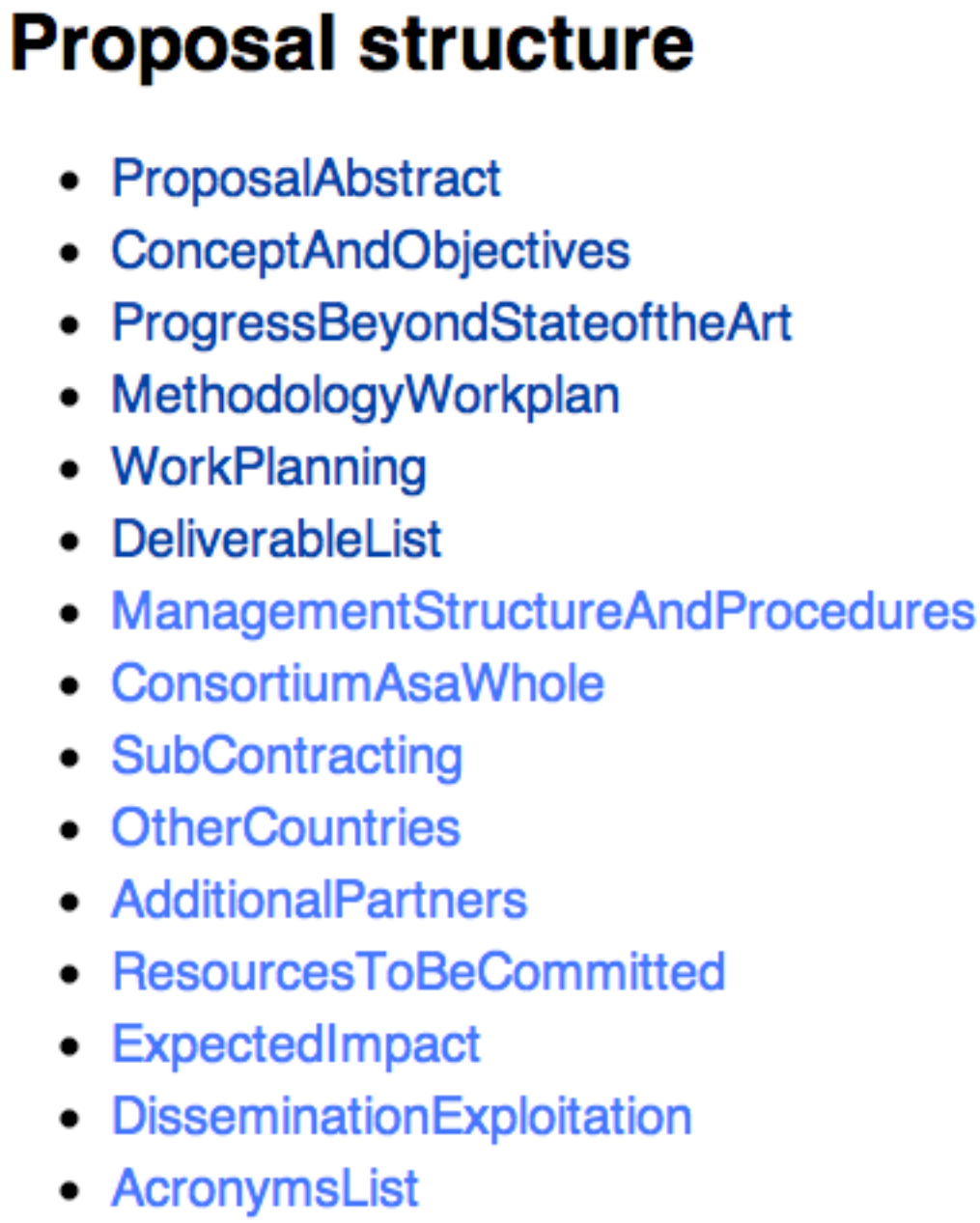


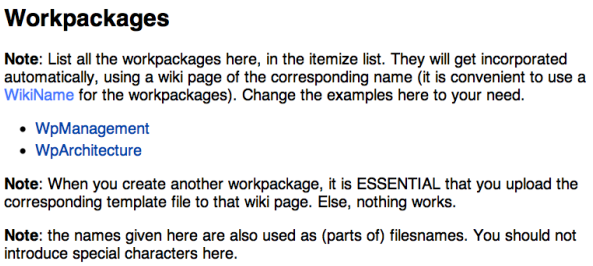
Figure 3.1: Screenshot of the example ProposalStructure section of the main page

Figure 3.2: Screenshot of main data section

Similar to the proposal structure section, it is a bullet list with individual items. Here, however, each line is a key/value combination, with the format "key: value". Most lines should be self-explanatory. Duration is the duration of the project in months; do not specify a unit, just the number.

3.1.3 Workpackages

The third part is a list of Wiki pages where a workpackage is described. Same syntax as in the proposal structure section, but the treatment of these pages is different. Unlike above, files here are not just downloaded and converted to LaTeX. Instead, a workpackage page needs to follow a certain structure to specify semantics of a workpackage. Details described below.



Workpackages

Note: List all the workpackages here, in the itemize list. They will get incorporated automatically, using a wiki page of the corresponding name (it is convenient to use a WikiName for the workpackages). Change the examples here to your need.

- WpManagement
- WpArchitecture

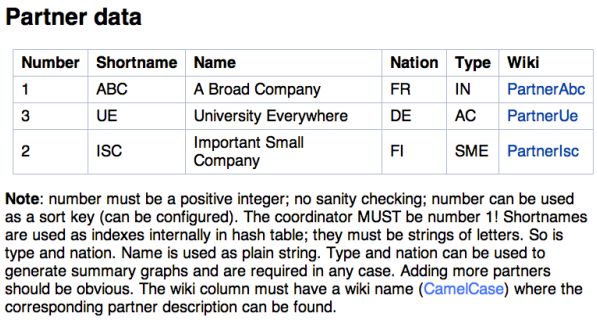
Note: When you create another workpackage, it is ESSENTIAL that you upload the corresponding template file to that wiki page. Else, nothing works.

Note: the names given here are also used as (parts of) filesnames. You should not introduce special characters here.

Figure 3.3: Screenshot of workpackage list section

3.1.4 Partner data

The partner data has more semantic structure. The important part is a table with four columns:



Partner data

Number	Shortname	Name	Nation	Type	Wiki
1	ABC	A Broad Company	FR	IN	PartnerAbc
3	UE	University Everywhere	DE	AC	PartnerUe
2	ISC	Important Small Company	FI	SME	PartnerIsc

Note: number must be a positive integer; no sanity checking; number can be used as a sort key (can be configured). The coordinator MUST be number 1! Shortnames are used as indexes internally in hash table; they must be strings of letters. So is type and nation. Name is used as plain string. Type and nation can be used to generate summary graphs and are required in any case. Adding more partners should be obvious. The wiki column must have a wiki name (CamelCase) where the corresponding partner description can be found.

Figure 3.4: Screenshot of the partner data section

Number Each partner organization needs a number, mostly used for sorting. The lead partner MUST be number 1.

Shortname The acronym to be used for this partner. It shows up in tables, lists, etc.

Name The full, official name of a partner.

Nation Stanard ISO abbreviation of the partner's legal nation.

Type Type of partners: academic, research institute, industry, ...

Wiki The name of the Wiki page where this partner's profile can be found. This wiki page is converted to LaTeX and can be included (put into the partner directory).

The order of the rows is irrelevant.

3.2 A generic text page

A generic text page is converted to LaTeX using the wikiParser class (Section *wikiParser*). It can convert a couple of the standard Wiki markup syntax into LaTeX code. The source code description provides more details.

In addition, there is a specific form of markup geared towards the generation of EU proposals: Whenever there is a pair of level-5 headings of "Start commission hints" and "End commission hints", the text between these headings is typeset AFTER the first real heading of the page (the text is moved accordingly). The example MoinMoin Wiki as well as the wiki templates contain the corresponding hints given by the commission in the official template. Whether this text is typeset in the final PDF can be controlled via the control setting "showCommissionHints" in settings.cfg. Obviously, before submitting a proposal, this setting should be set to False.

Any Wiki page may have, at the beginning, a line:

Start of text

and at the end, a corresponding line:

End of text

These lines need not be present, but if so, text before the start or after the end is ignored. It is still visible on the wiki, though - so it might be a useful place to put instructions for partners, kept discussions about the content of a page on the very page itself, etc. Note that these markers are available on all kinds of pages, also on the main page and the workpackage pages.

3.3 A workpackage page

A workpackage page is by far the most complex page, it also has a rather rigid structure. It consists of four sections, one for the workpackage's overview description, one for the task descriptions, one for adminstrative information about the WP as such, and one last section for adminstrative information on task level.

A level-1 heading can give the name of the WP; it is not analyzed.

3.3.1 Work package description

There must be a *level-2 heading* "Objectives". All the text after this headline, up to the following level-2 heading, it used to typeset the objectives of a work package.

Then, another *level-2 heading* "WP Description" is needed. Text after this heading, up to the next level-2 heading, is used to typeset a description of the WP in the WP forms.

3.3.2 Tasks descriptions

After that, the tasks inside a work package are described. The assumption is that a workpackage consists of a reasonable number of tasks (even only one, if so desired); it is not possible to do without tasks and only use workpackages. (Not without considerably reworking a lot of the templates, at least.) This part is started by a *level-2 heading* "Tasks".

Inside this section, there can be any number of *level-3 headings*, starting with "Task Description: ", and then the **symbolic label** of each task: Each task is assigned a label, typically a short word or phrase (no spaces!) via which information about this task at various places in this page can be cross-linked.

Then, there should be, per task, *two level-4 headings* "Objectives" and "Description of work". The purpose is similar to the WP descriptions, but they are specifically linked to the particular task.

3.3.3 Adminstrative information

The fourth *level-2 heading* is "Adminstrative information". It is follwed by the formal information about the workpackage. It is formatted as a bullet list with key/value pairs to provide information about the workpackage.

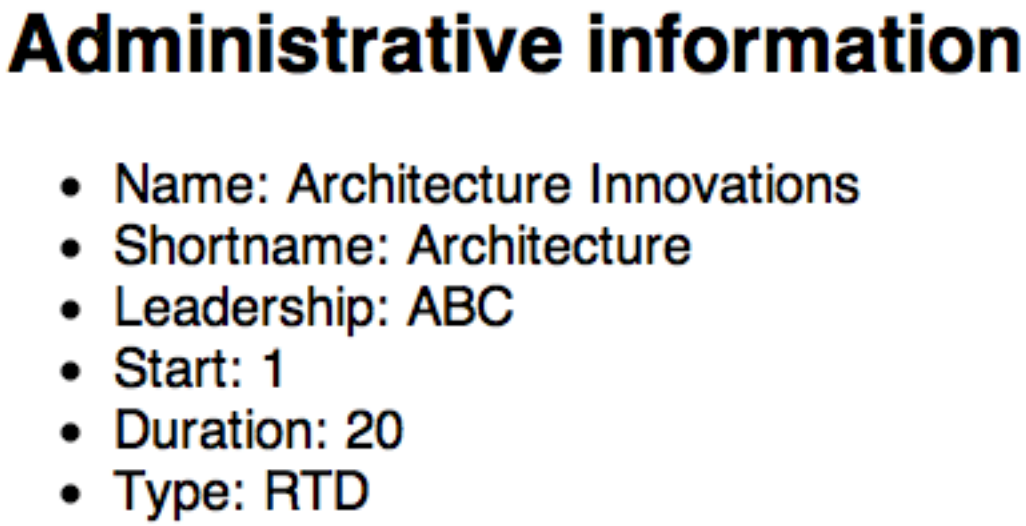


Figure 3.5: A bullet list of key/value pairs to provide adminstrative information about the workpackage.

The keys are as follows:

Name Full name of the workpackage, as shown in tables and titles.

Shortname Abbreviatiation, used in spaced-limited situations.

Leadership Shortname of the partner organization which leads this workpackage.

Start Month on which the workpackage starts (on the first day of this month); only put the number there, no "M1" or similar.

Duration Number of months the WP lasts. A WP starting in month 5 and last for 2 months, for example, will start on the first day of the fifth month of the project and ends on the last day of the *sixth* month.

Type Type designation of the WP according to the EU commission classification (e.g., RTD: Research and Development, MGMT: Management, DEMO: Demonstration).

3.3.4 Adminstrative information about Tasks

The *level-3 heading* "Tasks" (note the different levels for the "Tasks" headlines!) announces a table, immediately after the Tasks heading. It specifies the following information in columns:

Label The symbolic label of the task (the same as used above)

Start Month of the start of the task. It is assumed to start on the first day of the month.

Duration How long does the task last, in months? For example, a task with Start month 1 and Duration 1 month starts on the very first day of the project and ends at the end of the first month.

Name The full name of the task, e.g., used in tables and Gantt charts

Lead partner Shortname of the partner leading the particular task.

In general, there is a single row per task. A generalization is a task that consists of multiple *phases*: It is the task, but is interrupted and resumed. This can be realized by using one row per phase and using the *same* task Label in each row. In the second, third, ... row of a task, only the Start and Duration fields are considered. (It is currently not possible to specify different names or task leaders for different phases of a task; these should be treated as

Tasks

Label	Start	Duration	Name	Lead partner
architectureDesign	1	3	Architecture Design	UE
archImprovement	4	2	Improvement of the Architecture	ABC
archImprovement	8	2	Improvement of the Architecture	ABC
archImprovement	12	2	Improvement of the Architecture	ABC
archFalsify	10	10	Falsification of the proposed Architecture	ISC

Figure 3.6: A table listing all the tasks for the particular workpackage.

separate tasks!) Such a multi-phase task shows up in the Gantt charts; see the example of the "Improvment of the Architecture" task.

In this as in all following tables, the first row *MUST* be present and have the right labels. The labels are used to find out which column contains which information; missing labels and incorrectly formatted labels will result in errors.

3.3.5 Task efforts

Next, there is a *level-3* heading Effort. It is also followed by table with the following structure. The first colum - Partner - contains partner shortnames for partners active in this task. The following columns have one column per task in this workpackage (multiple phases are not mentioned separately).

Effort

Partner	architectureDesign	archImprovement	archFalsify
ABC	10	5	0
UE	3	7	3
ISC	0	0	8

Figure 3.7: Table to specify effort of partners in each task

Each row shows the participation of a partner in the corresponding tasks. Put person months in a field. No floating point numbers can be used, only integer numbers of months.

Partners not participating in a WP at all need not be mentioned.

3.3.6 Milestones

The *level-3 heading* Milestones announces the milestone table for the workpackage. Its structure follows the guidelines of the EU proposals. Specify the following columns:

Label A symbolic label for the milestone, used to refer to it.

Month due When must the milestone be achieved? Just the number of the month, no "M" or similar.

Title A full name for the milestone.

Contributors A comma-separated list of partner shortname, showing which partners will make contributions to this milestone. In case you want to specify a lead partner for a milestone (and similarly for a deliverable, see below), simply set it in boldface (using the Wiki's specifc markup syntax).

Description A brief description text. (Leave empty if not desired.) It is (typically) typeset in the milestone list per workpackage.

Producing task(s) Which tasks contribute to this milestone? Put a comma-separated list of *task labels* here. Task labels can also come from different workpackages; not restricted to the current one. Producing tasks can be marked as boldface (similar to contributors); however, this

Verification means Explain how the project will verify that it has achieved the milestone. (Required by standard commission template.)

Obviously, one row per milestone.

Milestones

Label	Month due	Title	Contributors	Description	Producing task(s)	Verification means
archDescrFormat	2	Architcture description format chosen	**ABC**	To ensure a precise communication of architectures, this milestone will select the proper formal description technique	architectureDesign, otherTask	Formal technique communicated to all partners

Figure 3.8: Table specifying milestones

3.3.7 Deliverables

Finally, the last *level-3 heading* Deliverables announces the table listing all the deliverables of this workpackage. It has a similar structure to the milestone table. The only difference is that "Verification means" is replaced by "Nature" (report, demonstration, software, ... -checkcommissiontemplate) and "Dissemination" (restricted RE, public PU, ... - check commission template for details).

3.4 Recognized Wiki markup

Only a limited set of Wiki markup syntax is actually recognized and translated into LaTeX commands. The following list summarizes the recognized features (see Section *wikiParser* for details):

- Headings are recognized. A level 1 heading is turned into a section, level 2 headings become subsections, etc.
- Bullet lists and enumeration lists are recognized and turned into itemize and enumerate environments. More precisely, compactitem and compactenum environments are used for compacter layouting. These

Deliverables

Label	Month due	Title	Contributors	Description	Producing task(s)	Nature	Dissemination
initialArch	3	Initial architecture description	UE, **ABC**	The initial version of the architecture is described in the chosen formal description technique	architectureDesign, plan	R	PU
improvArch	9	Improved architecture description	**UE**, ABC	Based on experiences with the initial architecture, an updated and improved version is described.	archImprovement	R	PU

Figure 3.9: Table specifying deliverables

- There is some support for figure inclusion. The basic idea is recognize a figure tag of the form <img key=value, key=value>. The following keys are recognized:

  file

  Its value is assumed to give the filename of a PDF file in the latex/figure directory.

  label

  Value will become the label in the LaTeX figure environment, for cross-referencing.

  caption

  The caption text to be used in the figure environment

  latexwidth

  The value is used to scale the figure proportionally to textwidth. It should be a value between 0 and 1. The default, if this key is not present, is to scale the figure to 0.8 of textwidth.

  Obviously, this is inspired by the twiki way of doing figures. In fact, the key/value pairs can be included in the twiki commands for figure inclusion. Moinmoin is much less amenable to extending its own syntax. It is indeed a bit cumbersome to include a figure in moinmoin, necessitating effectively two include commands: one to have the figure appear on the wiki, one to refer to the PDF file and trigger the generation of the LaTeX figure environment. Any goods ideas here are much appreciated.

- Tables are turned into tabular commands. Columns are equally wide, spaced to 80% of the textwidth. To fine-tuning the looks of these columns, you can specify the column layout: Write e.g.

  ## TABULAR: c|p{0.2textwidth}|r #

  on a separate line, immediately before the actual table in the wiki markup. Anything between TABULAR: and the closing # will be turned into the argument of the tabular environment in the LaTeX markup. No table environment is generated, if you want that, you can simply enclose the Wiki table with the corresponding LaTeX commands, they are passed through unhindered.

- Some attempts are made to maintain special characters. In particular:
  - Boldface markup is recognized

- Italics markup as well
- Line breaks specified by
 turn into newline commands
- Commands of the form: #TODO: some text # are turned into fxwarning commands of the fixme package (and appear in warnings.tex)
- Some attempts are made to provide correct left and right quotation marks; guessing is based on spaces before or after a quotation mark ".
- Pure hash marks # are protected by turning them in #.
- So are ampersand marks &.

3.5 Building a PDF file

When everything is setup correctly, doing an actual build is trivial: On a command line, go into the main project directory and type make!

CHAPTER

FOUR

HOW TO CUSTOMIZE PROPGREN

This section explains how the resulting PDF file can be customized.

4.1 Simple customization

Simple customizations can be done in settings.cfg; section *settings.cfg* gives a detailed description of all the available values.

In particular, the section CustomLaTeX gives a lot of flexibility: you can put python expressions there, evaluating on the main variables describing the project, and generate LaTeX code directly from it. Such expressions are then turned into LaTeX commands, directly accessible by writing them down in Wiki text.

4.2 Customize LaTeX templates

More powerful customization is available via the LaTeX templates in the file latexTemplate.cfg. Essentially, this give a small templating engine where LaTeX code can be put into variables, and the LaTex code can contain variable names to be replaced before the actual LaTeX is generated.

Almost all of the generated LaTeX code (ending up in generated/latex in the end) is controlled by this file. There are a few exceptions - notably, the headers of the workpackage is too complex to do via the templating engine; the files including the individual partner files and WP files are also generated automatically since they are fairly straightforward.

Details are explained in *latexTemplates.cfg*; how to use the various options in this file is described in detail in the function generateTemplates in the source code of the latexFromXML.py module *latexFromXML*.

4.3 Complex customization

For even deeper customization, you have to modify either the main.tex file (in the latex directory) or the Python scripts (in directory bin). This gives totally flexibility, but you really should understand what you are doing here.

CHAPTER

FIVE

DIRECTORY LAYOUT AND MAJOR FILES

This section summarizes the directory layout as used in the standard distribution setup. It is possible to widely reconfigure this (via the PathNames section in settings.cfg); however, unless there are concrete reasons to change it, it seems reasonable not to modify this.

5.1 Directory structure

bin All the Python scripts necessary to generate the proposal PDF are here.

doc The documentation in various formats.

docsource The sources for the documentation, in reStructuredText, based on Sphinx.

generated This directory contains all intermediately downloaded or generated files. It has several subdirectories:

latex All the generated LaTeX files go in here. Files in the directory as such are direct transliterations of the corresponding Wiki files. There are a few subdirectories:

figures All *generated* tables, gantts, and pie charts go in here (in respective subdirectories).

partners Files pertaining to the description of partners.

wp Files pertaining to individual workpackages. One file per workpackage, containing all relevant information.

wiki The raw download of the wiki sources. Very useful for error checking, in case some of the download fails, problems with special characters (e.g., Umlaute) appear, etc. Files in here should be verbatim copies of all the Wiki pages pertaining to the project. In particular, there are two further subdirectories:

partners All the partner description files, as linked from the Partner Description part of the main project page.

wp All the workpackage wiki pages, as linked from the workpackage description part of the main page.

xml A directory containing all intermediately generated XML files. Workpackage-specific information goes into the subdirectory wp. One file per workpackage. All the partner descriptions go in one LaTeX file.

The purpose of the intermediate XML representation is to give secondary tools a standard way to hook in (and it is a legacy of an older version of the program).

latex This directory contains all the manually added LaTeX files, as well as the root main.tex file invoked by pdflatex. Feel free to add any files you like in here.

It has a few subdirectories:

figures By convention, all figures necessary for the proposal go in here. In addition, there are subdirectories gantts, pies, tables pointing to generated material.

partners Same purpose as above.

styles All specific, non-standard, unusual or modified style files are collected here.

wp Same purpose as above.

In addition, there are symbolic links pointing from this directory to the directory generated, to each of the automatically generated files. This is done by the script ensureSymbolicLinks.py. The idea is to keep files on the Wiki as long as possible. But if, at some point in time, a file should be maintained only manually, remove the symbolic link, copy the generated file to the this directory, and the generation scripts will no longer overwrite this file; all manual changes are preserved. (Obviously, you should make it clear on the wiki that the wiki version is out of date.)

moin A distribution of the MoinMoin wiki, adapted to the needs of a project proposal. In particular, restricted login and a pre-configured superuser.

template All template files. It contains, for each supported wiki type, an example setup of the example project, separated by subdirectory. Currently, moinmoin and twiki. See also below for latexTemplates.cfg.

5.2 Relevant files

settings.cfg This file contains main configuration options: where to find the Wiki, which information to include, some basic settings about the look of the PDF file. See *settings.cfg* for the actual source code and detailed documentation.

template/latexTemplates.cfg In here, all the templates used to produce actual LaTeX code are maintained. Changes here allow a fine-grained customization of the result. See *latexTemplates.cfg* for the actual source code and detailed documentation.

latex/main.tex The main LaTeX file; from here, all other LaTeX files are included. You can change a lot of the behavior here, e.g., include other style files. This file is never manipulated automatically, feel free to make manual adjustments here. See *main.tex* for details.

latex/warnings.tex All warning generated during the production of XML or LaTeX code are collected here and can be typeset directly. The showWarnings flag in settings.cfg controls whether these warnings are included in the PDF output.

latex/settings.tex This file is produced from settings.cfg by turning all the True/False flags in this file into a corresponding LaTeX variable (which can be queried via the ifthenelse command). It also contains the *results* of executing all the commands in the CustomLaTeX section of settings.cfg, made accessible via a corresponding LaTeX variable.

latex/partners/partnersIncluder.tex A small file, generated to make sure that all downloaded partner descriptions are included. No need to change anything manually when a new partner is added.

latex/wp/wpIncluder.tex Similarly, a file to include all workpackage description files.

Makefile A standard Makefile guiding the generation process. It is fairly straightforward, simply running the consecutive steps of a build process after each other for the main target *pdf*. See *Makefile* for details.

projectname.pdf The ultimate PDF file. "projectname" is replaced by the value you gave in settings.cfg's projectName option. In the example setup, it will be called TestProject.pdf.

CHAPTER

SIX

OPEN ISSUES

6.1 Known bugs

None, of course :-). If you find any, let me know!

6.2 Things still to do (TODO)

Nothing immediately obvious.

6.3 Debatable aspects

1. The settings and the latexTemplates files could be put on the Wiki as well. Two-edged sword: Might make it easier for everybody to configure things, but that is a serious downside as well. Technically, this would not be difficult to do. Not made up my mind yet.

6.4 Ideas for future features

1. Integrate a version control system like SVN for the produced LaTeX files
2. Generate PDF files directly from the Wiki, make it possible to trigger that at least.
3. Integrate Etherpad into Wiki
4. Build a bridge to the financial planning of a project.

- Either by parsing from/ writing to an Excel (or similar) spreadsheet. Relatively easy, but hard to make this general
- Or by putting spreadsheet-functionality onto the wiki. Hard to do for different wiki types (a nightmare, probably).

5. Build support for latexdiff. Possibly triggered from wiki as well?
6. Better support for figures in both wiki in latex. One idea might be to upload a PDF to the wiki and have the Wiki convert it to a PNG file. And the pull the PDF file directly from the wiki, without need to manually put it in the LaTeX figure directory.

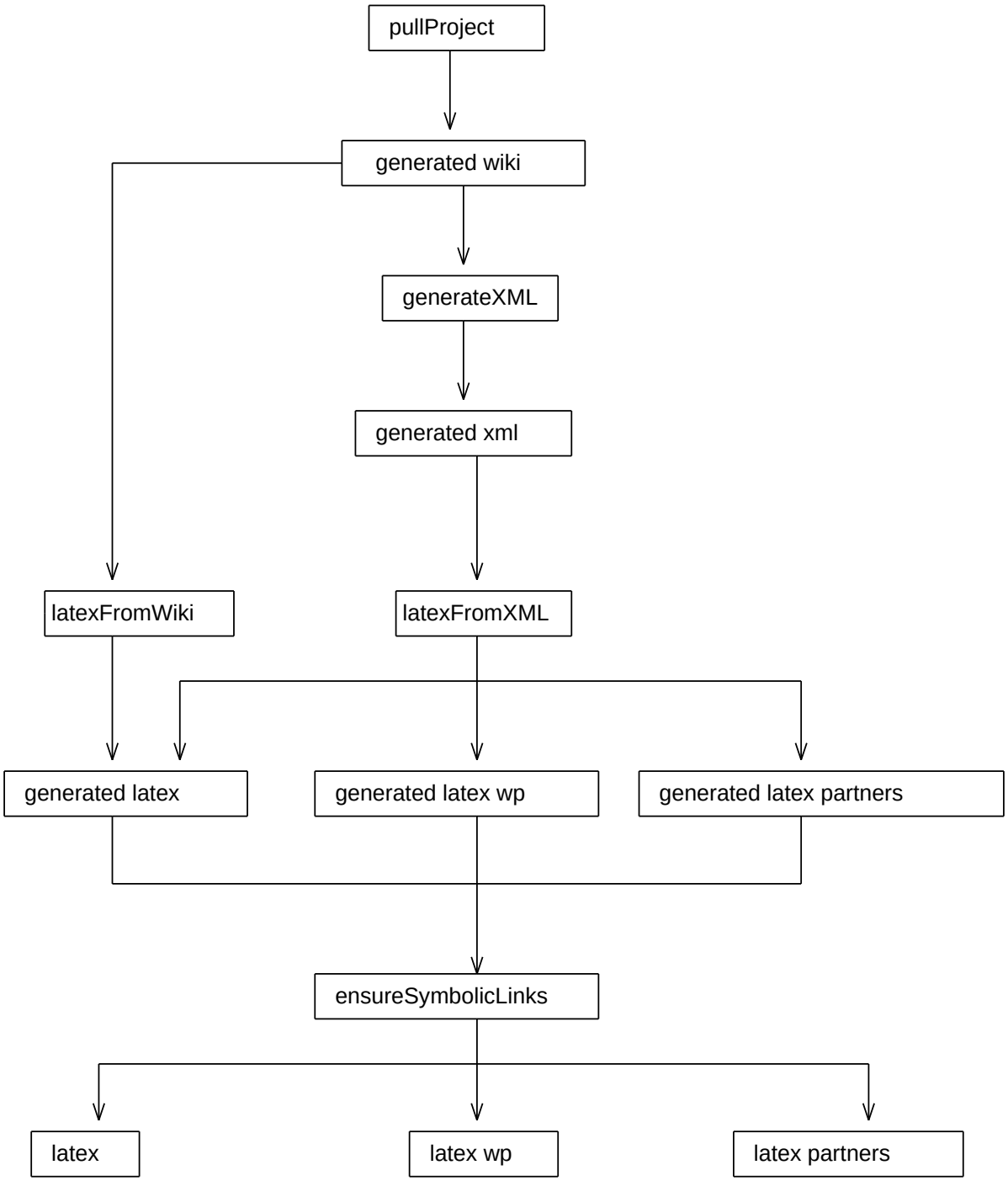
CHAPTER

SEVEN

SOURCE CODE DOCUMENTATION

The code described here lists in the bin directory. Some general remarks:

- The invocation sequence is pullproject -> generateXML -> latexFromWiki -> latexFromXML -> ensureSymbolicLinks
- Details are in the makefile in the main directory
- Many functions get passed a parameter "config". This is the content of settings.cfg, as parsed by the standard python configuration file parser ConfigParser.SafeConfigParser (see python library documentation for details).



7.1 pullproject

Pull the raw wiki files from wherever is specified in settings.cfg. Store the raw wiki syntax in the wikipath directory.

`pullProject.`**`ensureDirectories`**(*config*)
A small helper function that makes sure that all the directories that are mentioned in settings.cfg PathNames section actually exist. This can be useful after a make clean or in case directories have been manually and inadvertently removed.

`pullProject.`**`getPartners`**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
get all the partner description files

pullProject.**getProposalStructure**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
Extract all the relevant files for the actual proposal text from the wiki.

pullProject.**getWorkpackages**(*masterPage*, *pullInstance*, *config*, *parser*, *verbose=False*)
Identify all the workpackages and download them

7.2 wikiParser

7.2.1 The wikiParser module as such

We need to parse various wiki formats into useable latex. This module provides an abstract base class wikiParser that implements a lot of basic functions e.g., to extract tables, lists, etc.

This base class has to be subclassed to specialize for specific Wiki syntax variants. The subclasses can be fairly slim and mostly specify regular expressions to use (e.g., how to recognize headings).

A factory function is called to obtain an instance of such a parser.

wikiParser.**wikiParserFactory**(*config*)
Construct an instance of the correct parser class, choice depends on what is selected in settings.cfg.

7.2.2 The wikiParser base class

class wikiParser.**wikiParser**
Base class to get the interface for turning wiki syntax into useful stuff

applyLaTeXFunctions(*latex*)
Apply all the LaTeX conversions functions step by step. Note that the order is important!

buildFigure(*t*)
An attempt to allow direct figure inclusion. See documentation for details on synatx and limitations.

buildHeadings(*latex*)
Turn all the wiki headings in the latex parameter into the proper LaTeX heading commands, along with a label command as well. Uses the classes headingReplacements attribute where proper regular expressions are defined.

buildLists(*latex*)
Turn all the Wiki lists in the latex text into proper LaTeX lists. Take care to handle nested lists correctly.

buildTable(*t*)
Turn wiki tables into LaTeX tabular environments. Interpret #TABULAR commands to set the tabular header.

constructLabel(*t*)
Given a heading, construct a suitable label out of it. Remove whitspaces and obvious strange characters.

getLaTeX(*t*)
turn all of the wiki into LaTeX

getList(*wiki*)
Turn the first itemize in the wiki into a list.

Note: this seems to be identical for all known wiki syntax variations overwrite this in subclass if necessary

getListAsDict(*wiki*, *delimiter=':'*)
Take the next enumeration from the wiki content. Assume it is a list with key/value delimited by delimiter. Split them up, return a proper dictionary for that

getSection (*wiki*, *title*, *level*)
Extract the section with title at level from the text in the wiki parameter

getSectionRe (*wiki*, *startre*, *endre*)
Use the text in the wiki parameter, extract the next section matching the start and end regular expressions.

getTable (*wiki*)
turn the first table into list of dictionaries, using the first row as keys for the dictionaries. Remove boldfacing from the first row entries if present.

handleCharacters (*latex*)
Replace any special characters that might appear from attempts at manual HTML markup.

moveCommissionHints (*t*)
Make sure that commission hints appear after the first heading!

7.2.3 The moinmoin parser

class wikiParser.**wikiParserMoinmoin** (*config*)
Specialized for Moinmoin. Especially the order of the heading replacement regular expressions is tricky for moinmoin.

buildFigure (*t*)
For building a figure, the moinmoin syntax gets in out way - kick out the attachment syntax and rely on the twik iway of doing it - this needs fixing! TODO

getSection (*wiki*, *title*, *level*)
Extract the section with title at level

7.2.4 The twiki parser

class wikiParser.**wikiParserTwiki** (*config*)
Specialized for Twiki

getSection (*wiki*, *title*, *level*)
extract the section with title at level

localHeading (*title*, *level*)
How does a heading with the given title, at the given level, look in this wiki style? Describe it as a regular expression.

7.3 latexFromWiki

Generate LaTeX code from all the plain wiki files.

- This pertains to the plain files as well as to the partner files.
- Uses the settings file to find out relevant directories.
- XML files are not treated here; that is more complicated.

Essentially, this script just creates a WikiParser object via the libaray and applies it to the corresponding files.

latexFromWiki.**handleFile** (*f*, *outdir*, *parser*, *config*, *verbose=False*)
Translate a wiki file with name f to the corresponding LaTeX file.

Information where and how to translate are giving in config. Parser is a parser object for the correct wiki style.

Exception: if the filename contains bibtex in some capitalization, we append .bib, not .tex.

7.4 latexFromXML

(Note: key/value pairs for the main global variables in this module are described in *latexFromXML - key/value pairs used in global variables*.) This script reads in the XML files describing the project and generates the LaTeX output. To this end, it uses a couple of steps:

1. Analze the entire XML tree and put the content into the global variables allWPDicts, allMilestones, allDeliverables, allTasks, allEfforts, partnerList and titlepageDict. This is triggered by the analyzeTree function.
2. Statistics are computed - computeStatistics
3. The tables for the WP are generated - computeWPTable
4. The partner descriptions are generated - generatePartnerDescriptions
5. The templates from latexTemplates.cfg are processed; this is the main step for all the details of LaTeX producting - generateTemplates
6. Finally, LaTeX options in settings.cfg - processLaTeX

Details from where files are read and where files are put are controlled by settings.cfg.

class latexFromXML.**recursiveTemplate**(*template*)
Try to find recursively occuring patterns and replace them first

latexFromXML.**analyzePartners**(*tree*)
Produce the partnerList dictionaries, containing all partner descriptions, from the correspndng part of the XML tree.

latexFromXML.**analyzeTree**(*tree*, *config*, *verbose=False*)
Take an XML tree and create all the necessary data structures; just invokes various analyses functions.

latexFromXML.**analyzeWPs**(*tree*, *verbose=False*)
Given an XML tree pointing to a WP, extract all the information from it and build the global variables describing this WP.

latexFromXML.**computeGanttStrings**(*config*)
Determine the actual gantt chart, spearately for task bars, the deliverables, and the milestones. This function computes various extensions for the main global variables (see key documentation). Also a combined deliverables/milestones string enables easy mix and match

construct the relevant milestone, deliverable list question is whether to incorporate also the cross-WP milestones/deliverables; this is configurable option

latexFromXML.**computeStatistics**(*verbose*)
Embellish the global lists with some additional sums, statistics, cross-referencing to IDs, etc. Add whatever seems useful here, makes it later to generate the LaTeX strings later on.

latexFromXML.**computeWPTable**(*config*)
For each WP in allWPDicts, compute the LaTeX string for the header of the workpackage table. The table header is fairly complex to stitch together and is done here instead of in the latexTemplates.cfg.

latexFromXML.**dictFromXML**(*tree*)
Given an XML node (obtained via the xml.etree.ElementTree libary, build a dictionary where the keys are the tag of a child node and the text attribute of the child node is the value. Return this dictionary.

latexFromXML.**dictFromXMLWithMains**(*tree*)
Similar to dictFromXML, but this function in addition understands an XML-attribute main. This is used to differentiate, e.g., between lead contributor and non-lead contributors of a task; or to differentiate between a main producing task for a deliverable and an ordinary task. This XML attributes are generated based on boldface markup in generateXML; see function singleWorkpackageXML in generateXML.py. If such a key is found in the XML atribute, a corresponding entry is put into the dictionary that is to be generated from this XML subtree.

Example: Milestone dicts have a key Contributor, which has a list of partner shortnames. A milestone also *might* have a key ContributorMain, which is a partner shortname string, specifying a potential lead partner for this milestone.

latexFromXML.**fixProducingTask**(*ll*)
Input is a list of milestones or deliverables. We need to fix the ProducingtaskString; dictFromXMLWithMains is not specific enough for that job. Replace the task label by the task IDs and do proper sorting, keeping in mind the ordering of task ids

latexFromXML.**generatePartnerDescriptions**(*config*, *verbose*)
Generate the LaTeX code to describe a partner, including subsection heading and labels. Produces the partnersIncluder.tex file.

latexFromXML.**generateTemplates**(*config*, *verbose*)
This function processes the latexTemplates.cfg file. It goes over each section, expands the template option, considering the other options. The expansion of the template option is based on Python's string.Template expansion, with a few extras added in.

The expanded version of the template string is stored in a special dictionary expanded, using the section name as the key. It can be used like the other dictionaries; hence, complex entries can be constructed over multiple sections using the expanded dictionary as the dict from which the expansion strings are pulled. This is common practice e.g. for complex tables, where first individual lines are built and the the full table is constructued.

The following options are understood:

template

The actual template string. Write plain LaTeX code here. Add ${NAME} commands; these are expanded during processing just like the standard template class does; see below for details. In addition, it is possible to use %{ PYTHONCODE %}. This string is replaced by the evaluation result of the PYTHONCODE string.

dict Specify a dictionary in which the ${NAME} are looked up as keys; the value of that key then replaces the ${NAME} string in template. Either dict or list must be given.

list A list option must be followed by list variable (or other iterable). Then, the template is evaluated over each element of the list; the elements must be dictionaries where the ${NAME}s exist as keys.

A dict option can be given as well, then ${NAME}s are looked up both in the list elements as well as in the given dictionary.

Without any further options, a list of expanded template strings is put into expanded dictionary, under the sectionname as key.

numerator When a list option is given, it can be useful or desirable not to put a list into the expanded dictionary, but rather a separate entry under a separate key for each list element can be useful. The numerator option triggers this and specifies the suffix to be placed after the section name in the expanded dictionary. numerator can be a an expression like value['Shortname'], which is then evaluated over the dictionray in the provided list (in this example, this is evaluated over the list allWPDicts, Shortname is the WP shortname; then, sectionname-WPShortname can be used later on).

joiner If a list is given, but ony a single string entry in the expanded dictionary is desired (instead of a a list entry or a separate entries), then a string to be used with the string.join method can be given here.

sorter Specifies a Python expression to sort the list before being expaned. Typically, this should be a lambda function, evaluated over the list elements in the usual way the Python sorted built-in function's key option is used. Mostly makes sense in combination with the joiner option.

file Normally, results of the template expansion are only placed in the expanded dictionary. Giving file=True optino also writes the expansion result into a file called SECTIONNAME.tex. Directory is the genlatexpath value in settings.cfg.

dir Only relevant when file=True: this option specifies the directory (relative to the genlatexpath option).

latexFromXML.**generateTemplatesBuildListResult**(*templ*, *listtoworkon*, *keytosave*, *expandedresults*)
Helper function to deal with lists in the template substition process.

latexFromXML.**processLaTeX**(*config*)
Process the LaTeX-relevant sections of settings.cfg, turn the options therein into LaTeX commands. Some of them need specific processing (like showCommissionHints), others evaluate a LaTeX expression, others simply generate a boolean variable for the switches in settings.cfg.

latexFromXML.**produceCompressedGantt**(*l*, *config*)
Produce a Gantt chart where milestones and deliverables are packed in as few lines as possible, with horizontal separation being controlled by the corresponding setting in settings.cfg. Milestones/deliverables are sorted by their due date! If they should appear in some other order, change the first line in this function (computation of inputList). TODO: Thinky about making this a configurable option.

latexFromXML.**produceUncompressedGantt**(*l*, *config*)
produce a Gantt string for the elements in list, without trying to compress that into a complact representation. Each element a separate line.

latexFromXML.**writeTemplateResult**(*expanded*, *templ*, *keytouse=None*)
Use the information in templ to check whether it should be written out and write the particular template to disk.

latexFromXML.**allDeliverables = []**
A list storing one dictionary per deliverable. Filled in from XML file, with some additions.

latexFromXML.**allEfforts = []**
A list containing a dictionary per task/partner combination. Extracted from the effort tables on the Wiki.

latexFromXML.**allMilestones = []**
A list storing one dictionary per milestone. Filled in from XML file, with some additions.

latexFromXML.**allTasks = []**
A list storing one dictionary per task. Collected from all the tasks in all the workpackage pages. Some additions computed.

latexFromXML.**allWPDicts = []**
A list storing one dictionary per workpackage. Filled from XML file. Some additions computed.

latexFromXML.**expanded = {}**
This dictionary collects all the expansions of templates from latexTemplate.cfg. Like all the other ones, it can be used as an argument to the dict option therein, allowing to build templates that use the expansions of simpler templates as variables.

latexFromXML.**partnerList = []**
A list storing one dictionary per partner.

latexFromXML.**titlepageDict = {}**
Dictionary containing all information about the project in general; mostly it goes on the titlepage. It directly obtains its content from the main project wiki page, without any additions computed here.

7.5 latexFromXML - key/value pairs used in global variables

7.5.1 titlepageDict

call Type: <type 'str'>

Example: ICT FP7-ICT-2012-8

coordinatoremail Type: <type 'str'>

Example: peter.pan@neverland.org (peter.pan@neverland.org)

coordinatorname Type: <type 'str'>

Example: Peter Pan

coordinatorphone Type: <type 'str'>

Example: +12 345 5678

duration Type: <type 'str'>

Example: 24

instrument Type: <type 'str'>

Example: Integrated Project

projectname Type: <type 'str'>

Example: Investigating really important research topics

projectshort Type: <type 'str'>

Example: IRIR

topics Type: <type 'str'>

Example: Objective ICT-2011.1.1 Future Networks

7.5.2 allWPDicts

Duration Type: <type 'str'>

Example: 24

End Type: <type 'int'>

Example: 24

Leadernumber Type: <type 'str'>

Example: 2

Leadership Type: <type 'str'>

Example: ISC

Name Type: <type 'str'>

Example: Project Management

Number Type: <type 'str'>

Example: 1

Shortname Type: <type 'str'>

Example: Management

Start Type: <type 'str'>

Example: 1

Type Type: <type 'str'>

Example: MGMT

deliverableGanttLegend Type: <type 'str'>

Documentation: Compare the same corresponding key for milestones.

Example: \item \textbf{D\,1.1}: A research plan description \item \textbf{D\,2.1}: Initial architecture description

deliverableGanttString Type: <type 'str'>

Documentation: Compare the same corresponding key for milestones.

Example: \ganttmilestone{D\,2.1}{3} \ganttmilestone{D\,1.1}{12}

deliverableInGantt Type: <type 'list'>

Documentation: Compare the same corresponding key for milestones.

Example: ['reserachplan', 'initialArch']

deliverableUncompressedGanttString Type: <type 'str'>

Documentation: Compare the same corresponding key for milestones.

Example: \ganttmilestone{D\,1.1}{12}\\ \ganttmilestone{D\,2.1}{3}

groupbar Type: <type 'str'>

Documentation: In a horizontal bar is desired to separate WPs; this is a command for the pgfgantt package.

Example: \ganttgroup{}{1}{24} \\

milestoneGanttLegend Type: <type 'str'>

Documentation: The part of the Gantt legend pertaining to the milestones. Its look is controlled by the milestoneLegendTemplate in settings.cfg.

Example: \item \textbf{M\,1.1}: Website goes public \item \textbf{M\,2.1}: Architcture description format chosen

milestoneGanttString Type: <type 'str'>

Documentation: A LaTeX command string containing the commands to typeset the milestones of a particular WP (use pgfgantt commands). It is in compressed version, i.e., it tries to put milestones on as few lines as possible.

Example: \ganttmilestone[milestone={fill=orange, rounded corners=5pt}]{M\,2.1}{2} \ganttmilestone[milestone={fill=orange, rounded corners=5pt}]{M\,1.1}{7}

milestoneInGantt Type: <type 'list'>

Documentation: Which milestones (symbolic labels) appear in the Gantt chart of this WP? (This is NOT the same thing as the milestones hosted in a WP because of cross-WP milestones; this list might contain milestones of other WPs as well in case a task of this WP contributes to the milestone.

Example: ['website', 'archDescrFormat']

milestoneUncompressedGanttString Type: <type 'str'>

Documentation: A LaTeX command string containing the commands to typeset the milestones of a particular WP. This is in uncompressed form, i.e., each miilestone goes on a separate line.

Example: \ganttmilestone[milestone={fill=orange, rounded corners=5pt}]{M\,1.1}{7}\\ \ganttmilestone[milestone={fill=orange, rounded corners=5pt}]{M\,2.1}{2}

objectives Type: <type 'unicode'>

Example: The objectives of this workpage are Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla urna. Maecenas interdum nunc in augue. Mauris quis massa in ante tincidunt mollis. Proin imperdiet. Donec porttitor pede id est. Sed in ante. Integer id arcu. Nam lectus nisl, posuere sit amet, imperdiet ut, tristique ac, lorem. In erat. In commodo enim. Phasellus libero ipsum, tempor a, pharetra consequat, pellentesque sit amet, sem. Praesent ut augue luctus elit adipiscing ultricies. Vestibulum suscipit cursus leo. Nullam molestie justo.

Umlaut test: äöü é è ß

partnereffort Type: <type 'dict'>

Documentation: A dictionary mapping the shortname of each partner with positive effort in this WP to the effort (as integer).

Example: {'UE': 10, 'ISC': 0, 'ABC': 15}

tableheader **Type**: <type 'str'>

Documentation: The WP table header block, with effort lines per partner. It is a complete tabular environment, ready to be used in a template section of latexTemplates.cfg.

Example: \begin{tabular}{|p{0.15\\textwidth}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|@{\\hskip 0ex}p{0.100937\\textwidth}@{\\hskip 0ex}|} \hline\textbf{Workpackage no.} & \multicolumn{1}{c|}{1} & \multicolumn{7}{l|}{\textbf{Start date:} M 1} \\ \hline \textbf{Title} & \multicolumn{8}{l|}{Project Management} \\ \hline\textbf{Activity type} & \multicolumn{8}{l|}{MGMT \hfill} \\ \hline Part.\ no.\ & \centering{1} & \centering{\textbf{\textcolor{gray}{2}}} & \centering{3} & & & & & \\ \hline Short name & \centering{ABC} & \centering{\textbf{\textcolor{gray}{ISC}}} & \centering{UE} & & & & & \\ \hline Effort & \centering{15} & \centering{\textbf{\textcolor{gray}{0}}} & \centering{10} & & & & & \\ \hline \end{tabular}

taskGantt **Type**: <type 'str'>

Documentation: A command for pgfgantt to set the task part of the WP's Gantt chart. This is in principle straightforward, but the requirement for multi-phase tasks makes it a bit complicated.

Example: \ganttbar[name=T1-1-0]{T\,1.1: Research preparation}{1}{24} \\ \ganttbar[name=T1-2-0]{T\,1.2: Research execution}{5}{7} \\

taskeffort **Type**: <type 'dict'>

Documentation: A dictionary, mapping the symbolic label of each task of this WP to the total effort it consumes (as integer).

Example: {'otherTask': 12, 'plan': 13}

wpdescription **Type**: <type 'str'>

Example: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla urna. Maecenas interdum nunc in augue. Mauris quis massa in ante tincidunt mollis. Proin imperdiet. Donec porttitor pede id est. Sed in ante. Integer id arcu. Nam lectus nisl, posuere sit amet, imperdiet ut, tristique ac, lorem. In erat. In commodo enim. Phasellus libero ipsum, tempor a, pharetra consequat, pellentesque sit amet, sem. Praesent ut augue luctus elit adipiscing ultricies. Vestibulum suscipit cursus leo. Nullam molestie justo.

wpeffort **Type**: <type 'str'>

Documentation: The total effort of this WP (as string, not sure why?).

Example: 25

7.5.3 allTasks

Duration **Type**: <type 'int'>

Example: 24

Label **Type**: <type 'str'>

Example: plan

Leadpartner **Type**: <type 'str'>

Example: UE

Main **Type**: <type 'str'>

Example: True

Name Type: <type 'str'>

Example: Research preparation

Start Type: <type 'int'>

Example: 1

contributedDeliverables Type: <type 'list'>

Documentation: All the deliverables this task contributes to, using the label of the deliverable. A list; can be turned into a string by proper join operation.

Example: ['reserachplan', 'initialArch']

ganttid Type: <type 'str'>

Documentation: The string to typeset in the Gantt box of this task.

Example: T1-1

taskId Type: <type 'str'>

Documentation: Based on the tasknumber, construct a readbable number for this task.

Example: T\,1.1

taskdescription Type: <type 'str'>

Example: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla urna. Maecenas interdum nunc in augue. Mauris quis massa in ante tincidunt mollis. Proin imperdiet. Donec porttitor pede id est. Sed in ante. Integer id arcu. Nam lectus nisl, posuere sit amet, imperdiet ut, tristique ac, lorem. In erat. In commodo enim. Phasellus libero ipsum, tempor a, pharetra consequat, pellentesque sit amet, sem. Praesent ut augue luctus elit adipiscing ultricies. Vestibulum suscipit cursus leo. Nullam molestie justo.

Morbi dui. Morbi convallis mi sed sem. Nulla convallis lacus vitae risus. Phasellus adipiscing. Nullam tortor. Sed laoreet aliquam ante. Vestibulum diam. Pellentesque nec leo. Pellentesque velit. Praesent congue mi eu ipsum cursus fringilla. Etiam leo erat, tristique et, pharetra eget, mollis vitae, velit. In hac habitasse platea dictumst. In quam nibh, facilisis et, laoreet non, facilisis tempus, justo.

Donec nulla lectus, faucibus sit amet, auctor non, consectetuer quis, pede. Nullam dictum. Nullam suscipit, ligula in scelerisque posuere, sapien purus rutrum magna, vitae pharetra leo quam vel tortor. Donec eleifend condimentum sapien. Etiam sed orci. Aliquam tempor. Pellentesque egestas tortor id eros. Donec mauris justo, commodo id, pellentesque id, eleifend non, mi. Duis venenatis sagittis metus.

tasknumber Type: <type 'int'>

Documentation: Constructed tasknumber, makes sure that a multiple phase task only gets one number, consequetively increasing in a WP, ordered in the same order as the tasks appear on the WP's wiki page. If you want something like T 1.1, use taskId instead.

Example: 1

taskobjectives Type: <type 'str'>

Example: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla urna. Maecenas interdum nunc in augue. Mauris quis massa in ante tincidunt mollis. Proin imperdiet. Donec porttitor pede id est. Sed in ante. Integer id arcu. Nam lectus nisl, posuere sit amet, imperdiet ut, tristique ac, lorem. In erat. In commodo enim. Phasellus libero ipsum, tempor a, pharetra consequat, pellentesque sit amet, sem. Praesent ut augue luctus elit adipiscing ultricies. Vestibulum suscipit cursus leo. Nullam molestie justo.

\begin{compactitem} \item Objective one for a task \item Objective two for a task \item Objective three for a task \end{compactitem}

wp Type: <type 'str'>

Documentation: For simplicity, this field describes the number of the workpackage in which this task is hosted. Not strictly necessary, but makes a number of tests simpler later on.

Example: 1

7.5.4 allMilestones

Contributor Type: <type 'list'>

Example: ['ABC']

ContributorMain Type: <type 'str'>

Documentation: A main key for the acutal key Contributor

Example: ABC

ContributorString Type: <type 'str'>

Documentation: Since there is a main entry for key Contributor, we add here a key to the dictionary that contains a string concatening the list of individual entries, marking the main entry in boldface.

Example: \textbf{ABC}

Description Type: <type 'str'>

Example: Website for the project is ready

Label Type: <type 'str'>

Example: website

Monthdue Type: <type 'int'>

Documentation: Due dates are interpreted as being at the END of the given month. Relevant for correct placement of the markers in the Gantt charts.

Example: 7

Producingtask Type: <type 'list'>

Example: ['otherTask']

ProducingtaskString Type: <type 'str'>

Documentation: A string that contains all the tasks producing this milestone or deliverable; with a possible main contributor set in boldface.

Example: T\,1.2

Title Type: <type 'str'>

Example: Website goes public

Verificationmeans Type: <type 'str'>

Example: Website can be accessed

deco Type: <type 'str'>

Documentation: A string to be passed to the pgfgantt package, to make the milestones look differently from the deliverables markers. Controlled by the milestoneDecoration option in settings.cfg.

Example: [milestone={fill=orange, rounded corners=5pt}]

ganttLegend Type: <type 'str'>

Documentation: The string to be put into the legend of a Gantt chart for this milestone. Controlled by the deliverableLegendTemplate option in settings.cfg.

Example: \item \textbf{M\,1.1}: Website goes public

id Type: <type 'str'>

Documentation: Unique shortname for the milestone.

Example: M\,1.1

wp Type: <type 'str'>

Documentation: For simplicity, this field describes the number of the workpackage in which this task is hosted. Not strictly necessary, but makes a number of tests simpler later on.

Example: 1

7.5.5 allDeliverables

Contributor Type: <type 'list'>

Example: ['UE', 'ABC', 'ISC']

ContributorMain Type: <type 'str'>

Documentation: A main key for the acutal key Contributor

Example: ISC

ContributorString Type: <type 'str'>

Documentation: Since there is a main entry for key Contributor, we add here a key to the dictionary that contains a string concatening the list of individual entries, marking the main entry in boldface.

Example: ABC, \textbf{ISC}, UE

Description Type: <type 'str'>

Example: A complete research plan to be followed for the remained of the project is described.

Dissemination Type: <type 'str'>

Example: PU

Label Type: <type 'str'>

Example: reserachplan

Monthdue Type: <type 'int'>

Documentation: Due dates are interpreted as being at the END of the given month. Relevant for correct placement of the markers in the Gantt charts.

Example: 12

Nature Type: <type 'str'>

Example: R

Producingtask Type: <type 'list'>

Example: ['plan', 'otherTask']

ProducingtaskMain Type: <type 'str'>

Documentation: A main key for the acutal key Producingtask

Example: otherTask

ProducingtaskString Type: <type 'str'>

Documentation: A string that contains all the tasks producing this milestone or deliverable; with a possible main contributor set in boldface.

Example: T\,1.1, \textbf{T\,1.2}

Title Type: <type 'str'>

Example: A research plan description

ganttLegend Type: <type 'str'>

Documentation: The string to be put into the legend of a Gantt chart for this milestone. Controlled by the deliverableLegendTemplate option in settings.cfg.

Example: \item \textbf{D\,1.1}: A research plan description

id Type: <type 'str'>

Documentation: Unique shortname for the milestone.

Example: D\,1.1

wp Type: <type 'str'>

Documentation: For simplicity, this field describes the number of the workpackage in which this task is hosted. Not strictly necessary, but makes a number of tests simpler later on.

Example: 1

7.5.6 allEfforts

partner Type: <type 'str'>

Documentation: The partner shortname of the partner organization the effort of which is described here.

Example: ABC

resources Type: <type 'str'>

Documentation: The resources which this partner has in this task.

Example: 5

task Type: <type 'str'>

Documentation: The task label which identifies this task.

Example: otherTask

wp Type: <type 'str'>

Documentation: For simplicity, this field describes the number of the workpackage in which this task is hosted. Not strictly necessary, but makes a number of tests simpler later on.

Example: 1

7.5.7 partnerList

Name Type: <type 'str'>

Example: A Broad Company

Nation Type: <type 'str'>

Example: FR

Number Type: <type 'str'>

Example: 1

Shortname Type: <type 'str'>

Example: ABC

Type Type: <type 'str'>

Example: IN

Wiki Type: <type 'str'>

Example: PartnerAbc

7.6 ensureSymbolicLinks

Make sure that the symbolic links from the manual to the generated subtree exist, unless there is already a real file there.

ensureSymbolicLinks.**createLinks**(*config*)
Look into config, in the Paths section, for any directory with generated as prefix, and has a corresponding manual directory as peer. Put symbolic links there if necessary to all files in generated.

7.7 utils

A module with assorted helper and ultility functions and classes.

class utils.**documentedDict**(**args*, ***kwargs*)
A slightly extended dict class. It has a property keydoc. Assigning a string to this property stroes a keydoc for the most recently assigned key in a class dictionary. Reading this property returns a dictionary of these keydocs.

utils.**deepExpandInclude**(*prefix*, *infile*)
Recursively expaned #include commands in XML files. Make sure UTF-8 is used for writing.

utils.**dictAsRest**(*d*, *fp*)
Given a dictionary, print key/values as a description list in reStructuredText syntax. Checks if the dictionary is a documentedDict and then pulls the corresponding keydoc and prints it as documentation of the key/value example.

utils.**docuDict**(*s*, *d*, *fp*)
For documentation purposes: print a rest header using s, and then print the dictionary as a description list via dictAsRest().

utils.**expandInclude**(*prefix*, *infile*, *outfile*)
Expand an xML file, make sure UTF-8 is used for reading.

utils.**getSettings**(*filename*)
Try to find the settings file, turn it into a configParser object, and do some first preprocessing on it.

utils.**mapReduce**(*l*, *reducefct*)
Perform a map reduce operation on a list of (key, value) pairs. Apply the reducefct in the reduce step.

This is mostly used for building pie charts but might be generally useful.

utils.**roundPie**(*l*)
round the values to 100%, input: list of (name, value) tuples

utils.**searchListOfDicts**(*l*, *key*, *value*, *returnkey*)
Search a list l which contains dictionaries for an entry where key has value, and return the value of returnkey.

utils.**treeReduce**(*l*, *reducefct*)
Recursively apply a reduce function to a nested list structure. Atomic elements must be boolean values.

utils.**warning**(*w*)
Append the warning text w to warnings.tex

utils.**writefile**(*t*, *f*)
Write text t into file f. Main point: use utf-8 encoding for output. Just a shorthand. And empty t parameter causes the file to be overwritten with an empty file!

7.8 settings.cfg

7.8.1 Wiki

See LICENCE file for licencing information! Set up the necessary information to access the wiki: which type, where can it be found, what is the start page, which account and password to use to log in.

projectName
This option specificies the root Wiki page where the main project information can be found. It also serves as the filename of the final pdf file.

Default: TestProject

wikitype
The wikitype setting selects which type of wiki access is to be used. Wikitypes currently supported are: twiki, moinmoin and moinmoin-local

Option 1: moinmoin-local This option specifies that the a moinmoin wiki can be accessed in the same file system where the generation system executes. This option requires to sepcify the moinmoinpath option as well. No user and password need to be given, but the files must be accessible for reading.

Option 2: moinmoin

A moinmoin wiki is used and accessed remotely, using the mechanize library. To this end, both the wikiuser and wikipassword option need to be specified; they are used for login. Also, the baseURL option needs to be specified: it provides a URL where the wiki can be accessed, without any concrete page name. Example:

baseURL = http://hk-vm.cs.uni-paderborn.de:8080/

Option 3: twiki

It needs the same options as moinmoin as well as the loginURL setting. The difference is that the obtained files are parsed assuming the twiki syntax and the login process to a Twiki is slightly different from a moinmoin wiki. The distribution's configuration file example assumes a moinmoin-local

Default: moinmoin-local

moinmoinpath
Specify the moinmoin path in the standard PropGen distribution. This option is only relevant if wikitype = moinmoin-local and can be left out for other wiki types.

Default: ../moin/

baseURL
For remote access to a wiki. It specifies the "root" URL where the wiki can be accessed, without any particular page name. It is ignored for the moinmoin-local wikitype and only used for other wikitypes.

Default: http://hk-vm.cs.uni-paderborn.de:8080/

wikiuser
The wikiuser account name to use to log in. Ignored in the moinmoin-local wikitype. The default corresponds to the account name preset in the distribution's moinmoin wiki. Change this option to reflect your own user name.

Default: ProjectMaster

wikipassword
Password used to log in. The default is the one used in the distribution's example moinmoin wiki. You definitely want to change the password on the wiki and then reflect this change here. It is ignored in the moinmoin-local wikitype and only needed for remote access.

Default: 123abc

loginURL
Some remote wikis usually need a special login URL, e.g., Twiki wikis. Specify here. This setting is ignored in both the moinmoin-local wikitype (where no login is needed at all) and in the moinmoin wikitype (were the login URL is constructed directly from the baseURL and no separate URL is needed).

Default: https://twiki.sics.se/bin/login

httpProxyIP
For remote Wiki access through a proxy: These variables are used by the mechanize module, for access via an HTTP or HTTPS proxy. You can specify the proxy's IP, port number, and user and password, if needed. You can do that separately for HTTP and HTTPS access. But this does not always works out well. Also, this functionality is not well tested. Your mileage WILL vary! All the defaults are just empty.

Default:

httpProxyport
Default:

httpProxyuser
Default:

httpProxypassword
Default:

httpsProxyIP
Default:

httpsProxyport
Default:

httpsProxyuser
Default:

httpsProxypassword
Default:

7.8.2 PathNames

The section PathNames gives fine-grained controled over the dictionaries where intermediate files are stored. Caution: best not to touch these paths!!! You should know what you are doing here! Note: these paths are relative to the main directory

binpath
Where are all the scripts for the actual generation?

Default: bin

wikipath
Where should downloaded wiki files be stored?

Default: generated/wiki

wikiwppath
Default: generated/wiki/wp

wikipartnerpath
Default: generated/wiki/partners

xmlpath
Where do generated xml files go?

Default: generated/xml

xmlwppath
Default: generated/xml/wp

latexTemplates
Where is the LaTeX templates file?

Default: template/latexTemplates.cfg

genlatexpath
Where do generated LaTeX files, wp paths, partner files go?

Default: generated/latex

genlatexfigurespath
Default: generated/latex/figures

genlatexganttspath
Default: generated/latex/gantts

genlatextablespath
Default: generated/latex/tables

genlatexpiespath
Default: generated/latex/pies

genlatexwppath
Default: generated/latex/wp

genlatexpartnerspath
Default: generated/latex/partners

manuallatexpath
Where are the MANUAL latex files? Usage: in that directory, no files are EVER overwritten (as opposed to the genlatex directory tree, where all files are routinely overwritten). However, it is ensured that for all generated files, there is either a regular file of the same name in the manual directory (then nothing happens), or a symbolic link is created in the manual directory. NOTE: no good idea how to replicate that behavior in Windows

Default: latex

manuallatexfigurespath
Default: latex/figures

manuallatexganttspath
Default: latex/figures/gantts

manuallatextablespath
Default: latex/figures/tables

manuallatexpiespath
Default: latex/figures/pies

manuallatexwppath
Default: latex/wp

manuallatexpartnerspath
Default: latex/partners

7.8.3 Gantts

Control which and how Gantt charts are typeset. This section has a few True/False toggles to include/exclude some features in the charts. It also has a few parameters detailing layout options. For fine-grained control, see the latexTemplates.cfg file; in particular, the GanttPrefix section. Also, the true/false settings here can be easily used (via the LaTeX ifthenelse command) to control behavior in latexTemplates.cfg.

WpMilestonesUncompressedShow
Some simple examples - generate various WP-specific Gantt charts for milestones, deliverables.

Default: True

WpMilestonesShow
Default: True

WpDeliverablesUncompressedShow
Default: True

ShowWPBar
For the full-project milestones/deliverable gantt charts: build a WP bar to separate WPs ?

Default: True

ganttPerWPShowsLegend
should the WP-specifc Gantt charts show a legend?

Default: True

ganttLegendTwoColumn
Should a Gantt legend be one or two columns? (A one-column legend will consume a lot of space)

Default: True

ganttTaskbarsShowTaskname
Task bars show task names in the gantt chart? If True, the Gantt chart is likely easier to read, but the task name will typically extend outside of the taskbar. If False, the taskname should probably appear in the legend of a Gantt chart, else hard to figure out just from the task ID (Gantt chart taskbar will show task ID like T 1.2, using the standard templates).

Default: True

ganttDistanceBetweenMS
A compact layout of a Gantt chart requires that deliverables and milestones are set on the same line (else, every milestone/deliverable marker on separate line will create very long Gantt charts). This setting controls how many month a marker "occupies", how much space it needs before another marker can be set on the same line. Minimum value is 1, for the marker itself. Set it to a larger value to have each marker on its own line.

Default: 4

milestoneDecoration = fill=orange, rounded corners
How should a milestone look like? Put a "decoration string" according to the pdfgantt package here. (Use the GanttPrefix if you want to change the look of a deliverable marker).

Default: 5pt

milestonesShowCrossWP
Show cross-WP milestones? That means: Each milestone has a "home" workpackage where it is defined. But it can have contributing tasks from other WPs as well (called a "cross-WP" milestone). Should the milestone appear also in those "contributing WP"'s Gantt charts? (it will always appear in its home WP's Gantt chart)

Default: True

deliverablesShowCrossWP
Same thing: show cross-WP deliverables?

Default: True

milestoneLegendTemplate
Template to format the milestone legends in the Gantt charts. The useful when the legends are typeset using a itemize-like list. This setting will generate the milestoneGanttLegend (or deliverableGanttLegend); it is more a convenience function and could also be done in latexTemplates, but would be a little cumbersome there).

A plausible alternative for the legend strings is to use a description environment instead. That requires then a corresponding change in latexTemplates.cfg. Not difficult to do, look for compactitem there.

Default: \item \textbf{${id}}: ${Title}

deliverableLegendTemplate
See milestoneLegendTemplate for details.

Default: \item \textbf{${id}}: ${Title}

7.8.4 Summaries

Which summary tables, figures should appear in the document? Mostly True/False switches.

showEffortPartnerWPs
One table showing efforts only per partner and workpackage, over all workpackages

Default: True

showEffortPartnerTasks
One table showing effort per partner and task, over all tasks

Default: True

piePMsWPs
A pie chart, showing person month summaries for WPs

Default: True

piePMsPartners
A pie chart, showing person month summaries for each partner

Default: True

piePMsPartnerTypes
A pie chart, showing person month summaries for each partner type (industry, SME, academic)

Default: True

piePMsNations
A pie chart, showing person month summaries for each nation

Default: True

7.8.5 WPTables

This section collexts all information that is relevant for the actual workpackage tables.

maxPartnersPerRow
How many partners should be typeset in one row of the WP tables? The choice depends mostly on how long the partner shortnames are; 8 seems to be a good compromise between readability and space effiency.

Default: 8

firstColumnWidth
How wide should the first column be? Give the value as percent of textwidth!

Default: 15

wptablespacing
How to influence the spacing of the wp tables?

Default: @{\hskip 0ex}

tabularCorrection
Correction factor to eadjust total width of the WP tabular. This does require some fiddling if any of the above values is changed. (Note: I'd much appreciate help from a LaTeX wizard to ensure a tabular environment is EXACTLY textwidth wide)

Default: 0.95

tasklistShowsDuration
The WP forms has a section with a list of all the tasks. Should this list show the duration of a task? (Details can be fixed in latexTemplates.cfg, section WpTasks

Default: True

tasklistShowsPartners
Should it show which partners contribute to the task?

Default: True

tasklistShowsDeliverables
Should it show the deliverables to which this task contributes?

Default: True

tasklistShowsMilestones
And the milestones?

Default: True

wpdescriptionShowsLeader
The box "Description of Workpackage" - should it show the WP Leader?

Default: True

taskboxShowsLeader
Apart from the WP description, each task has an invidual box. How much details should they report? (this is used in latexTemplates.cfg in the WpTasksDescriptions section) Should it show the Leader of the particular task?

Default: True

taskboxShowsObjectives
Should it show the objectives of the task? (If this is False, this information does not appear anywhere.)

Default: True

taskboxShowsDescription
Should it show the description of the task? (If this is False, this information does not appear anywhere.)

Default: True

taskboxShowsDeliverables
Should it list the Deliverables contributed by each task? The standard template typesets this as a table at the end of each taskbox; easy to change in the WpTasksDescriptions section of latexTemplates.cfg

Default: True

taskboxShowsMilestones
And correspondingly the milestones?

Default: True

taskboxShowsPartners
And that taskbox can show the partners active in this task.

Default: True

deliverablesWPshowDue
Each workpackage has a list of deliverables it produces (only the "own" deliverables are relevant here). This list can contain several information; here, the Due date can be turned on or off. These settings are used in WpDeliverables in latexTemplates.cfg.

Default: True

deliverablesWPshowTasks
Should the deliverable list show which tasks contribute to a particular deliverable.

Default: True

deliverablesWPshowPartners
Similarly, which partners contribute to the deliverable?

Default: True

deliverablesWPshowDescription
And finally, a deliverable can have a description. Should it be shown? (If this option is False, the deliverable description does not appear anywhere, when using the standard templates).

Default: True

milestonesWPshowTasks
Similar to the deliverables, a list of milestones per WP can be controlled. See the WpMilestones section in latexTemplates.cfg for details.

Default: True

milestonesWPshowPartners
Default: True

milestonesWPshowDue
Default: True

milestonesWPshowDescription
Default: True

colorInactivePartner
Should an inactive partner (zero effort in a given workpackage)= be shown in a separate color? (color names as defined by the LaTeX xcolor package) (No highlight: simply use black)

Default: gray

7.8.6 Participants

Options to control the individual participant descriptions in Section 2.2

newpageAfterEachPartner
Should there be a newpage inserted after each partner description? To ensure that each partner description starts on an empty page?

Default: True

7.8.7 LaTeX

The LaTeX section coleects options controlling LaTeX processing. These are simle True/False flags to turn on/off various parts of the proposal. NOTE: these settings are only processed in the makefile; changing them and directly running pdflatx will have no effect.

showCommissionHints
Should the PDF file include the commission hints text? (as defined on the wiki by the commissionhint level 5 headings on the wiki pages). It is HIGHLY recommended to turn this off before producing the final version of the proposal.

Default: True

useShowkeys
Should the showkey package be used, highlighting the label, ref and cite commands? This can be useful to have an idea which labels exist for cross-referencing and looking for errors. Turn this off for the final version.

Default: False

showWarnings
Should warnings and fixmes be printed? This includes the list in warnings.tex. Turn this off for the final version.

Default: True

showListOfTables
Should the table of content show list of tables?

Default: True

showListOfFigures
Shoudl the table of content show list of figures?

Default: True

showAcronymList
Should there be a list of acronyms? main.tex include a file AcronymsList.tex. Your choice how to fill this list. NOTE: AcronymsList.tex is not included in the default distribution. Just setting this flag to True will result in an error.

Default: False

useMultipageDeliverableTable
Should the deliverable and milestone collection table be typeset using a standard tabular environment or a multipage tabularx environment? Short lists will look better in singlepage mode, but obviously long lists have to be split up over multiple pages. See latexTemplates.cfg, sections DeliverableTable and MilestoneTable for details. (the defaults False / True only for demonstration of the options, pick what you prefer...)

Default: False

useMultipageMilestoneTable
Default: True

useMultipageEffortTable
Should the effort table for the entire project be typesetting across multiple pages? (Similar to above, see latextemplates.cfg, SEction effortPerTaskTable).

Default: True

effortTableLandscape
Turn the effort table sideways? Can be useful for large consortia. Details are in effortPerTaskTableShort and effortPerTaskTableMultipage sections of latexTemplates.cfg.

Default: True

7.8.8 CustomLaTeX

A section for custom LaTeX commands. Rationale: there might be some things you'd like to include in your proposal that are not fit for making them general, but can be computed based on the numbers contained in files pulled form the wiki. E.g., the toal number of person months to this end, this section allows you to write python code that is executed once every thing else is done and assign the result to a LaTeX command. The defining command will end up in settings.tex; the option name will be used as the LaTeX command, to be replaced by the result of evaluating the python code here. In particular, you have access to the variables described in section *latexFromXML*.

CAUTION: you can really screw up everything here. It can delete your disk, insult for boss, and kill your pet. You are WARNED! To make use of this feature, you have to understand the Python code!

totalPM
As an example: total person months should be defined like this. This command is actually used.

Default: sum([int(e['resources']) for e in allEfforts])

tocLevel
Set the toc level for LaTeX; easier than to manipulate the main.tex file.

Default: 3

secNumDepth
How deeply should headings be numbered?

Default: 3

7.9 latexTemplates.cfg

This file contains the input for a little templating engine.

7.9.1 titleheader

Put all the templates to be used for LaTeX configuration here. this file works in concert with the settings.cfg file; some changes there have to be reflected here (see comments). Note on syntax: LaTeX template strings will often be multi-line. To get this right: First line must be on the same line as the =; contiunation lines must start with a whitespace character! See python library under ConfigParser or RFC 822, section 3.1.1 we build the titlepage in three steps: the header, the table rows for the partners, and then the titlepage complete out of these two parts. This third part is then written to file

template
Default: \begin{center} {\LARGE ${instrument} } \\[.2cm] {\large ${call} } \\[.4cm] {\LARGE \textbf{ ${projectname} }} \\[.3cm] {\LARGE Acronym: \textbf{ ${projectshort} }} \\[.3cm] \end{center} {\large Date of Preparation: \today }\\[1em] \begin{large} \begin{description} \item[Work program topics addressed:] ${topics} \item[Coordinator:] ${coordinatorname} \item[e-mail:] {\url{${coordinatoremail}}} \item[tel/fax:] ${coordinatorphone} \end{description} \end{large} \noindent

dict
Default: titlepageDict

7.9.2 partnerTableRow

Rows for partner table on titlepage:

template
Default: ${Number} & ${Name} & ${Shortname} & ${Nation}

list
Default: partnerList

joiner
Default: \\ \n

sorter
To demonstrate how to sort such a list, let's sort if by number Note: sorter is optional, but only available in conjunction with joiner attribute some alternative examples (which make no sense here, just to demonstrate): sorter = lambda x: x['Shortname'] sorter = lambda x: x['Nation']

Default: lambda x: int(x['Number'])

7.9.3 titlepage

And the actual titlepage:

template
Default: ${titleheader} { \begin{tabular}{cp{8cm}ll} \toprule Participant no. & Participant organisation & Short name & Country \\ \midrule ${partnerTableRow} \\ \bottomrule \end{tabular} }

dict
Default: expanded

file
Default: True

7.9.4 wpSummaryRows

The wp summary table; first the individual rows, one per WP:

template
Default: WP ${Number} & ${Name} & ${Type} & ${Leadernumber} & ${Leadership} & ${wpeffort} & ${Start} & ${End}

list
Default: allWPDicts

joiner
Default: \\ \n

7.9.5 wpsummarytable

Then build the complete WP summary table.

template
Default: \begin{table}[bhtp] \caption{Summary table of all work packages} \label{tab:wpsummary} \begin{tabular}{cp{0.25\textwidth}cccrcc} \toprule WP No. & WP name & Type of & Lead & Lead & Person- & Start & End \\ & & activity & part. no. & short name & months & month & month \\ \midrule ${wpSummaryRows} \\ \midrule \multicolumn{2}{l}{Total:} & & & & \totalPM & & \\ \bottomrule \end{tabular} \end{table}

dict
Default: expanded

file
Default: True

dir
Default: tables

7.9.6 ganttPrefix

Gantt charts start here! First, some building blocks for various Gantt charts

template
Default: \begin{tikzpicture} \begin{ganttchart}[vgrid,hgrid, x unit=0.371cm, y unit chart = 0.75cm, title label font={\footnotesize}, bar height = 0.55, bar top shift = 0.225, inline, milestone label font=\color{black}\small, milestone label inline anchor={right=.1cm}, bar label inline anchor={anchor=west}, bar label font=\small, link={-latex, rounded corners=1ex, thick}]{${duration}} \gantttitlelist{1,...,${duration}}{1}

dict
Default: titlepageDict

7.9.7 ganttPostfix

template
Default: \end{ganttchart} \end{tikzpicture}

7.9.8 WpMilestonesUncompressedShow

and the actual gantts: first, the wp-specific gantts

template
Default: \begin{figure}[htbp] ${ganttPrefix} \\ ${milestoneUncompressedGanttString} ${ganttPostfix} \caption{Gantt chart of all milestones of Work package ${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

7.9.9 WpMilestonesShow

template
Default: \begin{figure}[htbp] ${ganttPrefix} \\ ${milestoneGanttString} ${ganttPostfix} \caption{Gantt chart of all milestones of Work package ${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

7.9.10 WpDeliverablesUncompressedShow

template
Default: \begin{figure}[htbp] ${ganttPrefix} \\ ${deliverableUncompressedGanttString} ${ganttPostfix} \caption{Gantt chart of all deliverables of Work package ${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

7.9.11 WpDeliverablesShow

template
Default: \begin{figure}[htbp] ${ganttPrefix} \\ ${deliverableGanttString} ${ganttPostfix} \caption{Gantt chart of all deliverables of Work package ${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: True

numerator
Default: value['Shortname']

dir
Default: gantts

7.9.12 ganttWP

template
Default: \begin{figure}[htbp] \centering{${ganttPrefix}\\ \ifthenelse{\boolean{Gantts-ShowWPBar}}{ ${groupbar}}{} ${taskGantt} ${deliverableGanttString} \\ ${milestoneGanttString} ${ganttPostfix}} \ifthenelse{\boolean{Gantts-ganttPerWPShowsLegend}}{ \ifthenelse{\boolean{Gantts-ganttLegendTwoColumn}} {\begin{multicols}{2} \begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} \end{compactitem} \end{multicols} } { % single-column legends: \begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} \end{compactitem} } }{} \caption{Gantt chart for Work package ${Number}: ${Shortname}} \label{fig:gantt-WP${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

numerator
do not change the numerator; else, wPInclude.tex will look for the wrong files

Default: value['Shortname']

dir
Default: gantts

7.9.13 ganttWPLegend

template
separate legends per WP gantts only make sense if they are not already included in the

Default: \begin{figure}[htbp] \ifthenelse{\boolean{Gantts-ganttLegendTwoColumn}} {\begin{multicols}{2} \begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} \end{compactitem} \end{multicols} } {% single-column legends: \begin{compactitem} ${deliverableGanttLegend} ${milestoneGanttLegend} \end{compactitem} } \caption{Gantt chart of Work package ${Number}: ${Shortname}} \label{fig:gantt-Legend-WP${Number}} \end{figure}

list
Default: allWPDicts

dict
Default: expanded

file
Default: False

numerator
do not change the numerator; else, wPInclude.tex will look for the wrong files

Default: value['Shortname']

dir
Default: gantts

7.9.14 allTaskDelMSList

and prepate the complete Gantt chart for the entire project this happens in several steps

template
Default: \ifthenelse{\boolean{Gantts-ShowWPBar}}{ ${groupbar}{} ${taskGantt} ${deliverableGanttString} \\ ${milestoneGanttString}

list
Default: allWPDicts

dir
Default: gantts

7.9.15 allTaskDelMS

the next one is an example with an empty template: it is only used to turn a list into a string

template
Default:

list
Default: expanded['allTaskDelMSList']

joiner
Default: \\ \n

7.9.16 allDelLegend

template
Default: ${ganttLegend}

list
Default: allDeliverables

joiner
Default: \n

7.9.17 allMSLegend

template
Default: ${ganttLegend}

list
Default: allMilestones

joiner
Default: \n

7.9.18 CompleteGantt

template
Default: \begin{figure}[htbp] \centering\maxsizebox{0.95\textwidth}{0.95\textheight}{ ${ganttPrefix} \\ ${allTaskDelMS} ${ganttPostfix} } % closes adjustbox \caption[Overall Gantt chart for the entire project,

showing all tasks, deliverables, and milestones]{Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones (legend in Table~\ref{fig:allDelMSLegend})} \label{fig:completeGantt} \end{figure}

dict
Default: expanded

file
Default: True

dir
Default: gantts

7.9.19 allLegend

template
Default: \begin{table} \caption[List of all deliverables and milestones]{List of all deliverables and milestones shown in Figure~\ref{fig:completeGantt}} \label{fig:allDelMSLegend} \begin{multicols}{2} \begin{compactitem} ${allDelLegend} ${allMSLegend} \end{compactitem} \end{multicols} \end{table}

dict
Default: expanded

file
Default: True

dir
Default: gantts

7.9.20 CompleteGanttFacingLegend

template
this is particularly useful for a double-sided printing layout chart is on a left page, Legend on a right page

Default: \cleardoubleevenstandardpage \centering\maxsizebox{0.95\textwidth}{0.95\textheight}{ ${ganttPrefix} \\ ${allTaskDelMS} ${ganttPostfix} } % closes adjustbox \captionof{figure}[Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones]{Overall Gantt chart for the entire project, showing all tasks, deliverables, and milestones (legend in Table~\ref{fig:allDelMSLegend})} \label{fig:completeGantt} \clearpage \captionof{table}[List of all deliverables and milestones]{List of all deliverables and milestones shown in Figure~\ref{fig:completeGantt}} \label{fig:allDelMSLegend} \begin{multicols}{2} \begin{compactitem} ${allDelLegend} ${allMSLegend} \end{compactitem} \end{multicols}

file
Default: True

dict
Default: expanded

dir
Default: gantts

7.9.21 WpTasks

and finally: the actual WP files!

template
Default: \labitem{${taskId}}{task:${Label}} ${Name} \ifthenelse{\boolean{WPTablestasklistShowsDuration}}{ \hfill (%{ ‘, ‘.join([(‘M\,’ + str(t[’Start’]) + ‘ – M\,’ + str(t[’Start’] + t[’Duration’] -1)) for t in allTasks if t[’Label’] == ‘${Label}’]) %}) }{} \ifthenelse{\boolean{WPTablestasklistShowsPartners}}{ \\ Contributing partners: %{ ‘, ‘.join([((r”\textbf{%s}” % x) if (x ==

'${Leadpartner}') else x) for x in sorted([p['partner'] for p in allEfforts if p['task'] == '${Label}'], key = lambda x: [int(pl['Number']) for pl in partnerList if pl['Shortname'] == x][0])]) %} }{} \ifthenelse{\boolean{WPTables-tasklistShowsDeliverables}}{ %{ (r'\\ Contributing to Deliverables: ' + ', '.join([((r"\textbf{ %s }" % d['id']) if (d['ProducingtaskMain'] == '${Label}') else d['id']) for d in allDeliverables if '${Label}' in d['Producingtask']]) if [d['id'] for d in allDeliverables if '${Label}' in d['Producingtask']] else '') %} }{} \ifthenelse{\boolean{WPTables-tasklistShowsMilestones}}{ %{ (r'\\ Contributing to Milestones: ' + ', '.join([((r"\textbf{ %s }" % d['id']) if (d['ProducingtaskMain'] == '${Label}') else d['id']) for d in allMilestones if '${Label}' in d['Producingtask']]) if [d['id'] for d in allMilestones if '${Label}' in d['Producingtask']] else '') %} }{}

list = [t for t in allTasks if t['Main']=
we only show this for the main task, not for all the individual tasks need to think about the duration, though!

Default: 'True']

groupby
Default: wp

joiner
Default: \n

7.9.22 WpTasksDescriptions

template
Default: \begin{framed} \noindent \textbf{Description of Task ${taskId}: ${Name}} \ifthenelse{\boolean{WPTables-taskboxShowsLeader}}{(Task leader: ${Leadpartner})}{} \ifthenelse{\boolean{WPTables-taskboxShowsObjectives}}{ ~\\[0.2cm] \textbf{Task objectives:} ${taskobjectives}}{} \ifthenelse{\boolean{WPTables-taskboxShowsDescription}}{~\\[0.2cm] \textbf{Description of work:} ${taskdescription}}{} \ifthenelse{\boolean{WPTables-taskboxShowsDeliverables}}{ %{ (r'"\\[0.3cm] \noindent {\centering \begin{tabular}{lp{0.7\textwidth}l} \multicolumn{3}{l}{\textbf{Deliverables contributed to by Task ${taskId}:} } \\ \toprule Del.\ no. & Deliverable name & Due \\ \midrule %s \\ \bottomrule \end{tabular} }"' % (r'\\ '.join([(d['id'] + " & " + d['Title'] + " & M\," + str(d['Monthdue'])) for d in allDeliverables if '${Label}' in d['Producingtask']])) if [d for d in allDeliverables if '${Label}' in d['Producingtask']] else "") %} }{} \ifthenelse{\boolean{WPTables-taskboxShowsMilestones}}{ %{ (r'"\\[0.3cm] \noindent {\centering \begin{tabular}{lp{0.7\textwidth}l} \multicolumn{3}{l}{\textbf{Milestones contributed to by Task ${taskId}:} } \\ \toprule MS.\ no. & Milestone name & Due \\ \midrule %s \\ \bottomrule \end{tabular} }"' % (r'\\ '.join([(d['id'] + " & " + d['Title'] + " & M\," + str(d['Monthdue'])) for d in allMilestones if '${Label}' in d['Producingtask']])) if [d for d in allMilestones if '${Label}' in d['Producingtask']] else "") %} }{} \ifthenelse{\boolean{WPTables-taskboxShowsPartners}}{ ~ \\[0.2cm] \noindent \textbf{Partners contributing to this task:} %{ ", ".join([(r'\textbf{%s}' % x['partner'] if x['partner'] == '${Leadpartner}' else x['partner']) for x in allEfforts if x['task'] == '${Label}' and int(x['resources']) > 0]) %} }{} \end{framed}

list = [t for t in allTasks if t['Main']=
Default: 'True']

groupby
Default: wp

joiner
Default: \n

7.9.23 WpDeliverables

the deliverable and milestone list per WP, along with more detailed description

template
Default: \item \textbf{${id}: ${Title} \ifthenelse{\boolean{WPTables-deliverablesWPshowDue}}{ \hfill (M\,${Monthdue})}{} \ifthenelse{\boolean{WPTables-deliverablesWPshowTasks}}{ \\ Contributing

tasks: ${ProducingtaskString}}{} \ifthenelse{\boolean{WPTables-deliverablesWPshowPartners}}{ \\ Contributing partners: ${ContributorString}}{} \ifthenelse{\boolean{WPTables-deliverablesWPshowDescription}}{ \\ Brief description: ${Description}}{}

list
Default: allDeliverables

groupby
Default: wp

joiner
Default: \n

7.9.24 WpMilestones

template
Default: \item \textbf{${id}}: ${Title} \ifthenelse{\boolean{WPTables-milestonesWPshowDue}}{ \hfill (M\,${Monthdue})}{} \ifthenelse{\boolean{WPTables-milestonesWPshowTasks}}{ \\ Contributing tasks: ${ProducingtaskString}}{} \ifthenelse{\boolean{WPTables-milestonesWPshowPartners}}{ \\ Contributing partners: ${ContributorString}}{} \ifthenelse{\boolean{WPTables-milestonesWPshowDescription}}{ \\ Brief description: ${Description}}{}

list
Default: allMilestones

groupby
Default: wp

joiner
Default: \n

7.9.25 Wp

template
Default: \newpage \noindent \addcontentsline{toc}{subsubsection}{WP ${Number}: ${Shortname} } ${tableheader} \begin{framed} \noindent \textbf{Objectives of Workpackage ${Number}:} ${objectives} \end{framed} \begin{framed} \noindent \textbf{Tasks of Workpackage ${Number}:} \begin{compactdesc} ${WpTasks_${Number}} \end{compactdesc} {\footnotesize \emph{Lead partners are shown in bold.}} \end{framed} \begin{framed} \noindent \textbf{Description of Workpackage ${Number}:} \ifthenelse{\boolean{WPTables-wpdescriptionShowsLeader}}{(workpackage leader: ${Leadership})}{} ${wpdescription} \end{framed} ${WpTasksDescriptions_${Number}} \begin{framed} \noindent \textbf{Deliverables for Workpackage ${Number}:} \begin{compactdesc} ${WpDeliverables_${Number}} \end{compactdesc} \noindent \textbf{Milestones for Workpackage ${Number}:} \begin{compactdesc} ${WpMilestones_${Number}} \end{compactdesc} \end{framed} % let us pull in the gantt chart for this WP directly here, % no need to use a separate file: ${ganttWP_${Shortname}}

list
Default: allWPDicts

dict
Default: expanded

numerator
Default: value['Shortname']

file
Default: True

dir
Default: wp

7.9.26 DeliverableTableRows

a table to summarize all the deliverables again: two steps: build the rows, and then the table

template
Default: ${id} & ${Title} & ${Monthdue} & ${Nature} & ${Dissemination} & ${ProducingtaskString}

list
Default: allDeliverables

joiner
Default: \\ \n

sorter
this sorts according to due date, and where the due date is the same, use deliverable id:

Default: lambda x: '%03d' % x['Monthdue'] + x['id']

7.9.27 DeliverableTableShort

if you just want to sort by id, then simply use: sorter = lambda x: x['id']

template
Default: \begin{table}[hbtp] \caption{Summary of all deliverables (Nature: O=Other, P=Prototype, R=Report; Dissemination: PU=Public, RE=Restricted, CO=Confidential)} \label{tab:deliverablesummary} \begin{center} \begin{tabular}{l|p{0.4\textwidth}|c|c|c|p{0.1\textwidth}} \toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\\ \midrule ${DeliverableTableRows} \\ \bottomrule \end{tabular} \end{center} \end{table}

dict
Default: expanded

7.9.28 DeliverableTableLong

template
Default: \topcaption{Summary of all deliverables} \label{tab:deliverablesummary} \tablefirsthead{ \toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\\ \midrule } \tablehead{\toprule \multicolumn{6}{r}{\emph{Table~\ref{tab:deliverablesummary} continues from previous page}} \\ \toprule Number & Title & Due date & Nature & Diss. & Contributing task(s)\\ \midrule } \tabletail{\bottomrule \multicolumn{6}{r}{\emph{Table~\ref{tab:deliverablesummary} continues on next page}} \\ \bottomrule} \tablelasttail{\multicolumn{6}{r}{Table~\ref{tab:deliverablesummary} ends} \\ \bottomrule} \begin{center} \begin{mpxtabular}{l|p{0.4\textwidth}|c|c|c|p{0.1\textwidth}} ${DeliverableTableRows}\\ \bottomrule \end{mpxtabular} \end{center}

dict
Default: expanded

7.9.29 DeliverableTable

template
Default: \ifthenelse{\boolean{LaTeX-useMultipageDeliverableTable}}{${DeliverableTableLong}}{${DeliverableTableShor

dict
Default: expanded

file
Default: True

dir
Default: tables

7.9.30 MilestoneTableRows

and a table for the milestones - same structure as for the deliverable table

template
Default: ${id} & ${Title} & ${Monthdue} & ${Verificationmeans} & ${ProducingtaskString}

list
Default: allMilestones

joiner
Default: \\ \n

sorter
Default: lambda x: '%03d' % x['Monthdue'] + x['id']

7.9.31 MilestoneTableShort

template
Default: \begin{table}[hbtp] \caption{Summary of all milestones} \label{tab:milestonessummary} \begin{center} \begin{tabular}{l|p{0.3\textwidth}|c|p{0.3\textwidth}|p{0.1\textwidth}} \toprule Number & Title & Due date & Means of verification & Contributing task(s) \\ \midrule ${MilestoneTableRows}\\ \bottomrule \end{tabular} \end{center} \end{table}

dict
Default: expanded

7.9.32 MilestoneTableLong

template
Default: \topcaption{Summary of all milestones} \label{tab:milestonessummary} \tablefirsthead{ \toprule Number & Title & Due date & Means of verification & Contributing task(s) \\ \midrule } \tablehead{\toprule \multicolumn{5}{r}{\emph{Table~\ref{tab:milestonessummary} continues from previous page}} \\ \toprule Number & Title & Due date & Means of verification & Contributing task(s) \\ \midrule } \tabletail{\bottomrule \multicolumn{5}{r}{\emph{Table~\ref{tab:milestonessummary} continues on next page}} \\ \bottomrule} \tablelasttail{\multicolumn{5}{r}{Table~\ref{tab:milestonessummary} ends} \\ \bottomrule} \begin{center} \begin{mpxtabular}{l|p{0.3\textwidth}|c|p{0.3\textwidth}|p{0.1\textwidth}} ${MilestoneTableRows}\\ \bottomrule \end{mpxtabular} \end{center}

dict
Default: expanded

7.9.33 MilestoneTable

template
Default: \ifthenelse{\boolean{LaTeX-useMultipageMilestoneTable}}{${MilestoneTableLong}}{${MilestoneTableShort}}

dict
Default: expanded

file
Default: True

dir
Default: tables

7.9.34 summaryEffortRows

template
Default: ${Shortname} & %{ ' & '.join([str(x['partnereffort']['${Shortname}']) for x in allWPDicts]) %} & %{ str(sum([x['partnereffort']['${Shortname}'] for x in allWPDicts])) %}

list
Default: partnerList

joiner
Default: \\ \n

7.9.35 summaryEffort

template
Default: \ifthenelse{\boolean{Summaries-showEffortPartnerWPs}}{\begin{table} \caption{Summary of all effort over all partners and workpackages} \label{tab:summaryEffortperWP} \begin{center} \begin{tabular}{l%{'c'*len(allWPDicts)%}r} \toprule Partner & %{ ' & '.join(['WP\,%s' % x['Number'] for x in allWPDicts]) %} & Total \\ \midrule ${summaryEffortRows} \\ \midrule Total & %{ ' & '.join([x['wpeffort'] for x in allWPDicts]) %} & %{ str(sum([int(x['wpeffort']) for x in allWPDicts])) %} \\ \bottomrule \end{tabular} \end{center} \end{table} }{}

dict
Default: expanded

file
Default: True

dir
Default: tables

7.9.36 effortPerTaskRows

template
Default: ${taskId} & ${Name} & {\cellcolor[gray]{0.9} %{ str(sum([int(e['resources']) for e in allEfforts if e['task'] == '${Label}'])) %} } & %{ ' & '.join([((r'\textbf{%s}' % e['resources']) if '${Leadpartner}' == p['Shortname'] else (e['resources'] if int(e['resources']) > 0 else (r'{\textcolor{%s}0}' % config.get('WPTables','colorInactivePartner')))) for p in partnerList for e in [ee for ee in allEfforts if ee['task'] == '${Label}' and ee['partner'] == p['Shortname']]]) %}

list = [t for t in allTasks if t['Main']=
Default: 'True']

groupby
Default: wp

joiner
Default: \\ \hline \n

7.9.37 effortPerTaskRowsWP

template
Default: ${effortPerTaskRows_${Number}} \\ \hline \rowcolor[gray]{0.9} \multicolumn{2}{l|}{\cellcolor[gray]{0.9} \textbf{WP ${Number}:}} & %{ str(sum(${partnereffort}.values())) %} & %{ ' & '.join([r'\textbf{%s}' % str(${partnereffort}[p['Shortname']]) if p['Shortname']=='${Leadership}' else str(${partnereffort}[p['Shortname']]) for p in partnerList]) %}

list
Default: allWPDicts

joiner
Default: \\ \hline \hline \n

dict
Default: expanded

7.9.38 effortHeader

template
Default: \hline Task & Task name & Total PM & %{ ' & '.join([p['Shortname'] for p in partnerList]) %} \\ \hline

dict
Default: expanded

7.9.39 effortSum

template
Default: \rowcolor[gray]{0.8} \multicolumn{2}{l|}{\cellcolor[gray]{0.8} \textbf{Project total: }} & \textbf{\totalPM} & %{ ' & '.join([r'\textbf{%s}' % str(sum([int(e['resources']) for e in allEfforts if e['partner'] == p['Shortname']])) for p in partnerList]) %}

dict
Default: expanded

7.9.40 effortPerTaskTableShort

template
Default: \ifthenelse{\boolean{LaTeX-effortTableLandscape}}{\begin{landscape}}{} \begin{table} \caption{Effort per tasks and partners for entire project (in personmonths)} \label{tab:effortPerTasks} \begin{center} \small \begin{tabular}{c|p{0.15\textwidth}|c%{' |c' * len(partnerList)%}} ${effortHeader} ${effortPerTaskRowsWP} \\ \hline \hline ${effortSum} \\ \hline \hline \end{tabular} \end{center} \end{table} \ifthenelse{\boolean{LaTeX-effortTableLandscape}}{\end{landscape}}{}

dict
Default: expanded

7.9.41 effortPerTaskTableMultipage

template
Default: \ifthenelse{\boolean{LaTeX-effortTableLandscape}}{\begin{landscape}}{} \topcaption{Effort per tasks and partners for entire project (in personmonths)} \label{tab:effortPerTasks} \tablefirsthead{ ${effortHeader} } \tablehead{ \toprule \multicolumn{%{str(3+len(partnerList))%}}{r}{\emph{Table~\ref{tab:effortPerTasks} continues from previous page}} \\ ${effortHeader}} \tabletail{\bottomrule \multicolumn{%{str(3+len(partnerList))%}}{r}{\emph{Table~\ref{tab:effortPerTasks} continues on next page}} \\ \bottomrule} \tablelasttail{ \multicolumn{%{str(3+len(partnerList))%}}{r}{\emph{Table~\ref{tab:effortPerTasks} ends}} \\ \bottomrule} \begin{center} \begin{mpxtabular}{c|p{0.15\textwidth}|c%{' |c' * len(partnerList)%}} ${effortPerTaskRowsWP} \\ \hline \hline ${effortSum} \\ \hline \hline \end{mpxtabular} \end{center} \ifthenelse{\boolean{LaTeX-effortTableLandscape}}{\end{landscape}}{}

dict
Default: expanded

7.9.42 effortPerTaskTable

template
Default: \ifthenelse{\boolean{Summaries-showEffortPartnerTasks}} {\ifthenelse{\boolean{LaTeX-useMultipageEffortTable}} {${effortPerTaskTableMultipage}} {${effortPerTaskTableShort}} }{}

dict
Default: expanded

file
Default: True

dir
Default: tables

7.9.43 piePMsPartners

pie charts a pie chart showing person months distributed over partners

template
Default: \ifthenelse{\boolean{Summaries-piePMsPartners}}{ \begin{figure}[htbp] \centering \begin{tikzpicture} \pie[scale font]{ %{ utils.roundPie ([(x['Shortname'], sum([int(e['resources']) for e in allEfforts if e['partner']==x['Shortname']])) for x in partnerList]) %}} \end{tikzpicture} \caption{Distribution of person months over partners (in percent)} \label{fig:pie:pm:partner} \end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

7.9.44 piePMsWPs

template
Default: \ifthenelse{\boolean{Summaries-piePMsWPs}}{ \begin{figure}[htbp] \centering \begin{tikzpicture} \pie[scale font]{ %{ utils.roundPie ([('WP\,%s: %s'%(x['Number'], x['Shortname']), sum(x['partnereffort'].values())) for x in allWPDicts]) %}} \end{tikzpicture} \caption{Distribution of person months over work packages (in percent)} \label{fig:pie:pm:wp} \end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

7.9.45 piePMsNations

pie chart over person months assigned to different nations a little bit more complicated: we need to pull out the (Nation/Effort) pairs. those we get from allEfforts, where we look up the nation in the partnerList that leaves us with many entries in the list with the same nation we add up those efforts by a mapReduce operation to which we pass a suitable reduce function: adding up two values

template
Default: \ifthenelse{\boolean{Summaries-piePMsNations}}{ \begin{figure}[htbp] \centering \begin{tikzpicture} \pie[scale font]{ %{ utils.roundPie(utils.mapReduce ([(utils.searchListOfDicts(partnerList, 'Shortname', e['partner'], 'Nation'), int(e['resources'])) for e in allEfforts], lambda a,b: a+b)) %}} \end{tikzpicture} \caption{Distribution of person months over nations (in percent)} \label{fig:pie:pm:nations} \end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

7.9.46 piePMsPartnerTypes

same thing that worked for the nation pie charts works for the partner type pie charts as well

template
Default: \ifthenelse{\boolean{Summaries-piePMsPartnerTypes}}{ \begin{figure}[htbp] \centering \begin{tikzpicture} \pie[scale font]{ %{ utils.roundPie(utils.mapReduce ([(utils.searchListOfDicts(partnerList, 'Shortname', e['partner'], 'Type'), int(e['resources'])) for e in allEfforts], lambda a,b: a+b)) %}} \end{tikzpicture} \caption{Distribution of partner types over nations (in percent)} \label{fig:pie:pm:partnertype} \end{figure} }{}

dict
Default: expanded

file
Default: True

dir
Default: pies

7.10 Makefile

.. code-block:: make

```
# See LICENCE file for licencing information

# Where is the settings file? PAth is relative to the main directory (same as this Makefile)
SETTINGS = settings.cfg

# Which flags to use? -v is verbose for all scripts;
# in production use, -v is usually not necessary
# FLAGS = -v
FLAGS =


# Relevant path names. They are extracted from settings.cfg

PROJECTNAME  = $(strip $(shell grep "projectName " ${SETTINGS} | cut -f 2 -d = ))
BINPATH = $(shell grep "binpath " ${SETTINGS} | cut -f 2 -d = )
WIKIPATH = $(shell grep "wikipath " ${SETTINGS} | cut -f 2 -d = )
XMLPATH = $(shell grep "xmlpath " ${SETTINGS} | cut -f 2 -d = )
LATEXPATH = $(strip $(shell grep "manuallatexpath " ${SETTINGS} | cut -f 2 -d = ))
GENERATEDLATEXPATH = $(shell grep "genlatexpath " ${SETTINGS} | cut -f 2 -d = )
LATEXLINKS =  $(shell find ${LATEXPATH} -type l)
```

```
###################################
.PHONY: proposal pdf clean pullproject xml latexFromWiki latexFromXML ensureSymbolicLinks fullcomm
###################################


proposal:
        make pullproject
        make xml
        make latexFromWiki
        make latexFromXML
        make ensureSymbolicLinks


pullproject:
        cd ${BINPATH} ; python pullProject.py -s ../$(SETTINGS) $(FLAGS)

xml:
        cd ${BINPATH} ; python generateXML.py -s ../$(SETTINGS) $(FLAGS)

latexFromWiki:
        cd ${BINPATH} ; python latexFromWiki.py -s ../$(SETTINGS) $(FLAGS)

latexFromXML:
        cd ${BINPATH} ; python latexFromXML.py  -s ../$(SETTINGS) $(FLAGS)

ensureSymbolicLinks:
        cd ${BINPATH} ; python ensureSymbolicLinks.py   -s ../$(SETTINGS) $(FLAGS)

pdf:
        cd ${LATEXPATH}; pdflatex main; bibtex main; pdflatex main; pdflatex main
        cp ${LATEXPATH}/main.pdf ${PROJECTNAME}.pdf


clean:
        find ${WIKIPATH} -type f -print | grep -v README | xargs rm
        find ${XMLPATH} -type f -print | grep -v README | xargs rm
        find ${GENERATEDLATEXPATH} -type f -print | grep -v README | xargs rm
        # remove empty symbolic links from latex path - this is debatable!
        cd ${LATEXPATH} ; rm -f main.aux main.lof main.log main.lot main.lox main.out main.toc ma
        for d in ${LATEXLINKS}; do test ! -e $$d && rm $$d ; done

doc:
        cd docsource ; make install
doclean:
        find doc -type f -print | grep -v README | xargs rm

fullcommit:
        make clean
        make proposal
        make pdf
        cd docsource ; make install
        make clean
        git commit -a -m "a full commit triggered by the makefile"
```

7.11 main.tex

TODO

```
\documentclass[a4paper,11pt,twoside]{scrreprt}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% packages were the user might want to configure a few things:


% should there be space for page headings in the page layout?
\usepackage{fullpage}
% \usepackage[headings]{fullpage}

%% KOMA Options (for details and further options, look into the KOMA
%% documentation)
% should there be horizontal lines in header and footer?
% \KOMAoptions{headsepline=true,footsepline=true}

% should there be standard headings?
%\KOMAoptions{headings=normal}


% how should fixme's be displayed?
\usepackage[inline,nomargin,draft]{fixme}

% to display the commission hints:
\newcommand{\commissionhints}[1]{\small \textit{#1}}




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% packages that are needed or highly advisable,
% but which should not needed any configuration

\usepackage[utf8]{inputenc}

\usepackage{checkend}
\usepackage{varioref}
\usepackage{subfigure}
\usepackage{hyperref}
\usepackage{booktabs}
\usepackage{xtab}
\usepackage{tabularx}
\usepackage{adjustbox}
\usepackage{capt-of}
\usepackage{acronym}

\usepackage{pdflscape}
\newlength\landscapewidth
\newlength\landscapeheight

\usepackage{multicol}

\usepackage[table]{xcolor}
\usepackage[alwaysadjust]{paralist}
\usepackage{url}
\usepackage{calc}
\usepackage{ifthen}
\usepackage{tikz}
\usepackage{styles/pgf-pie}

\usepackage{framed}
\usepackage{comment}

\usepackage{mathptmx}
\usepackage[scaled=.90]{helvet}
\usepackage{courier}
```

```
\usepackage{textcomp}           %for euro symbol

\usepackage{graphicx}
\graphicspath{{Figures/}{figures/}}

%%% make sure to use the pgfgantt file from the SVN!
\usepackage{styles/pgfgantt}

%%% Needed to redefine bibliography environment to get rid of the
%%% chapter heading
\usepackage{styles/bibhack}



%%%% to get references to WPs and tasks
\newcounter{wpcounter}
\newcounter{taskcounter}[wpcounter]


%%%% to get references to the deliverables
\makeatletter
\newcommand{\labitem}[2]{%
\def\@itemlabel{\textbf{#1}:}
\item
\def\@currentlabel{#1}\label{#2}}
\makeatother



% compact bibliography:
\let\oldbibliography\thebibliography
\renewcommand{\thebibliography}[1]{%
  \oldbibliography{#1}%
  \setlength{\itemsep}{0pt}%
}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% import all the settings that result from switches in settings.cfg
\input{settings}

% how deeply should headings be numbered?
\setcounter{secnumdepth}{\secNumDepth}

% to what level should the table of content display sections?
\setcounter{tocdepth}{\tocLevel}

%%%%%
% compact lists options
% TODO: check whether this works before loading the package?
\setlength{\pltopsep}{0.5ex}
\setlength{\plitemsep}{0.25ex}


\begin{document}

\setlength\landscapewidth{\textheight}
\setlength\landscapeheight{\textwidth+2\headsep}
```

```
\pagenumbering{roman}
\thispagestyle{empty}
\input{titlepage}


\newpage
\noindent \textbf{Proposal Abstract}
\addcontentsline{toc}{chapter}{\numberline{}Abstract}
\par\medskip\noindent
\input{../generated/latex/ProposalAbstract}


{\small\tableofcontents
\ifthenelse{\boolean{LaTeX-showListOfTables}}{
  \listoftables
  \addcontentsline{toc}{chapter}{\numberline{}List of Tables}
}{}
\ifthenelse{\boolean{LaTeX-showListOfFigures}}{
  \listoffigures
  \addcontentsline{toc}{chapter}{\numberline{}List of Figures}
}{}
\ifthenelse{\boolean{LaTeX-showAcronymList}}{
  \input{AcronymsList}}{}
}


\ifthenelse{\boolean{LaTeX-showWarnings}}{
\listoffixmes
\IfFileExists{warnings.tex}{
  \begin{compactitem}
    \input{warnings}
  \end{compactitem}}{No warnings found!}
}{\renewcommand{\fxnote}[1]{}
\renewcommand{\fxwarning}[1]{}
\renewcommand{\fxerror}[1]{}
\renewcommand{\fxfatal}[1]{}
}


\cleardoublepage
\setcounter{part}{2}% part B

\chapter{Scientific and Technical Quality}
\label{chap:quality}
\pagenumbering{arabic}
\commissionhints{MAximum length for the whole of Section 1
--  twenty pages, not including the tables in Section 1.3}

\input{ConceptAndObjectives}

\input{ProgressBeyondStateoftheArt}


\input{MethodologyWorkplan}


\input{WorkPlanning}



\input{figures/tables/wpsummarytable}

%%% a little demo to include all gantt charts, likely not useful!
```

```
% \input{figures/gantts/demoGantts}

% long Gantt charts might be best put on two pages, left the chart,
% right the legend:
%\input{figures/gantts/CompleteGanttFacingLegend}

% smaller Gantt chart and legend might be nicely put in
% standard floating environments
\input{figures/gantts/CompleteGantt}
\input{figures/gantts/allLegend}

% %%%

\input{DeliverableList}


% \input{tables/alldeliverableTable}


\subsection{Work Packages}
\label{sec:workpackages}
\input{wp/wpIncluder}

\clearpage
\input{figures/tables/summaryEffort}
\input{figures/tables/effortPerTaskTable}




\chapter{Implementation}
\label{chap:implementation}


\input{ManagementStructureAndProcedures}



\section{Individual Participants}
\label{sec:partners}
\commissionhints{  For each participant in the proposed project, provide a brief description of th
  organisation, the main tasks they have been attributed, and the previous experience
  relevant to those tasks. Provide also a short profile of the staff members who will be
  undertaking the work.
  \\
Maximum length for Section 2.2: one page per participant. However, where two or more departments w
The maximum length applying to a legal entity composed of several members, each of which is a sepa
}

\input{partners/partnersIncluder}



\input{ConsortiumAsaWhole}



\input{SubContracting}

\input{OtherCountries}

\input{AdditionalPartners}
```

```
\input{ResourcesToBeCommitted}


%%%%%%%%

\input{figures/pies/piePMsWPs}
\input{figures/pies/piePMsPartners}
\input{figures/pies/piePMsNations}
\input{figures/pies/piePMsPartnerTypes}

% can't do contribs yet, needs spreadsheet input
% \ifthenelse{\boolean{Summaries-pieContribPerPartner}}{
% \begin{figure}[htbp]
%   \centering
%   \input{wp/contribPartner}
%   \caption{Distribution of EU contribution over partners  (in percent)}
%   \label{fig:contrib:pie:partner}
% \end{figure}
% }{}


% \ifthenelse{\boolean{Summaries-pieContribPerPartnerType}}{
% \begin{figure}[htbp]
%   \centering
%   \input{wp/contribType}
%   \caption{Distribution of EU contribution over partner kinds (in percent)}
%   \label{fig:contrib:pie:kind}
% \end{figure}
% }{}



% \ifthenelse{\boolean{Summaries-pieContribPerNation}}{
% \begin{figure}[htbp]
%   \centering
%   \input{wp/contribNation}
%   \caption{Distribution of EU contribution over partner nations  (in percent)}
%   \label{fig:contrib:pie:nation}
% \end{figure}
% }{}


\chapter{Impact}\label{chap:impact}
\commissionhints{Recommended length for the whole of Section 3 --- ten
  pages}


\input{ExpectedImpact}

\input{DisseminationExploitation}


\chapter{Ethical Issues}\label{chap:ethical}
\commissionhints{ %
  Describe any ethical issues that may arise in the project. In particular, you should
  explain the benefit and burden of the experiments and the effects it may have on the
  research subject. Identify the countries where research will be undertaken and which
  ethical committees and regulatory organisations will need to be approached during the
  life of the project.
\\
  Include the Ethical issues table below.  If you indicate YES to any issue, please
  identify the pages in the proposal where this ethical issue is described. Answering
```

```
  'YES' to some of these boxes does not automatically lead to an ethical review1.  It
  enables the independent experts to decide if an ethical review is required. If you are
  sure that none of the issues apply to your proposal, simply tick the YES box in the last
  row.
\\
In particular: Data protection issues: Avoid the unnecessary collection and use of
personal data. Identify the source of the data, describing whether it
is collected as part of the research or is previously collected data
being used. Consider issues of informed consent for any data being
used. Describe how personal identify of the data is protected. Data
protection issues require authorisation from the national data
protection authorities.
}

\begin{small}
\begin{tabular}{|p{1em}p{11cm}|l|l|}\hline
%   \multicolumn{2}{|l|}{\cellcolor{lightgray}{\strut}} &
  \multicolumn{2}{|l|}{{\strut}} &
 {YES} &
%   \cellcolor{lightgray}{PAGE}\\\hline
  {PAGE}\\\hline
 \multicolumn{2}{|l|}{\bf{Informed Consent}} & & \\\hline
  & Does the proposal involve children?  & & \\\hline
  & Does the proposal involve patients or persons not able to give consent? & & \\\hline
  & Does the proposal involve adult healthy volunteers? & & \\\hline
  & Does the proposal involve Human Genetic Material? & & \\\hline
  & Does the proposal involve Human biological samples? & & \\\hline
  & Does the proposal involve Human data collection? & & \\\hline
  \multicolumn{2}{|l|}{\bf{Research on Human embryo/foetus}}  & & \\\hline
  & Does the proposal involve Human Embryos? & & \\\hline
  & Does the proposal involve Human Foetal Tissue / Cells? & & \\\hline
  & Does the proposal involve Human Embryonic Stem Cells? & & \\\hline
  \multicolumn{2}{|l|}{\bf{Privacy}} & & \\\hline
  & Does the proposal involve processing of genetic information
        or personal data (eg. health, sexual lifestyle, ethnicity,
        political opinion, religious or philosophical conviction)  & & \\\hline
  & Does the proposal involve tracking the location or observation
        of people? & & \\\hline
  \multicolumn{2}{|l|}{\bf{Research on Animals}} & & \\\hline
  & Does the proposal involve research on animals? & & \\\hline
  & Are those animals transgenic small laboratory animals? & & \\\hline
  & Are those animals transgenic farm animals? & & \\\hline
  & Are those animals cloned farm animals? & & \\\hline
  & Are those animals non-human primates?  & & \\\hline
  \multicolumn{2}{|l|}{\bf{Research Involving Developing Countries}} & & \\\hline
  & Use of local resources (genetic, animal, plant etc) & & \\\hline
  & Benefit to local community (capacity building
        i.e. access to healthcare, education etc) & & \\\hline
  \multicolumn{2}{|l|}{\bf{Dual Use}} & & \\\hline
  & Research having direct military application  & & \\\hline
  & Research having the potential for terrorist abuse & & \\\hline
  \multicolumn{2}{|l|}{\bf{ICT Implants}} & & \\\hline
  & Does the proposal involve clinical trials of ICT implants?  & & \\\hline
  \multicolumn{2}{|l|}{\bf\footnotesize{I CONFIRM THAT NONE OF THE ABOVE ISSUES APPLY TO MY PROPOS
     &  \textbf{X} &\\\hline
\end{tabular}
\end{small}

\appendix
\chapter{References}

\begin{footnotesize}
  \bibliographystyle{savetrees}
```

```
  \bibliography{BibtexReferences}
\end{footnotesize}

\end{document}
```

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX

M

P

R

S

T

U

W