Preparing Proposals in LaTeX with `proposal.cls`

Michael Kohlhase
Computer Science, Jacobs University Bremen
http://kwarc.info/kohlhase

January 8, 2015

Abstract

The `proposal` class supports many of the generic elements of Grant Proposals. It is optimized towards collaborative projects, and should specialized to particular funding agencies.

Contents

1 Introduction

Writing grant proposals is a collaborative effort that requires the integration of contributions from many individuals. The use of an ASCII-based format like LaTeX allows to coordinate the process via a source code control system like Subversion, allowing the proposal writing team to concentrate on the contents rather than the mechanics of wrangling with text fragments and revisions.

The `proposal` class supports many of the generic elements of Grant Proposals. The package documentation is still preliminary, fragmented and incomplete.

The `proposal` class is distributed under the terms of the LaTeX Project Public License from CTAN archives in directory `macros/latex/base/lppl.txt`. Either version 1.0 or, at your option, any later version. The CTAN archive always contains the latest stable version, the development version can be found on GitHub at `https://github.com/KWARC/LaTeX-proposal`. For bug reports please use the issue tracker there.

2 The User Interface

In this section we will describe the functionality offered by the `proposal` class along the lines of the macros and environments the class provides.

2.1 Package Options

The `proposal` package takes the options `submit`, `noworkareas`, `RAM`, `deliverables`, `wpsubsection`, `keys`, `svninfo`, `gitinfo`, and `public`.

`submit` The `submit` option will disable various proposal management decorations which are enabled by default for submission.

`noworkareas` The `noworkareas` option specifies that we do not want to structure our work plan into work areas (see section 2.5).

`RAM` The `RAM` option specifies that we specify research assistant months in the effort tallies (see section 2.5).

`deliverables` The `deliverables` option specifies that we specify deliverables in the grant proposal (see section 2.9). As the deliverables management needs extra support, we only activate them via this option.

`wpsubsection` The `wpsubsection` option specifies that we want to see subsections headings for the WPs (and WAs, if we have them).

`report` The `report` option specifies that we want to use the `report.cls` class as a basis for `proposal` instead of the default `article.cls`.

`keys` The `keys` option specifies that we want to see the values of various keyval arguments in the margin.

`svninfo` The `svninfo` option specifies specifies that we want to use the `svninfo` package for displaying version control metadata in the document (except when the `submit` option is also given). For this we need the `svninfo` metadata line of the form

```
\SVN $Id: proposal.tex 13610 2007-07-11 04:30:16Z kohlhase $
\svnKeyword $HeadURL: https://svn.kwarc.info/../proposal.tex $
```

at the beginning of each file (or in the preamble).

`gitinfo` Analogously, the `gitinfo` option uses the `gitinfo2` package for GIT metadata. Note that you will need to install the post-commit hooks in your working copy according to [Lon] for this to work.

`public` `private` Finally, the `public` option allows to hide certain sensitive (e.g. financial) parts of the proposal. For this, the `proposal` class provides the `private` environment. If the option `public` is set, the parts of the document between `\begin{private}` and `\end{private}` do not produce output.

This is useful for producing public versions of the proposal that hide confidential parts. Note that both `\begin{private}` and `\end{private}` *have to be on lines of their own may not have any leading whitespace* otherwise an error occurs and LaTeX gives error messages that are difficult to comprehend. An alternative way to distinguish private and public sections are to use the `\ifpublic` conditional: `\ifpublic{3}\else{5}\fi` will result in "5" in the submitted draft and "3" in the public document.

`\ifpublic`

2.2 Proposal Metadata and Title page

`proposal` The metadata of the proposal is specified in the `proposal` environment, which also generates the title page and the first section of the proposal as well as the last pages of the proposal with the signatures, enclosures, and references. The `proposal` environment should contain all the mandatory parts of the proposal text. The `proposal` environment uses the following keys to specify metadata.

- `title` for the proposal title (used on the title page),
- `instrument` for the instrument of funding that you would like to apply for,
- `acronym` for the proposal acronym, possibly accompanied by an `acrolong` that explains it. The acronym will also be used in the page headings.
- `start` for the start date of the proposed fragment of the project, and `months` for the length of the proposal in months. Both have to be specified for the `proposal` class to work.
- If the proposal only concerns a part of a longer-running project, the `since` key allows to specify the date since when the overall project runs. Finally, the `fundsuntil` allows to specify a date until which the funds last.
- `discipline` for the academic discipline and `areas` for the research areas in that discipline.
- `PI` to declare the principal investigator. For collaborative proposals we can use the `PI` key multiple times. The `proposal` package uses the `workaddress` package for representation of personal metadata, see [Koh14c] or the file `proposal.tex` for details.
- Many collaborative proposals are shared between two institutions, which we can declare with the `site` key. As this changes the interface this should not be used for single-institution proposals. We will describe the setup for a single-site proposal below and point out the differences. The example `proposal.tex` is a two-site proposal.

`title` `instrument` `acronym` `acrolong` `start` `months` `since` `fundsuntil` `discipline` `PI` `site`

`\pn` `\pnlong` If the `acronym` and `acrolong` are given, then they automatically define the macros `\pn` and `\pnlong` which allow to use the project acronym (project nname) and its long version in the text. Note that these macros use `\xspace` internallly, so they do not have to be enclosed in curly braces.

2.3 Proposal Appearance

The `proposal` environment takes a second set of keyval arguments that allow to fine-tune the appearance of the proposal document. [1] EdN:1

- `compactht` If the `compactht` key is given (it does not need a value), then the header tables[2] are made compact, i.e. the sites that do not have a contribution to the work package or work area do not get listed. This is useful for proposals with more than 8 partners. EdN:2

`emphbox` The `proposal` package supplies the `emphbox` environment to create boxes of emphasized material we want to call attention to.

2.4 Objectives

The work plan starts with a discussion of objectives, which may be referenced in the text later. The `proposal` package provides the `objective` environment that allows to mark up individual objectives. It takes a keyval argument with the keys `id` for identification, `title` for the objective title, and `short` for a short title that can be used for referencing when the title is too long. The objectives can be referenced via `\OJBref{⟨id⟩}` by their label and via `\OJBtref{⟨id⟩}` by label

`objective` `\OBJref` `\OBJtref`

[1] EdNote: move the RAM, wpsectionheadings,... options here.

[2] EdNote: describe them somewhere and reference here

and (short if it was specified) title.

2.5 Work Areas and Work Packages

Grant proposals have another part that is often highly stylized; the work plan. This is usually structured into "work packages" — i.e. work items that address a cohesive aspect of the proposed work. These work packages are usually consecutively numbered, have a title, and an associated effort estimation. As work packages are the "atomic" planning units, they are usually heavily cross-referenced. A well-written proposal usually contains a table giving an overview over the work packages and their efforts and a Gantt chart showing the temporal distribution of the proposed work to allow the reviewers to get a clear picture of the feasibility of the research and development proposed. But this picture is also essential during the development of a proposal (which the `proposal` package aims to support), when the work packages (and their estimated efforts) usually change considerably. Therefore the `proposal` class standardizes markup for work packages and automatically computes the work package table (which can be inserted into the table via the `\wpfig` macro) and the Gantt Chart (see Section 2.8). `\wpfig`

`workplan` To achieve the automation, work plan is marked up by the `workplan` environment, which sets up various internal counters and bookeeping macros. It contains texts and `workpackage` environments for the work packages.

`workpackage` The purpose of the `workpackage` environment is to mark up a fragment of text as a work package description and specify the metadata so that it can be used in the work package table and Gantt chart generation. The metadata is specified by the following keys:

- `id` The `id` key is used to specify a label for cross-referencing the work package or work group, it must be document-unique.
- `title` `short` The `title` and `short` keys are used for the work package/group title. The short title is used in tables and should not be longer than 15 characters.
- `wphases` The `wphases` key is used according to Section 2.7
- `requires` The `requires` key can be used to mark, up dependencies between tasks. If `requires=\taskin{`$\langle rid\rangle$`}{`$\langle wp\rangle$`}` is given in a task with `id=`$\langle t\rangle$, then task $\langle rid\rangle$ in work package $\langle wp\rangle$ must be completed for task $\langle t\rangle$ to become possible. This key will draw an arrow into the gantt chart from the end of task $\langle rid\rangle$ to $\langle t\rangle$. Note that dependencies should always point forward in time. Furthermore, note that the fact that dependencies always go from the end of the source to the beginning of the target work phase is intentional, if this does not meet your needs, then you should probably break a work phase into pieces that can be addressed separately.
- `RM` `RAM` In single-site proposals, the `RM` (and `RAM` if the `RAM` option was given) keys are used to specify the estimated efforts to be expended on research and development in this work package. Both are specified in person months. `RM` is used for "researcher months" (wissenschaftlicher Mitarbeiter) and `RAM` for "research assistant months" (wissenschaftliche Hilfskraft).
- `*RM` `*RAM` In multi-site proposals, the `proposal` package generates the keys $\langle site\rangle$`RM` (and $\langle site\rangle$`RAM`) where $\langle site\rangle$ is any site label declared via the `site` key in the top-level `proposal` environment. This can be used to specify the person months that the site spends on this work package (the value for work groups is automatically computed (remember to run LaTeX twice for this)).
- `lead` In multi-site proposals the `lead` key specifies the work package or work group lead, the value of this feature should be the short name of the respective partner.

It is often useful to group the work packages in a proposal further (especially for larger, collab- `workarea` orative proposals). This can be done via the `workarea` environment, which groups work packages. This environment takes the same keys as the `workpackage` environment, except for the efforts, which can be computed automatically from the work packages it groups.

As the author of the `proposal` class likes more structured proposals, using work areas is the default, but the `proposal` class can also be used with the `noworkareas` option for less structured (smaller) proposals.

2.6 Tasks

`tasklist` `task` In the work packages we can list tasks that need to be undertaken with the `tasklist` environment. The individual tasks are marked up with the `task` environment. This takes a keyval argument with the keys `id` for identification, `title` for a title, and the workphase keys `wphrases`, `start`, `end`, and `force` (see Section 2.7). For planning involvement we can specify the overall person months via the `PM` key, the task lead via `lead`, and the partners involved via the `partners` key. Finally task dependencies can be specified via the `requires` key.

`\taskref` `\tasktref` `\localtaskref` Tasks can be referenced by the `\taskref` macro that takes two arguments: the work package identifier and the task identifier. As for work packages and work areas, there is a long reference variant with work package title: `\tasktref`. Finally, `\localtaskref` references a task in the local work package by the identifier in its argument.

2.7 Work Phase Metadata

`wphases` The `task` and `workpackage` allow the `wphases` key to specify the a list of work phases. The value of this key is comma-separated list of work phase specifications of the form ⟨*start*⟩`-`⟨*end*⟩ or ⟨*start*⟩`-`⟨*end*⟩`!`⟨*force*⟩, where ⟨*start*⟩ and ⟨*end*⟩ delimit the run time of the work phase and the optional `!`⟨*force*⟩ specifies the work force, i.e. the intensity of work as a number between 0 and 1. If no force is given, the default is 1. The main reason for specifying this metadata for tasks is to generate a Gantt chart (see Section 2.8).

2.8 Gantt Charts

`gantt` `xscale` `yscale` `step` `draft` Gantt charts are used in proposals to show the distribution of activities in work packages over time. A gantt chart is represented by the `gantt` environment that takes a on optional keyval argument. The keys `xscale` and `yscale` are used to specify a scale factors for the chart so that it fits on the page. The `step` key allows to specify the steps (in months) of the vertical auxiliary lines. Finally, the `draft` key specifies that plausibility checks (that can be expensive to run) are carried out. Note that the value does not have to be given, so `\begin{gantt}{draft,yscale=.5,step=3}` is a perfectly good invocation.

`\ganttchart` Usually, the `gantt` environment is not used however, since it is part of the macro that takes the same keys. This generates a whole Gantt chart automatically from the work phase specifications in the work packages. As above we have to run LaTeX two times for the work phases to show up.

2.9 Milestones and Deliverables

Many proposal formats foresee that project progress will be tracked in the form of *milestones* – points in the project, where a predefined state of affairs is reached – and *deliverables* – tangible project outcomes that have to be delivered. Correspondingly, milestones and deliverables have to be specified in the proposal and accounted for in the project reports. To facilitate this the `proposal` class and its instances provide a simple infrastructure for dealing with milestones and deliverables.

`milestones` `\milestone` `\milestone@label` `\mileref` `\miletref` Milestones are usually given in a special table[1], which we markup up with the `milestones` environment that takes care of initialization and numbering issues. This contains a list of milestone descriptions via the `\milestone` macro which is invoked as `\milestone[`⟨*keys*⟩`]{`⟨*title*⟩`}{`⟨*desc*⟩`}`, where ⟨*keys*⟩ supports the keys `id` for identification `month` for specifying the milestone date (in months of the project duration), and `verif` for specifying a means of verification[2] Milestones are numbered with labels whose shape can be customized by redefining `\milestone@label` and referenced by the `\mileref{`⟨*id*⟩`}` and `\miletref{`⟨*id*⟩`}` for a reference with milestone title. `\pdatacount{all}{miles}` gives the number of milestones.

[1]this is the default provided by the base `proposal` class, it can be specialized for proposal class instances by redefining the `@milestones` environment and correspondingly the `milestone` macro.

[2]Arguably, this set of keys is inspired by EU proposals, but can be extended in class instances.

Deliverables are usually defined as part of the work package descriptions (see Section 2.5) and listed in an overview table in a separate of the proposal. As for the milestones, we use an environment `wpdelivs` that contains the deliverable descriptions. These are marked up via the environment which takes an optional keyval argument for the deliverable metadata a regular argument for the title and contains the description of the deliverable as the body. For the metadata we have the keys `id` for the deliverable identifier, `due` for the target date (a number that denotes the project month), `nature` and `dissem` for specifying the deliverable nature and dissemination status (usually as short strings prescribed by the proposal template), and `miles` for the milestone this deliverable is targeted for (specified by the milestone identifier). For repeating deliverables (e.g. project reports), both `due` and `miles` can contain comma-separated lists. Deliverables are numbered by labels whose shape can be customized by number, where the shape of the label can be specified by redefining `\deliv@label` and referenced by `\delivref{`⟨*wp*⟩`}{`⟨*id*⟩`}` where ⟨*wp*⟩ is the work package identifier and ⟨*id*⟩ that if the deliverable and `\delivtref{`⟨*wp*⟩`}{`⟨*id*⟩`}` for a reference with title. `\pdatacount{`⟨*wp*⟩`}{delivs}` gives the number of milestones of the work package ⟨*wp*⟩ `\pdatacount{all}{delivs}` that of all deliverables (aggregating over all work packages).

wpdelivs

wpdeliv

\deliv@label

\delivref

\delivtref

Some proposal templates ask for an overview table of the deliverables which aggregates the deliverables of the respective work packages and areas ordered by due date. This can be generated with the `\inputdelivs` macro. This works index generation in LaTeX. The `wpdeliv` environment writes the deliverable data to a file ⟨*main*⟩`.delivs`, which can be processed externally (usually just sorting with `sort` in Unix is sufficient) into ⟨*main*⟩`.deliverables`, which is then input via the `\inputdelivs` macro.

\inputdelivs

In some proposals, also work areas can have deliverables, then the above hold analogously for `wpdelivs` and `wadeliv` environments.

wadelivs

wadeliv

Note that handling deliverables adds considerable overhead to proposal formatting and adds auxiliary files, so they are only activated if the `deliverables` option is given (see Section 2.1).

2.10 Referencing and Hyperlinking

The `proposal` package extends the hyperlinking provided by the `hyperref` package it includes to work packages, work groups, ... Whenever these are defined using the `proposal` infrastructure, the class saves the relevant information in the auxiliary file ⟨*proposal*⟩`.aux`. This information can be referenced via the `\pdataref` macro, which takes three arguments.

\pdataref

In a reference `\pdataref{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*aspect*⟩`}` the first argument ⟨*type*⟩ specifies the type of the object (currently one of `wp`, `wa`, and `partner`) to be referenced, ⟨*id*⟩ specifies the identifier of the referenced object (it matches the identifier given in the `id` key of the object), and ⟨*aspect*⟩ specifies the aspect of the saved information that is referenced.

For a partner ⟨*aspect*⟩ can be one of `number` (partner number), `short` (partner acronym), `long` (official partner name), `nationality` (partner nationality).

For a work package ⟨*aspect*⟩ can be `number`, (the work package number), `label` (the label **WP**n where n is the work package number for referencing), `title` (the work package title), `lead` the work package leader, `short` (a short version of the WP title for tables). For work groups we have the same aspects with analogous meanings. In all cases, the referenced information carries a hyperlink to the referenced object.

\pdataRef

The `\pdataRef` macro is a variant of `\pdataref` that also carries a hyperlink (if the `hyperref` package is loaded).

\pdatacount

The `\pdatacount` macro gives access to the numbers of certain aspects. For instance, the number of work packages in the proposal can be cited by `\pdatacount{all}{wp}`, similarly for work areas (if they are enabled), and finally, `\pdatacount{`⟨*wa*⟩`}{wp}` gives the number of work packages for a work area ⟨*wa*⟩. This is very useful for talking about work plans in a general way. Other objects that can be counted are deliverables (`\pdatacount{all}{deliverables}`) and milestones (`\pdatacount{all}{milestones}`).

Note that since the referencable information is written into the project data file ⟨*proposal*⟩`.pdata` file, it is available for forward references. However, it will only become available when the project data file is read, so the proposal has to be formatted twice for references to be correct.

Finally, the `proposal` package supplies specialized reference macros for work packages and areas. The `\WPref` macro takes a work package identifier as an argument and makes a reference: `\WPref{`⟨*id*⟩`}` abbreviates `\pdataRef{wp}{`⟨*id*⟩`}{label}`. The `\WPtref` macro is similar, but also prints out the (short) title: `\WPref{`⟨*id*⟩`}` abbreviates `\pdataRef{wp}{`⟨*id*⟩`}{label}: \pdataRef{wp}{`⟨*id*⟩`}{title}`. Unless the `noworkareas` macro is set, we also have the variants `\WAref` and `\WAtref` for work areas.

`\WPref` `\WPtref` `\WAref` `\WAtref`

2.11 Coherence

Many proposals require ways to show coherence between the partners. The `proposal` class offers the macro `\coherencematrix` for this which generates a matrix of symbols specifying joint publications and joint projects by the project partners that have been declared by the `\jointpub`, `\jointproj`, and `\jointorga` macros before. These macros all take a comma-separated list of site identifiers as an argument. Use for instance `\jointproj{a,b,c}` to specify that the sites with the identifiers `a`, `b` and `c` have a joint project. `\coherencetable` is a variant which packages the coherence table in a table figure with label `tab:collaboration`.

`\coherencematrix` `\jointpub` `\jointproj` `\jointorga` `\coherencetable`

The symbols used an be configured by redefining `\jpub`, `\jproj`, and `\jorga`.

`\jpub` `\jproj` `\jorga`

2.12 Localization

The `proposal` class offers some basic support for localization. This is still partial though, and I am not sure that this is the best way of setting things up. What I do is to define macros for all generated texts that can be redefined in the proposal classes that build in `proposal`. For instance the `dfgproposal` class [Koh14b] provides an option `german` for german-language proposals and project reports that triggers a redefinition of all of these macros at read time.

3 Limitations and Enhancements

The `proposal` is relatively early in its development, and many enhancements are conceivable. We will list them here.

1. macros cannot be used in work package and work area titles. They really mess up our `\wpfig` automation. The problem is that they are evaluated too early, and our trick with making them undefined while collecting the parts of the table-rows only works if we know which macros we may expect. We might specify all "allowable" macros in an optional key `protectmacro`, which is defined via

   ```
   \define@key{wpfig}{protectmacro}{\epandafter\let\csname #1\endcsname=\relax}
   ```

   But I am not sure that this will work.

2. It would be great, if in the Gantt Charts, we could include some plausibility checks (for draft = not `submit` mode). I can see two at the moment:

   - calculating the effort (i.e. the weight of the black area) and visualizing it. Then we could check whether that is larger than the effort declared for the work package.
   - calculating (and visualizing) the monthly effort. That should be kind of even (or it has to be explained in the positions requested).

3. we currently do not have a way to relate `PI`s to `site`s, but we do not really need to.

If you have other enhancements to propose or feel you can alleviate some limitation, please feel free to contact the author.

Acknowledgements

The author is indebted to Jake Hartenstein, Christoph Lange, Florian Rabe, Lutz Schröder, and Tsanko Tsankov for error reports, feature suggestions, and code snippets.

4 The Implementation

In this section we describe the implementation of the functionality of the `proposal` package.

4.1 Package Options and Format Initialization

We first set up the options for the package.

```
⟨*cls | reporting⟩
\newif\if@wpsubsection\@wpsubsectionfalse
\newif\ifsubmit\submitfalse
\newif\ifpublic\publicfalse
\newif\ifkeys\keysfalse
\newif\ifdelivs\delivsfalse
\newif\ifwork@areas\work@areastrue
\newif\if@RAM\@RAMfalse
\newif\if@svninfo\@svninfofalse
\newif\if@gitinfo\@gitinfofalse
\def\proposal@class{article}
\DeclareOption{wpsubsection}{\@wpsubsectiontrue}
\DeclareOption{submit}{\submittrue}
\DeclareOption{gitinfo}{\@gitinfotrue}
\DeclareOption{svninfo}{\@svninfotrue}
\DeclareOption{public}{\publictrue}
\DeclareOption{noworkareas}{\work@areasfalse\PassOptionsToClass{\CurrentOption}{pdata}}
\DeclareOption{RAM}{\@RAMtrue}
\DeclareOption{report}{\def\proposal@class{report}}
\DeclareOption{keys}{\keystrue}
\DeclareOption{deliverables}{\delivstrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
```

Then we load the packages we make use of

```
\LoadClass[a4paper,twoside]{\proposal@class}
\RequirePackage{amssymb}
\RequirePackage{url}
\RequirePackage{graphicx}
\RequirePackage{colortbl}
\RequirePackage{xcolor}
\RequirePackage{rotating}
\RequirePackage{fancyhdr}
\RequirePackage{array}
\RequirePackage{xspace}
\RequirePackage{comment}
\AtBeginDocument{\ifpublic\excludecomment{private}\fi}
\RequirePackage{tikz}
\RequirePackage{paralist}
\RequirePackage{a4wide}
\RequirePackage{boxedminipage}
% so that ednotes in wps do not run out of symbols
\renewcommand{\thempfootnote}{\roman{mpfootnote}}
\renewcommand{\familydefault}{\sfdefault}
\RequirePackage[scaled=.90]{helvet}
\RequirePackage{textcomp}
\RequirePackage[hyperref=auto,style=numeric,defernumbers=true,backend=bibtex,backref=true,firstinits=true,max
\RequirePackage{csquotes}
\RequirePackage{mdframed}
\RequirePackage{pdata}
```

in submit mode, we make the links a bit darker, so they print better.

```
\definecolor{darkblue}{rgb}{0,0,.7}
\ifsubmit\def\prop@link@color{darkblue}\else\def\prop@link@color{blue}\fi
\RequirePackage[bookmarks=true,linkcolor=\prop@link@color,
 citecolor=\prop@link@color,urlcolor=\prop@link@color,colorlinks=true,
 breaklinks=true, bookmarksopen=true]{hyperref}
```

the `ed` package [Koh14a] is very useful for collaborative writing and passing messages between collaborators or simply reminding yourself of editing tasks, so we preload it in the class. However, we only want to show the information in draft mode. Furthermore, we adapt the options for the `svninfo` and `gitinfo2` packages.

```
\ifsubmit
\RequirePackage[hide]{ed}
\if@svninfo\RequirePackage[final,today]{svninfo}\fi
\else
\RequirePackage[show]{ed}
\if@svninfo\RequirePackage[eso-foot,today]{svninfo}\fi
\if@gitinfo\RequirePackage[mark]{gitinfo2}\fi
\fi
\renewcommand\ednoteshape{\sl\footnotesize}
```

`private` We configure the `comment` package, so that it provides the `private` environment depending on the status of the `public` option.

```
\ifpublic\excludecomment{private}\else\includecomment{private}\fi
```

And we set up the appearance of the proposal. We want numbered subsubsections.

```
\setcounter{secnumdepth}{3}
```

We specify the page headings.

```
\newif\ifofpage\ofpagefalse
\fancyhead[RE,LO]{\prop@gen@acronym}
\fancyhfoffset{0pt}
\fancyfoot[C]{}
\newcommand\prop@of@pages[2]{page~#1\ifofpage~of~#2\fi}
\fancyhead[LE,RO]{\prop@of@pages\thepage{\pdataref@num{prop}{page}{last}}}
\pagestyle{fancyplain}
⟨/cls | reporting⟩
```

4.2 Proposal Metadata

`pdata` Most of the metadata functionality is encapsulated into the `pdata` package, which is shared by the proposal and report classes. `pdata.sty` first loads the `workaddress` package from sTeX and supplies the Euro symbol.

```
⟨*pdata⟩
\RequirePackage{workaddress}[2011/05/03]
\RequirePackage{eurosym}
```

We define the keys for metadata declarations in the `proposal` environment, they park their argument in an internal macro for use in the title page. The `site` key is the most complicated, so we take care of it first: We need a switch `\if@sites` that is set to true when the `site` key is used. Furthermore `site=`⟨*site*⟩ makes new keys ⟨*site*⟩`RM` and ⟨*site*⟩`RAM` (if the `RAM` option was set) for the `workpackage` environment and records the sites in the `\prop@gen@sites` token register.

```
\newif\if@sites\@sitesfalse\let\prop@gen@sites=\relax%
\newcounter{@site}%
\define@key{prop@gen}{site}{\@sitestrue\@dmp{site=#1}%
\stepcounter{@site}\pdata@def{site}{#1}{number}{\the@site}%
\@ifundefined{prop@gen@sites}{\xdef\prop@gen@sites{#1}}{\xdef\prop@gen@sites{\prop@gen@sites,#1}}%
\define@key{prop@gen}{#1RM}{\pdata@def{site}{#1}{intendedRM}{##1}}%
```

```
\if@RAM\define@key{prop@gen}{#1RAM}{\pdata@def{site}{#1}{intendedRAM}{##1}}\fi
\define@key{workpackage}{#1RM}{\pdata@def\wp@id{#1}{RM}{##1}}%
\if@RAM\define@key{workpackage}{#1RAM}{\pdata@def\wp@id{#1}{RAM}{##1}}\fi
\define@key{prop@gen}{#1employed}{{\let\tabularnewline\relax\let\hline\relax\let\wa@ref\relax%
\@ifundefined{prop@gen@employed@lines}%
{\xdef\prop@gen@employed@lines{\wa@ref{institution}{#1}{shortname} & ##1\tabularnewline\hline}}%
{\xdef\prop@gen@employed@lines{\prop@gen@employed@lines \wa@ref{institution}{#1}{shortname} & ##1\tabularnewl
```

If there are no sites, then we have to define keys `RM` and `RAM` that store the intended research (assistant months). Unfortunately, we cannot just include this in the `\if@sites` conditional here, since that is only set at runtime.

```
\define@key{prop@gen}{RM}{\@dmp{RM=#1}\if@sites%
\PackageWarning{Do not use the RM key in the presence of sites}\else%
\pdata@def{all}{intended}{RM}{#1}\fi}
\define@key{prop@gen}{RAM}{\@dmp{RAM=#1}\if@sites%
\PackageWarning{Do not use the RAM key in the presence of sites}\else%
\pdata@def{all}{intended}{RAM}{#1}\fi}
```

similarly, the `PI` keys are registered in `\prop@gen@PIs`.

```
\define@key{prop@gen}{PI}{\@dmp{PI=#1}%
\@ifundefined{prop@gen@PIs}{\xdef\prop@gen@PIs{#1}}{\xdef\prop@gen@PIs{\prop@gen@PIs,#1}}}
```

and the `pubspage` keys in `\prop@gen@pubspages`.

```
\define@key{prop@gen}{pubspage}{\@ifundefined{prop@gen@pubspages}%
{\xdef\prop@gen@pubspages{#1}}{\xdef\prop@gen@pubspages{\prop@gen@pubspages,#1}}}
```

the `importfrom` key reads the proposal data from its argument.

```
\define@key{prop@gen}{importfrom}{\message{importing proposal data from #1.pdata}\readpdata{#1}}
```

The rest of the keys just store their value.

```
\define@key{prop@gen}{instrument}{\def\prop@gen@instrument{#1}%
\pdata@def{prop}{gen}{instrument}{#1}\@dmp{inst=#1}}
\define@key{prop@gen}{title}{\def\prop@gen@title{#1}%
\pdata@def{prop}{gen}{title}{#1}}
\define@key{prop@gen}{acronym}{\gdef\prop@gen@acronym{#1}%
\pdata@def{prop}{gen}{acronym}{#1}\@dmp{acro=#1}}
\define@key{prop@gen}{acrolong}{\def\prop@gen@acrolong{#1}%
\pdata@def{prop}{gen}{acrolong}{#1}}
\define@key{prop@gen}{discipline}{\def\prop@gen@discipline{#1}%
\pdata@def{prop}{gen}{discipline}{#1}}
\define@key{prop@gen}{areas}{\def\prop@gen@areas{#1}%
\pdata@def{prop}{gen}{areas}{#1}}
\define@key{prop@gen}{start}{\def\prop@gen@start{#1}%
\pdata@def{prop}{gen}{start}{#1}}
\define@key{prop@gen}{months}{\def\prop@gen@months{#1}%
\pdata@def{prop}{gen}{months}{#1}}
\define@key{prop@gen}{since}{\def\prop@gen@since{#1}%
\pdata@def{prop}{gen}{since}{#1}}
\define@key{prop@gen}{totalduration}{\def\prop@gen@totalduration{#1}%
\pdata@def{prop}{gen}{totalduration}{#1}}
\define@key{prop@gen}{fundsuntil}{\def\prop@gen@fundsuntil{#1}%
\pdata@def{prop}{gen}{fundsuntil}{#1}}
```

and the default values, these will be used, if the author does not specify something better.

```
\newcommand\prop@gen@acro@default{ACRONYM}
\def\prop@gen@acro{\prop@gen@acro@default}
\newcommand\prop@gen@months@default{???months???}
\def\prop@gen@months{\prop@gen@months@default}
\newcommand\prop@gen@title@default{???Proposal Title???}
\def\prop@gen@title{\prop@gen@title@default}
```

```
\newcommand\prop@gen@instrument@default{??? Instrument ???}
\def\prop@gen@instrument{\prop@gen@instrument@default}
```

`\prop@tl` An auxiliary macro that is handy for making tables of WorkAddress data.

```
\newcommand\prop@tl[2]{\xdef\tab@line{}
\@for\tl@ext:={#1}\do{\xdef\tab@line{\tab@line&#2}}
\tab@line}
```

4.3 Proposal Appearance

We define the keys for the proposal appearance

```
\def\prop@gen@compactht{false}
\define@key{prop@gen}{compactht}[true]{\def\prop@gen@compactht{#1}}
⟨/pdata⟩
```

`emphbox`

```
⟨*cls⟩
\newmdenv[settings=\large]{emphbox}
```

4.4 Title Page

`prop@proposal` This internal environment is called in the `proposal` environment from the `proposal` class. The implementation here is only a stub to be substituted in a specialized class.

```
\newenvironment{prop@proposal}
{\thispagestyle{empty}%
\begin{center}
  {\LARGE \prop@gen@instrument}\\[.2cm]
  {\LARGE\textbf{\prop@gen@title}}\\[.3cm]
  {\LARGE Acronym: {\prop@gen@acronym}}\\[.2cm]
  {\large\today}\\[1em]
  \begin{tabular}{c*{\the@PIs}{c}}
    \prop@tl\prop@gen@PIs{\wa@ref{person}\tl@ext{name}}\\
    \prop@tl\prop@gen@PIs{\wa@ref{institution}{\wa@ref{person}\tl@ext{affiliation}}{name}}
 \end{tabular}\\[2cm]
\end{center}
\setcounter{tocdepth}{2}\tableofcontents\newpage\setcounter{page}{1}}
```

Now we come to the end of the environment:

```
{\section{List of Attachments}
\begin{itemize}
\@for\@I:=\prop@gen@PIs\do{%
\item Curriculum Vitae and list of publications for
  \wa@ref{person}\@I{personaltitle} \wa@ref{person}\@I{name}
\end{itemize}}\newpage
\printbibliography[heading=warnpubs]}
```

`proposal` The `proposal` environment reads the metadata keys defined above, and if there were no `site` keys, then it defines keys `RM` and `RAM` (unless the `noRAM` package option was given) for the `workpackage` environment. Also it reads the project data file and opens up the project data file `\pdata@out`, which it also closes at the end.

The environment calls an internal version of the environment `prop@proposal` that can be customized by the specializing classes.

```
\newenvironment{proposal}[1][]{\readpdata\jobname
\ofpagetrue\setkeys{prop@gen}{#1}
\pdata@open\jobname
\if@sites\else
\define@key{workpackage}{RM}{\pdata@def{wp}\wp@id{RM}{##1}\@dmp{RM=##1}}
\if@RAM\define@key{workpackage}{RAM}{\pdata@def{wp}\wp@id{RAM}{##1}\@dmp{RAM=##1}}\fi
\fi
\newcounter{@PIs}
\@ifundefined{prop@gen@PIs}{}{\@for\@I:=\prop@gen@PIs\do{\stepcounter{@PIs}}}
\newcounter{@sites}
\@ifundefined{prop@gen@sites}{}{\@for\@I:=\prop@gen@sites\do{\stepcounter{@sites}}}
\setcounter{page}{0}
\begin{prop@proposal}}
```

Now we come to the end of the environment, we take care of the last page and print the references.

```
{\end{prop@proposal}
\pdata@def{prop}{page}{last}{\thepage}\ofpagefalse
\pdata@close}
⟨/cls⟩
```

The `report` environment is similar, but somewhat simpler

`report`

```
⟨*reporting⟩
\newif\if@report\@reportfalse
\newenvironment{report}[1][]%
{\@reporttrue\readpdata\jobname%
\ofpagetrue\setkeys{prop@gen}{#1}%
\pdata@open\jobname%
\@ifundefined{prop@gen@PIs}{}{\newcounter{@PIs}\@for\@I:=\prop@gen@PIs\do{\stepcounter{@PIs}}}%
\@ifundefined{prop@gen@sites}{}{\newcounter{@sites}\@for\@I:=\prop@gen@sites\do{\stepcounter{@sites}}}%
\setcounter{page}{0}%
\begin{prop@report}}
{\end{prop@report}%
\pdata@def{prop}{page}{last}{\thepage}\ofpagefalse\newpage
\printbibliography[heading=warnpubs]
\pdata@close}
```

`prop@report`

```
\newenvironment{prop@report}
{\begin{center}
  {\LARGE Final Project Report}\\[.2cm]
  {\LARGE\textbf{\prop@gen@title}}\\[.3cm]
  {\LARGE Acronym: {\prop@gen@acronym}}\\[.2cm]
  {\large\today}\\[1em]
  \begin{tabular}{c*{\the@PIs}{c}}
    \prop@tl\prop@gen@PIs{\wa@ref{person}\tl@ext{name}}\\
    \prop@tl\prop@gen@PIs{\wa@ref{institution}{\wa@ref{person}\tl@ext{affiliation}}{name}}
 \end{tabular}\\[2cm]
\end{center}
\setcounter{tocdepth}{2}\tableofcontents\newpage\setcounter{page}{1}}
{}
⟨/reporting⟩
```

`\site*`

```
⟨*cls⟩
\newcommand\site[1]{\hyperlink{site@#1@target}{\wa@ref{institution}{#1}{acronym}}}
\newcommand\sitename[1]{\hyperlink{site@#1@target}{\wa@ref{institution}{#1}{name}}}
```

4.5 Objectives

We first define a presentation macro for objectives

`\objective@label`

```
\newcommand\objective@label[1]{O#1}
```

We define the keys for the objectives environment

```
\define@key{obj}{id}{\def\obj@id{#1}\@dmp{id=#1}}
\define@key{obj}{title}{\def\obj@title{#1}\@dmp{title=#1}}
\define@key{obj}{short}{\def\obj@short{#1}\@dmp{short=#1}}
```

And a counter for numbering objectives

```
\newcounter{objective}
```

`objective`

```
\newenvironment{objective}[1][]
{\let\obj@id\relax\let\obj@title\relax\let\obj@short\relax%
\setkeys{obj}{#1}\stepcounter{objective}%
\goodbreak\smallskip\par\noindent%
\textbf{\objective@label{\arabic{objective}}:%
~\pdata@target{obj}{\obj@id}{\pdataref{obj}{\obj@id}{title}}\ignorespaces}%
\pdata@def{obj}\obj@id{label}{\objective@label\theobjective}%
\@ifundefined{obj@title}{}{\pdata@def{obj}\obj@id{title}\obj@title}%
\@ifundefined{obj@short}{}{\pdata@def{obj}\obj@id{short}\obj@short}}
{}
```

`\OBJref`

```
\newcommand\OBJref[1]{\pdataRef{obj}{#1}{label}}
\newcommand\OBJtref[1]{\pdataRef{obj}{#1}{label}: \pdataRef{obj}{#1}{title}}
```

4.6 Work Packages and Work Groups

We first define keys for work groups (if we are in an IP).

```
\ifwork@areas
\define@key{workarea}{id}{\def\wa@id{#1}\@dmp{id=#1}}
\define@key{workarea}{title}{\pdata@def{wa}\wa@id{title}{#1}}
\define@key{workarea}{short}{\pdata@def{wa}\wa@id{short}{#1}}
\define@key{workarea}{lead}{\pdata@def{wa}\wa@id{lead}{#1}}
\fi
```

work packages have similar ones.

```
\define@key{workpackage}{id}{\def\wp@id{#1}\@dmp{id=#1}}
\define@key{workpackage}{title}{\pdata@def{wp}\wp@id{title}{#1}}
\define@key{workpackage}{lead}{\pdata@def{wp}\wp@id{lead}{#1}\def\wp@lead{#1}\@dmp{lead=#1}}
\define@key{workpackage}{short}{\pdata@def{wp}\wp@id{short}{#1}}
\define@key{workpackage}{type}{\def\wp@type{#1}\pdata@def{wp}\wp@id{type}{#1}}
\define@key{workpackage}{wphases}{\def\wp@wphases{#1}\pdata@def{wp}\wp@id{wphases}{#1}}
\define@key{workpackage}{rotate}[true]{\def\wp@rotate{#1}}
```

We define the constructors for the work package and work group labels and titles.

```
\newcommand\wp@mk@title[1]{Work Package {#1}}
\newcommand\wp@label[1]{WP{#1}}
\ifwork@areas
\newcommand\wa@label[1]{WA{#1}}
\newcommand\wa@mk@title[1]{Work Area {#1}}
\fi
```

The `wa` and `wp` counters are for the work packages and work groups, the counter `deliv` for deliverables.

```
\ifwork@areas\newcounter{wa}\newcounter{wp}[wa]\else\newcounter{wp}\fi
\ifdelivs\newcounter{deliv}[wp]\fi
\newcounter{allwp}
```

`\update@*` update the list `\@wps` of the work packages in the local group and the list `\@was` work groups for the staff efforts table: if `\@wps` is undefined, then initialize the comma-separated list, otherwise extend it.[3] EdN:3

```
\newcommand\update@wps[1]{\@ifundefined{@wps}{\xdef\@wps{#1}}{\xdef\@wps{\@wps,#1}}}
\newcommand\update@tasks[1]{\@ifundefined{@tasks}{\xdef\@tasks{#1}}{\xdef\@tasks{\@tasks,#1}}}
\newcommand\update@deps[1]{\@ifundefined{task@deps}{\xdef\task@deps{#1}}{\xdef\task@deps{\task@deps,#1}}}
\ifwork@areas\def\update@was#1{\@ifundefined{@was}{\xdef\@was{#1}}{\xdef\@was{\@was,#1}}}\fi
```

`\decode@wphase` `\decode@wphase` decodes a string of the form ⟨*start*⟩-⟨*end*⟩!⟨*force*⟩ and defines the macros `\wphase@start`, `\wphase@end`, and `\wphase@force` with the three parts and also computes `\wphase@len`. The intermediate parsing macro `\decode@p@start` parses out the start (a number), and passes on to `\decode@p@end`, which parses out the end (another number) and the force string, which is either empty (if the !⟨*force*⟩ part is omitted) or of the form !⟨*force*⟩. In the first case the default value 1 is returned for `\decode@force` in the second ⟨*force*⟩.

```
\newcommand\decode@wphase[1]{\expandafter\decode@p@start#1@%
\local@count\wphase@end\advance\local@count by -\wphase@start%
\def\wphase@len{\the\local@count}}
\def\decode@p@start#1-#2@{\def\wphase@start{#1}\decode@p@end#2!@}
\def\decode@p@end#1!#2@{\def\wphase@end{#1}\def\@test{#2}%
\ifx\@test\@empty\def\wphase@force{1}\else\decode@p@force#2\fi}
\def\decode@p@force#1!{\def\wphase@force{#1}}
```

`\startend@wphases` We first iteratively decode the work phases, so that the last definition of `\wphase@end` remains, then we parse out the start of the first workphase to define `\wphase@start`

```
\def\wphases@start#1-#2@{\def\wphase@start{#1}}
\newcommand\startend@wphases[1]{\def\@test{#1}
\ifx\@test\@empty\def\wphase@start{0}\def\wphase@end{0}\else%
\@for\@I:=#1\do{\expandafter\decode@p@start\@I @}
\expandafter\wphases@start#1@\fi}
```

with these it is now relatively simple to define the interface macros.

`work@package` The `workpackage` environment collects the keywords, steps the counters, writes the metadata to the aux file, updates the work packages in the local group, generates the work package number `\wp@num`.

```
\newcounter{wp@RM}
\if@RAM\newcounter{wp@RAM}\fi
\newenvironment{work@package}[1][]%
{\def\wp@wphases{0-0}% default values
\def\wp@rotate{false}
\setkeys{workpackage}{#1}\stepcounter{wp}\stepcounter{allwp}%
\startend@wphases\wp@wphases%
\pdata@def{wp}\wp@id{start}\wphase@start\pdata@def{wp}\wp@id{end}\wphase@end%
\@ifundefined{wp@type}{}{\pdata@def{wp}\wp@id{type}\wp@type}%
\let\@tasks=\relax%
\edef\wp@num{\ifwork@areas\thewa.\fi\thewp}%
\pdata@def{wp}\wp@id{label}{\wp@label\wp@num}%
\pdata@def{wp}\wp@id{number}{\thewp}%
\pdata@def{wp}\wp@id{page}{\thepage}%
\update@wps\wp@id%
\edef\wp@num{\ifwork@areas\thewa.\fi\thewp}%
\pdata@def{wp}{\wp@id}{num}{\thewp}%
```

If we have sites, we have to compute the total RM and RAM for this WP.

```
\if@sites%
```

[3]EdNote: with the current architecture, we cannot have work areas that do not contain work packages, this leads to the error that wps is undefined in endworkplan

```
\setcounter{wp@RM}{0}\if@RAM\setcounter{wp@RAM}{0}\fi%
\@for\@site:=\prop@gen@sites\do{%
\edef\@RM{\pdataref@num\wp@id\@site{RM}}\addtocounter{wp@RM}{\@RM}%
\if@RAM\edef\@RAM{\pdataref@num\wp@id\@site{RAM}}\addtocounter{wp@RAM}{\@RAM}\fi}
\pdata@def{wp}\wp@id{RM}{\thewp@RM}%
\if@RAM\pdata@def{wp}\wp@id{RAM}{\thewp@RAM}\fi%
\fi}% if@sites
{\@ifundefined{@tasks}{}{\pdata@def{\wp@id}{task}{ids}\@tasks}}
```

`workpackage` With this, it becomes simple to define a work package environment. We consider two cases, if we have sites, then we make a header table. If not, we can make things much simpler: we just generate a subsection

```
\newenvironment{workpackage}[1][]%
{\begin{work@package}[#1]%
%\if@wpsubsection\subsubsection*{{\wp@mk@title\thewp}: \pdataref{wp}\wp@id{title}}\fi
\if@sites\goodbreak\medskip\wpheadertable%
\else\subsubsection*{{\wptitle} (\wprm)}\fi%
\addcontentsline{toc}{paragraph}{{\wp@mk@title\thewp}: \pdataref{wp}\wp@id{title}}%
\noindent\ignorespaces}
{\end{work@package}}
```

EdN:4 `\wptitle` [4]

```
\newcommand\wptitle{\wp@mk@title{\wp@num}: \pdata@target{wp}{\wp@id}{\pdataref{wp}\wp@id{title}}}
```

EdN:5 `\wprm` [5]

```
\newcommand\wprm{\pdataref@safe{wp}\wp@id{RM}\if@RAM\ RM+\pdataref{wp}\wp@id{RAM} RAM\fi}
```

`@site@contributes` Called as `\if@site@contributes{`⟨*site*⟩`}{`⟨*tokens*⟩`}` the following happens: If `\prop@gen@compactht` is `\@true` (set by the `compactht` attribute on the `proposal` environment), then ⟨*tokens*⟩ is processed. Otherwise, ⟨*tokens*⟩ is only processed if ⟨*site*⟩ contributes to the current work package (i.e the `RM` $\neq 0$ and `RAM` $\neq 0$)

```
\newcount\site@contribution%
\newcommand\if@site@contributes[2]{%
\ifx\prop@gen@compactht\@true
\if@RAM\ifnum\pdataref@num\wp@id{#1}{RM} > 0 \ifnum \pdataref@num\wp@id{#1}{RAM} > 0 #2\fi\fi
\else\ifnum\pdataref@num\wp@id{#1}{RM} > 0 #2\fi\fi
\else #2\fi}
```

`\wp@sites@line` `\wp@efforts@line` `\wp@sites@num` The following macro computes the sites line (in the token register `\wp@sites@line`), the efforts line (in `\wp@efforts@line`), and the sites number (in the counter `\sites@num`) for later inclusion in the `\wpheadertable`. If `\prop@gen@compactht` is `\@true`, then no sites without contributions are listed in the table.

```
\newcounter{wp@sites@num}
\newcommand\wp@sites@efforts@lines{%
\setcounter{wp@sites@num}{0}
{\let\G@refundefinedtrue=\relax\let\@latex@warning=\relax\let\@sw\relax%
\let\site\relax\let\textbf\relax\let\sum@style\relax\let\lead@style\relax%
\let\pn\relax\let\sys\relax%
\xdef\wp@sites@line{\wp@legend@site}\xdef\wp@efforts@line{\wp@legend@effort}%initialize lines
\@for\@site:=\prop@gen@sites\do{\if@site@contributes\@site{\stepcounter{wp@sites@num}}%
\xdef\wp@sites@line{\wp@sites@line%
\if@site@contributes\@site{&%
\ifx\wp@rotate\@true%
\@sw{\ifx\@site\wp@lead\lead@style{\site{\@site}}\else\site{\@site}\fi}%
\else\ifx\@site\wp@lead\lead@style{\site{\@site}}\else\site{\@site}\fi%
```

[4]EdNote: document above
[5]EdNote: document above

```
\fi}}%
\xdef\wp@efforts@line{\wp@efforts@line%
\if@site@contributes\@site{&%
\ifx\@site\wp@lead%
\lead@style{\pdataref@safe\wp@id\@site{RM}\if@RAM+\pdataref@safe\wp@id\@site{RAM}\fi}
\else\pdataref@safe\wp@id\@site{RM}\if@RAM+\pdataref@safe\wp@id\@site{RAM}\fi\fi}}%
}% do
\xdef\wp@sites@line{\wp@sites@line&\sum@style{\wp@legend@all}}%
\xdef\wp@efforts@line{\wp@efforts@line&
\sum@style{\textbf{\pdataref{wp}\wp@id{RM}\if@RAM+\pdataref{wp}\wp@id{RAM}\fi}}}}}
```

`\wpheadertable` This macro computes the default work package header table, if there are sites.

```
\newcommand\wpheadertable{%
\wp@sites@efforts@lines%
\par\noindent\begin{tabular}{|l||l|*{\thewp@sites@num}{c|}|c|}\hline%
\textbf{\wp@mk@title{\wp@num}}&\wp@sites@line\\\hline%
\textsf{\pdata@target{wp}{\wp@id}{\pdataref{wp}\wp@id{title}}} &\wp@efforts@line\\\hline%
\end{tabular}\smallskip\par\noindent\ignorespaces}
```

and now multilinguality support

```
\newcommand\wp@legend@site{Site}
\newcommand\wp@legend@effort{Effort\if@RAM{ (RM+RAM)}\fi}
\newcommand\wp@legend@all{\textbf{all}}
```

`workarea` the `workarea` environment for work groups is almost the same, but we also have to initialize the work package counters. Also, the efforts can be computed from the work packages in this group via the `wa@effort` counter

```
\newcounter{prop@RM}\if@RAM\newcounter{prop@RAM}\fi
\ifwork@areas
\newcounter{wa@RM}\if@RAM\newcounter{wa@RAM}\fi\newcounter{wa@wps}
\newenvironment{workarea}[1][]
{\setkeys{workarea}{#1}
\let\@wps=\relax
\stepcounter{wa}
\pdata@def{wa}{\wa@id}{label}{\wa@label\thewa}
\pdata@def{wa}{\wa@id}{number}{\thewa}
\pdata@def{wa}{\wa@id}{page}{\thepage}
\update@was{\wa@id}
\pdata@def{wa}{\wa@id}{num}{\thewa}
\setcounter{wa@RM}{0}\if@RAM\setcounter{wa@RAM}{0}\fi\setcounter{wa@wps}{0}
\edef\@@wps{\pdataref@aux\wa@id{wp}{ids}}
\@for\@wp:=\@@wps\do{\stepcounter{wa@wps}%
\if@sites
\@for\@site:=\prop@gen@sites\do{%
  \edef\@RM{\pdataref@num\@wp\@site{RM}}
  \if@RAM\edef\@RAM{\pdataref@num\@wp\@site{RAM}}\fi
  \addtocounter{wa@RM}{\@RM}\addtocounter{prop@RM}{\@RM}
  \if@RAM\addtocounter{wa@RAM}{\@RAM}\addtocounter{prop@RAM}{\@RAM}\fi}
\else
\edef\@RM{\pdataref@num{wp}\@wp{RM}}
\if@RAM\edef\@RAM{\pdataref@num{wp}\@wp{RAM}}\fi
\addtocounter{wa@RM}{\@RM}\addtocounter{prop@RM}{\@RM}
\if@RAM\addtocounter{wa@RAM}{\@RAM}\addtocounter{prop@RAM}{\@RAM}\fi
\fi}
\pdata@def{wa}\wa@id{RM}\thewa@RM
\pdata@def{prop}{all}{RM}\theprop@RM
\if@RAM
\pdata@def{wa}\wa@id{RAM}\thewa@RAM
\pdata@def{prop}{all}{RAM}\theprop@RAM
\fi
\subsubsection*{{\wa@mk@title\thewa}: {\pdata@target{wa}\wa@id{\pdataref{wa}\wa@id{title}}}}
\addcontentsline{toc}{subsubsection}{{\wa@mk@title\thewa}: \pdataref{wa}\wa@id{title}}%
\ignorespaces}
{\@ifundefined{@wps}{}{\pdata@def\wa@id{wp}{ids}\@wps}\pdata@def\wa@id{wp}{count}\thewa@wps}\fi
```

`workplan` The `workplan` environment sets up the accumulator macros `\@wps`, `\@was`, for the collecting the identifiers of work packages and work groups. At the end of the workplan description it writes out their content to the aux file for reference.

```
\ifdelivs\newwrite\wpg@delivs\fi
\newenvironment{workplan}%
{\ifdelivs\immediate\openout\wpg@delivs=\jobname.delivs\fi
\ifwork@areas\let\@was=\relax\else\let\@wps=\relax\fi}%
{\@ifundefined{task@deps}{}{\pdata@def{all}{task}{deps}{\task@deps}}
\pdata@def{all}{task}{count}{\thealltasks}
\ifwork@areas
\@ifundefined{@was}{}{\pdata@def{all}{wa}{ids}\@was}
\else
\@ifundefined{@wps}{}{\pdata@def{all}{wp}{ids}\@wps}
\fi
\ifdelivs\@ifundefined{mile@stones}{}
{\@for\@I:=\mile@stones\do{%
\pdata@def{mile}\@I{delivs}{\@ifundefined{\@I delivs}{}{\csname\@I delivs\endcsname}}}}\fi
\ifwork@areas\pdata@def{all}{wa}{count}{\thewa}\fi
\pdata@def{all}{wp}{count}{\theallwp}
\ifdelivs
\pdata@def{all}{deliverables}{count}{\thedeliverable}
\pdata@def{all}{milestones}{count}{\themilestone}
\fi
\ifdelivs\closeout\wpg@delivs\fi}
```

4.7 Milestones and Deliverables

`deliv@error` this macro raises an error if deliverable commands are used without the `deliverables` option being set.

```
\newcommand\deliv@error{\PackageError{proposal}
{To use use deliverables, you have to specify the option 'deliverables'}}
```

`wpdelivs`

```
\newenvironment{wpdelivs}{\begin{wp@delivs}}{\end{wp@delivs}}
```

`wp@delivs`

```
\newenvironment{wp@delivs}
{\ifdelivs\textbf\deliv@legend@delivs:\\[-3ex]%
\begin{compactdesc}\else\deliv@error\fi}
{\ifdelivs\end{compactdesc}\fi}
```

and now multilinguality support

```
\newcommand\deliv@legend@delivs{Deliverables}
```

`\wadelivs`

```
\newenvironment{wadelivs}
{\textbf\deliv@legend@delivs:\\[-3ex]\begin{wp@delivs}}
{\end{wp@delivs}}
```

`\lec` This macro is generally useful to put a comment at the end of the line, possibly making a new one if there is not enough space.

```
\newcommand\lec[1]{\strut\hfil\strut\null\nobreak\hfill\hbox{$\leadsto$#1}\par}
```

\deliv@label

```
\newcommand\deliv@label[1]{D{#1}}
```

\delivref This macro is generally useful to put a comment at the end of the line, possibly making a new one if there is not enough space.

```
\newcommand\delivref[2]{\pdataRef{deliv}{#1#2}{label}}
\newcommand\delivtref[2]{\pdataRef{deliv}{#1#2}{label}: \pdataRef{deliv}{#1#2}{short}}
```

\wpg@deliv We first define the keys

```
\define@key{deliv}{id}{\def\deliv@id{#1}}
\define@key{deliv}{due}{\def\deliv@due{#1}}
\define@key{deliv}{dissem}{\def\deliv@dissem{#1}}
\define@key{deliv}{nature}{\def\deliv@nature{#1}}
\define@key{deliv}{miles}{\def\deliv@miles{#1}}
\define@key{deliv}{short}{\def\deliv@short{#1}}
```

The `\wpdeliv` macro cycles over the due dates and generates the relevant entries into the deliverables file. The first step is to write the general metadata to the pdata file.

```
\newcounter{deliverable}
\newcommand{\wpg@deliv}[3]{% keys, title, type
\stepcounter{deliverable}
\let\deliv@miles=\relax% clean state
\def\@type{#3}\def\@wp{wp}% set up ifx
\def\wpg@id{\csname #3@id\endcsname}
\setkeys{deliv}{#1}\stepcounter{deliv}% set state
\ifx\@type\@wp\def\current@label{\deliv@label{\ifwork@areas\thewa.\fi\thewp.\thedeliv}}
\else\def\current@label{\deliv@label{\thewa.\thedeliv}}\fi
\pdata@def{deliv}{\wpg@id\deliv@id}{label}{\current@label}
\pdata@def{deliv}{\wpg@id\deliv@id}{title}{#2}
\@ifundefined{deliv@short}
{\pdata@def{deliv}{\wpg@id\deliv@id}{short}{#2}}
{\pdata@def{deliv}{\wpg@id\deliv@id}{short}{\deliv@short}}
\pdata@def{deliv}{\wpg@id\deliv@id}{nature}{\deliv@nature}
\pdata@def{deliv}{\wpg@id\deliv@id}{dissem}{\deliv@dissem}
```

Then we iterate over the due dates and generate an entry for teach of them.

```
\@ifundefined{deliv@due}{}{%
\@for\@I:=\deliv@due\do{\protected@write\wpg@delivs{}{\string\deliverable%
{\ifnum\@I<10 0\@I\else\@I\fi}% sort key
{\@I}% due date
{\current@label}% label
{\@ifundefined{deliv@id}{\protect\G@refundefinedtrue\@latex@warning{key 'id' for Deliv #1
      undefined}??}{\wpg@id\deliv@id}}% id
{\@ifundefined{deliv@dissem}{\protect\G@refundefinedtrue\@latex@warning{key 'dissem' for
      Deliv #1 undefined}??}{\deliv@dissem}}% dissemination level
{\@ifundefined{deliv@nature}{\protect\G@refundefinedtrue\@latex@warning{key 'nature' for Deliv
      #1 undefined}??}{\deliv@nature}}% nature
{#2}
{\ifx\@type\@wp{WP\ifwork@areas\thewa.\fi\thewp}\else{WA\thewa}\fi}}}%WP
```

And finally, we generate the entry into the deliverables table.

```
\item[\current@label: (Month \deliv@due; nature: \deliv@nature, dissem.: \deliv@dissem)] \pdata@target{deliv}
\@ifundefined{deliv@miles}{}{% print the milestones and update their deliverables
\let\m@sep=\relax% do not print the separator the first time round
\lec{\@for\@I:=\deliv@miles\do{% Iterate over the milestones mentioned
\m@sep\pdataRef{mile}{\@I}{label}% print the milestone reference
\let\m@sep=,}}%set the separator for the next times
\def\d@sep{,}
 \@for\@I:=\deliv@miles\do{% Iterate over the milestones mentioned
\expandafter\ifx\csname\@I delivs\endcsname\relax% Check that the miles@delivs is empty
 {\expandafter\xdef\csname\@I delivs\endcsname{\wpg@id\deliv@id}}% if so, skip the separator
  \else\expandafter\xdef\csname\@I delivs\endcsname%if not add it
     {\csname\@I delivs\endcsname\d@sep\wpg@id\deliv@id}\fi}}
```

Now, we only need to instantiate

`wadeliv`

```
\newenvironment{wadeliv}[2][]{\ifdelivs\wpg@deliv{#1}{#2}{wa}\else\deliv@error\fi}{}
```

`wpdeliv`

```
\newenvironment{wpdeliv}[2][]{\ifdelivs\wpg@deliv{#1}{#2}{wp}\else\deliv@error\fi}{}
```

`\milestone@label`

```
\newcommand\milestone@label[1]{M{#1}}
```

`\mileref` This macro is generally useful to put a comment at the end of the line, possibly making a new one if there is not enough space.

```
\newcommand\mileref[1]{\pdataRef{mile}{#1}{label}}
\newcommand\miletref[1]{\pdataRef{mile}{#1}{label}: \pdataRef{mile}{#1}{short}}
```

`\milestone` create a new milestone, initialize its deliverables accumulator macro, set up hyperlinking, and extend the milestones list.

```
\newcounter{milestone}
\define@key{milestone}{id}{\gdef\mile@id{#1}}
\define@key{milestone}{month}{\gdef\mile@month{#1}}
\define@key{milestone}{verif}{\gdef\mile@verif{#1}}
\newcommand\milestone[3][]{%
\ifdelivs%
\setkeys{milestone}{#1}\stepcounter{milestone}%
\pdata@def{mile}\mile@id{label}{\milestone@label{\themilestone}}%
\pdata@def{mile}\mile@id{month}{\mile@month}%
\pdata@def{mile}\mile@id{verif}{\mile@verif}%
\pdata@def{mile}\mile@id{title}{#2}%
\@ifundefined{mile@stones}{\xdef\mile@stones{\mile@id}}{\xdef\mile@stones{\mile@stones,\mile@id}}%
\@milestone{#1}{#2}{#3}% presentation
\else\deliv@error\fi}
```

`\@milestone` the corresponding presentation macro.

```
\newcommand\@milestone[3]{%
\pdata@target{mile}\mile@id{\textbf{\milestone@label\themilestone}}&
\textbf{#2} &
\prop@milesfor\mile@id &
\pdataref{mile}\mile@id{month} &
\pdataref{mile}\mile@id{verif}\\\hline
\multicolumn{5}{|p{14cm}|}{#3}\\\hline\hline}
```

`milestones`

```
\newenvironment{milestones}{\begin{@milestones}}{\end{@milestones}}
```

`@milestones`

```
\newenvironment{@milestones}
{\ifdelivs\begin{longtable}{|l|p{4cm}|p{5cm}|l|p{2.5cm}|}\hline
\#&\miles@legend@name&\miles@legend@involved&\miles@legend@month&\miles@legend@verif\\\hline\hline%
\else\deliv@error\fi}
{\ifdelivs\end{longtable}%
\footnotetext\miles@legend@footnote\fi}
```

now the multilinguality support

```
\newcommand\miles@legend@name{Name}
\newcommand\miles@legend@month{Mo}
\newcommand\miles@legend@verif{Means of Verif.}
\newcommand\miles@legend@involved{WPs\footnotemark/Deliverables involved}
\newcommand\miles@legend@footnote{The work package number is the first number in the deliverable number.}
```

`\prop@milesfor` the due date is the first argument to facilitate sorting

```
\newcommand\prop@milesfor[1]{\edef\@delivs{\pdataref@safe{mile}{#1}{delivs}}%
\let\m@sep=\relax\def\new@sep{,\ }%
\@for\@I:=\@delivs\do{\m@sep\pdataRef{deliv}\@I{label}\let\m@sep=\new@sep}}
```

`\deliverable` the first argument is an extended due date to facilitate sorting.

```
\newcommand{\deliverable}[8]{\pdataRef{deliv}{#4}{label}&#7&#8&#6&#5&#2\\\hline}%sortkey,due,label,id,title,t
```

`deliverables`

```
\newenvironment{deliverables}[1]{\ifdelivs\begin{longtable}{|l|p{#1}|l|l|l|l|}\hline
\#&\delivs@legend@name&\delivs@legend@wp&\delivs@legend@nature&
\delivs@legend@level&\delivs@legend@due\\\hline\hline\else\deliv@error\fi}
{\ifdelivs\end{longtable}\fi}
```

now the multilingual support

```
\newcommand\delivs@legend@name{Deliverable name}
\newcommand\delivs@legend@wp{WP}
\newcommand\delivs@legend@nature{Nature}
\newcommand\delivs@legend@level{Level}
\newcommand\delivs@legend@due{Due}
```

`\inputdelivs`

```
\newcommand{\inputdelivs}[1]{%
\begin{deliverables}{#1}%
\IfFileExists{\jobname.deliverables}%
{\input{\jobname.deliverables}}%
{\IfFileExists{\jobname.delivs}{\input{\jobname.delivs}}{}}
\end{deliverables}}
```

4.8 Tasks and Work Phases

`tasklist`

```
\newenvironment{tasklist}
{\begin{compactenum}}{\end{compactenum}}
```

The next step is to

```
\newcommand\task@label[1]{T#1}
```

We define the keys for the task macro

```
\define@key{task}{id}{\def\task@id{#1}\@dmp{id=#1}}
\define@key{task}{wphases}{\def\task@wphases{#1}\pdata@def{task}{\taskin\task@id\wp@id}{wphases}{#1}\@dmp{wph
\define@key{task}{requires}{\@requires\task@id{#1}\@dmp{req=#1}}
\define@key{task}{title}{\def\task@title{#1}\pdata@def{task}{\taskin\task@id\wp@id}{title}{#1}}
\define@key{task}{lead}{\def\task@lead{#1}\pdata@def{task}{\taskin\task@id\wp@id}{lead}{#1}\@dmp{lead=#1}}
\define@key{task}{partners}{\def\task@partners{#1}\pdata@def{task}{\taskin\task@id\wp@id}{partners}{#1}\@dmp{
\define@key{task}{PM}{\def\task@PM{#1}\pdata@def{task}{\taskin\task@id\wp@id}{PM}{#1}\@dmp{PM=#1}}
```

then we define an auxiliary function that gives them sensible defaults and sets the internal macros.

```
\def\task@set#1{\edef\task@id{task\thetask@all}
\def\task@wphases{0-0}\def\task@partners{}\def\task@lead{}
\setkeys{task}{#1}}
```

@post@title@space make the space after the title tweakable

```
\def\task@post@title@space{\quad}
```

task

```
\newcounter{alltasks}
\def\task@post@title@space{\quad}
\newenvironment{task}[1][]%
{\stepcounter{alltasks}
\@task{#1}\item[\pdata@target{task}{\taskin\task@id\wp@id}{\task@label{\thetask@wp}}]%
\@ifundefined{task@title}{}{\textbf\task@title}\task@post@title@space%
\def\@initial{0-0}\ifx\task@wphases\@initial\else%
\ (\let\@@sep=\relax\@for\@I:=\task@wphases%
\do{\decode@wphase\@I\@@sep\show@wphase\wphase@start\wphase@end\wphase@force\let\@@sep=\sep@wphases}%
\ifx\task@lead\@empty\else; \task@legend@partners: \site\task@lead~(\legend@lead)\fi%
\ifx\task@partners\@empty\else\@for \@I:=\task@partners\do{, \site\@I}\fi)\\\fi}
{\ignorespaces}
```

now the multilingual support and presentation configuration

```
\newcommand\month@label[1]{M#1}
\newcommand\show@wphase[3]{\def\@test{#3}\month@label{#1}-\month@label{#2}%
\ifx\@test\@empty\@ #3}
\newcommand\sep@wphases{; }
\newcommand\legend@partners{Partners}
\newcommand\legend@lead{lead}
\newcommand\task@label@long{Task}
```

\@task The \@task macro is a internal macro which takes a bunch of keyword keys and writes their values to the aux file.

```
\newcounter{task@all}\newcounter{task@wp}[wp]
\newcount\task@@end
\def\@task#1{\stepcounter{task@all}\stepcounter{task@wp}%
\task@set{#1}%
\pdata@def{task}{\taskin\task@id\wp@id}{wphases}\task@wphases
\pdata@def{task}{\taskin\task@id\wp@id}{label}{\task@label\thetask@wp}%
\pdata@def{task}{\taskin\task@id\wp@id}{number}{\thetask@wp}%
\pdata@def{task}{\taskin\task@id\wp@id}{page}{\thepage}%
\update@tasks{\taskin\task@id\wp@id}}
```

\workphase

```
\newcommand\workphase[1]{\PackageError{proposal}
  {The \protect\workphase macro is deprecated,\MessageBreak
    use the attributes wphase on the workpackage environment instead!}}
```

\localtaskref

```
\newcommand\localtaskref[1]{\pdataRef{task}{\wp@id @#1}{label}}
```

\taskref

```
\newcommand\taskin[2]{#2@#1}
\newcommand\taskref[2]{\WPref{#1}.\pdataRef{task}{#1@#2}{label}}
\newcommand\taskreflong[2]{\WPref{#1}.\pdataRef{task}{#2}{label}}
\newcommand\tasktref[2]{\WPref{#1} (\task@label@long \pdataRef{task}{#1@#2}{number})}
\newcounter{gantt@deps}
\def\@requires#1#2{\stepcounter{gantt@deps}%
\edef\dep@id{taskdep\thegantt@deps}%
\pdata@def{taskdep}\dep@id{from}{\taskin{#1}\wp@id}%
\pdata@def{taskdep}\dep@id{to}{#2}%
\update@deps\dep@id}
⟨/cls⟩
```

4.9 Project Data, Referencing & Hyperlinking

`\pdata@*` `\pdata@out` is the file handle for the project data file, we define internal macros to open and close it.

```
⟨*pdata⟩
\newif\ifwork@areas\work@areastrue
\DeclareOption{noworkareas}{\work@areasfalse}
\ProcessOptions
\RequirePackage{xspace}
\newwrite\pdata@out
\newcommand\pdata@open[1]{\immediate\openout\pdata@out=#1.pdata}
\newcommand\pdata@close{\closeout\pdata@out}
```

`\readpdata` This macro reads the project data file and its error handling

```
\newcommand\readpdata[1]{\IfFileExists{#1.pdata}
{\message{proposal: Reading Project Data}\makeatletter\input{#1.pdata}\makeatother}
{proposal: No Project Data found, (forward) references may be compromized}}
```

`\pdata@target` This internal macro makes a hyper-target: `\pdata@target{`⟨*cat*⟩`}{`⟨*id*⟩`}{`⟨*label*⟩`}` prints ⟨*label*⟩ with a target name ⟨*cat*⟩`@`⟨*id*⟩`@target` attached to it.

```
\newcommand\pdata@target[3]{\hypertarget{#1@#2@target}{#3}}
```

`\pdata@def` This macro writes an `\@pdata@def` command to the current aux file and also executes it.

```
\newcommand\pdata@def[4]{%\@pdata@def{#1}{#2}{#3}{#4}%
  \protected@write\pdata@out{}{\string\@pdata@def{#1}{#2}{#3}{#4}}}
```

`\@pdata@def` This macro stores the value of its last argument in a custom macro for reference.

```
\newcommand\@pdata@def[4]{\expandafter\gdef\csname #1@#2@#3\endcsname{#4}}
```

`\pdataref`

```
\newcommand\pdataref[3]{\@ifundefined{#1@#2@#3}%
              {\protect\G@refundefinedtrue\@latex@warning{#3 for #1 #2 undefined}??}%
               {\csname #1@#2@#3\endcsname}}%
\newcommand\pdataref@aux[3]{\@ifundefined{#1@#2@#3}{??}{\csname #1@#2@#3\endcsname}}%
\newcommand\pdataref@num[3]{\@ifundefined{#1@#2@#3}{0}{\csname #1@#2@#3\endcsname}}%
\newcommand\pdataref@safe[3]{\@ifundefined{#1@#2@#3}{}{\csname #1@#2@#3\endcsname}}%
```

`\pdataRef`

```
\newcommand\pdataRef[3]{\@ifundefined{#1@#2@#3}%
{\protect\G@refundefinedtrue\@latex@warning{#3 for #1 #2 undefined}??}%
{\hyperlink{#1@#2@target}{\csname #1@#2@#3\endcsname}}}
```

`\pdatacount`

```
\newcommand\prop@count[1]{\ifcase #1 zero\or one\or two\or three\or four\or five\or six\or seven \or
  eight\or nine\or ten\or eleven \or twelve\else#1\fi}
\newcommand\pdatacount[2]{\prop@count{\pdataref@num{#1}{#2}{count}}}
```

`\pn*`

```
\newcommand\pn{\pdataref{prop}{gen}{acronym}\xspace}
\newcommand\pnlong{\pdataref{prop}{gen}{acrolong}\xspace}
```

`\W*ref`

```
\newcommand\WPref[1]{\pdataRef{wp}{#1}{label}}
\newcommand\WPtref[1]{\pdataRef{wp}{#1}{label}: \pdataRef{wp}{#1}{short}}
\ifwork@areas
\newcommand\WAref[1]{\pdataRef{wa}{#1}{label}}
\newcommand\WAtref[1]{\pdataRef{wa}{#1}{label}: \pdataRef{wa}{#1}{title}}
\fi
⟨/pdata⟩
```

4.10 The Work Package Table

`\prop@lead`

```
⟨*cls⟩
\newcommand\prop@lead[1]{\@ifundefined{wp@#1@lead}%
{\protect\G@refundefinedtrue\@latex@warning{lead for WP #1 undefined}??}%
{\csname wp@#1@lead\endcsname}}
```

EdN:6 `\*@style` [6]

```
\definecolorset{gray/rgb/hsb/cmyk}{}{}%
{leadgray,.90/.90,.90,.90/0,0,.90/0,0,0,.10;%
wagray,.70/.70,.70,.70/0,0,.70/0,0,0,.30}
\newcommand\sum@style[1]{\cellcolor{wagray}{\textbf{#1}}}
\newcommand\wa@style[1]{\cellcolor{wagray}{\textbf{#1}}}
\newcommand\wp@style[1]{#1}
\newcommand\lead@style[1]{\cellcolor{leadgray}{\textit{#1}}}
\newcommand\wp@lead@style@explained{light gray italicised}
```

`wp@figure`

```
\newcounter{wpfig@options}
\define@key{wpfig}{size}{\def\wpfig@size{#1}\@dmp{size=#1}}
\def\@true{true}
\def\wpfig@pages{false}
\define@key{wpfig}{pages}[true]{\def\wpfig@pages{#1}\stepcounter{wpfig@options}}
\def\wpfig@type{false}
\define@key{wpfig}{type}[true]{\def\wpfig@type{#1}\stepcounter{wpfig@options}}
\def\wpfig@start{false}
\define@key{wpfig}{start}[true]{\def\wpfig@start{#1}\stepcounter{wpfig@options}}
\def\wpfig@length{false}
\define@key{wpfig}{length}[true]{\def\wpfig@length{#1}\stepcounter{wpfig@options}}
\def\wpfig@end{false}
\define@key{wpfig}{end}[true]{\def\wpfig@end{#1}\stepcounter{wpfig@options}}
\def\@sw#1{\begin{sideways}#1\end{sideways}}
\newenvironment{wp@figure}{\begin{figure}[ht]\wpfig@style\begin{center}
{\let\@sw\relax\let\textbf\relax\let\site\relax\let\pn\relax\let\sys\relax%
\gdef\wpfig@headline{\wpfig@legend@wap&\wpfig@legend@title%
\ifx\wpfig@type\@true&\wpfig@legend@type\fi%
\ifx\wpfig@pages\@true&\@sw{\wpfig@legend@page}\fi%
\ifx\wpfig@start\@true&\@sw{\wpfig@legend@start}\fi%
\ifx\wpfig@length\@true&\@sw{\wpfig@legend@length}\fi
\ifx\wpfig@end\@true&\@sw{\wpfig@legend@end}\fi}%
\if@sites%
\@for\@site:=\prop@gen@sites\do{%
\xdef\wpfig@headline{\wpfig@headline&\@sw{\wpfig@legend@siteRM{\@site}}}%
\if@RAM\xdef\wpfig@headline{\wpfig@headline&\@sw{\wpfig@legend@siteRAM{\@site}}}\fi}%
\xdef\wpfig@headline{\wpfig@headline&\@sw{\wpfig@legend@totalRM}}%
\if@RAM\xdef\wpfig@headline{\wpfig@headline&\@sw{\wpfig@legend@totalRAM}}\fi%
\else% if@sites
\xdef\wpfig@headline{\wpfig@headline &\@sw{\wpfig@legend@RM}\if@RAM&\@sw{\wpfig@legend@RAM}\fi}
\fi}%if@sites
\if@RAM\begin{tabular}{|l|l|*{\thewpfig@options}{r|}*{\the@sites}{r|r|}|r|r|}\hline
\else\begin{tabular}{|l|l|*{\thewpfig@options}{r|}|*{\the@sites}{r|}|r|}\hline\fi%|
\wpfig@headline\\\hline\hline}
{\end{tabular}\smallskip\\
\wpfig@legend@RAM@expl
\if@sites; \wpfig@legend@lead@expl\fi
\caption{\wpfig@legend@caption}\label{fig:wplist}
```

[6]EdNote: This (and wpfig) should be documented above

```
\end{center}\end{figure}}
```

and now multilinguality support

```
\newcommand\wpfig@legend@wap{\textbf{\ifwork@areas{WA/P}\else{WP}\fi}}
\newcommand\wpfig@legend@title{\textbf{Title}}
\newcommand\wpfig@legend@type{\textbf{type}}
\newcommand\wpfig@legend@page{\textbf{page}}
\newcommand\wpfig@legend@start{\textbf{start}}
\newcommand\wpfig@legend@length{\textbf{length}}
\newcommand\wpfig@legend@end{\textbf{end}}
\newcommand\wpfig@legend@siteRM[1]{\site{#1}\if@RAM\ RM\fi}
\newcommand\wpfig@legend@siteRAM[1]{\site{#1}\ RAM}
\newcommand\wpfig@legend@totalRM{total\if@RAM\ RM\fi}
\newcommand\wpfig@legend@totalRAM{total RAM}
\newcommand\wpfig@legend@RM{RM}
\newcommand\wpfig@legend@RAM{RAM}
\newcommand\wpfig@legend@RAM@expl{\if@RAM R(A)M $\widehat=$ Researcher (Assistant) Months\else\ Efforts in PM
\newcommand\wpfig@legend@lead@expl{WP lead efforts \wp@lead@style@explained}
\newcommand\wpfig@legend@caption{{\ifwork@areas Work Areas and \fi}Work Packages}
```

EdN:7 `\wpfigstyle` [7]

```
\def\wpfig@style{}
\newcommand\wpfigstyle[1]{\def\wpfig@style{#1}}
```

EdN:8 `\wpfig` [8]

```
\newcount\local@count
\newcount\@@@RM\if@RAM\newcount\@@@RAM\fi
\newcount\all@@@RM\if@RAM\newcount\all@@@RAM\fi
\newcommand{\wpfig}[1][]{\setcounter{wpfig@options}{0}\setkeys{wpfig}{#1}
```

the first thing to do is to build the body of the table programmatically by (globally) extending the `\@wp@lines` token register inside a bracket group which locally redefines all macros we are using in the extensions, so that they do not get into the way. We start this group now.

```
{\gdef\@wp@lines{}%initialize
\let\tabularnewline\relax\let\hline\relax\let\lead@style\relax% so they
\let\wa@style\relax\let\wp@style\relax \let\@sw\relax\let\textbf\relax% do not
\let\G@refundefinedtrue=\relax\let\@latex@warning=\relax\let\hyperlink=\relax% bother
\let\pn\relax\let\xspace\relax% us
```

The code that follows now, could be more elegant, if we had a better way of organizing the data, but this works for now, we have four cases: with/without work areas and with/without sites. All do something very similar.

```
\ifwork@areas
\edef\@@was{\pdataref@safe{all}{wa}{ids}}%
\@for\@@wa:=\@@was\do{% iterate over the work areas
\xdef\@@wa@line{\wa@style{\pdataRef{wa}\@@wa{label}}%
&\wa@style{\@ifundefined{wa@\@@wa @short}{\pdataref{wa}\@@wa{title}}{\pdataref{wa}\@@wa{short}}}%
\ifx\wpfig@type\@true&\wa@style{\pdataref{wa}\@@wa{type}}\fi%
\ifx\wpfig@pages\@true&\wa@style{\pdataref{wa}\@@wa{page}}\fi%
\ifx\wpfig@start\@true&\wa@style{\pdataref{wa}\@@wa{start}}\fi%
\ifx\wpfig@length\@true&\wa@style{\pdataref{wa}\@@wa{len}}\fi%
\ifx\wpfig@end\@true&\wa@style{\pdataref{wa}\@@wa{end}}\fi}
\if@sites
\@for\@site:=\prop@gen@sites\do{%
\edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}%
\local@count 0%
```

[7]EdNote: document above

[8]EdNote: The computation can be distributed much more efficiently (by intermingling the counter advances with the row creation), but this works now

```
\@for\@@wp:=\@@wps\do{\advance\local@count by \pdataref@num\@@wp\@site{RM}}%
\pdata@def\@@wa\@site{RM}{\the\local@count}%
\xdef\@@wa@line{\@@wa@line&\wa@style{\the\local@count}}%
\if@RAM
\local@count 0%
\@for\@@wp:=\@@wps\do{\advance\local@count by \pdataref@num\@@wp\@site{RAM}}
\pdata@def\@@wa\@site{RAM}{\the\local@count}%
\xdef\@@wa@line{\@@wa@line&\wa@style{\the\local@count}}%
\fi}
\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num\@@wa\@site{RM}}%
\xdef\@@wa@line{\@@wa@line &\wa@style{\textbf{\the\local@count}}}
\if@RAM
\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num\@@wa\@site{RAM}}%
\xdef\@@wa@line{\@@wa@line &\wa@style{\textbf{\the\local@count}}}
\fi
\else% if@sites
\edef\@@wps{\pdataref@safe{all}{wp}{ids}}%
\xdef\@@wa@line{\@@wa@line&\wa@style{\pdataref{wa}\@@wa{RM}}
\if@RAM&\wa@style{\pdataref{wa}\@@wa{RAM}}\fi}%
\fi% if@sites
\xdef\@wp@lines{\@wp@lines\@@wa@line\tabularnewline\hline}% add the line for the workarea
\edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over its work packages
\xdef\@@wp@line{\pdataRef{wp}\@@wp{label}%
&\@ifundefined{wp@\@@wp @short}{\pdataref{wp}\@@wp{title}}{\pdataref{wp}\@@wp{short}}%
\ifx\wpfig@type\@true&\pdataref{wp}\@@wp{type}\fi%
\ifx\wpfig@pages\@true&\pdataref{wp}\@@wp{page}\fi%
\ifx\wpfig@start\@true&\pdataref{wp}\@@wp{start}\fi%
\ifx\wpfig@length\@true&\pdataref{wp}\@@wp{len}\fi%
\ifx\wpfig@end\@true&\pdataref{wp}\@@wp{end}\fi}
\if@sites
\@for\@site:=\prop@gen@sites\do{%
\edef\@@lead{\pdataref@safe{wp}\@@wp{lead}}
\edef\@@RM{\ifx\@@lead\@site\lead@style{\pdataref@safe\@@wp\@site{RM}}\else\wp@style{\pdataref@safe\@@wp\@sit
\xdef\@@wp@line{\@@wp@line&\@@RM}
\if@RAM
\edef\@@RAM{\ifx\@@lead\@site\lead@style{\pdataref@safe\@@wp\@site{RAM}}\else\wp@style{\pdataref@safe\@@wp\@s
\xdef\@@wp@line{\@@wp@line&\@@RAM}
\fi}
\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num\@@wp\@site{RM}}%
\xdef\@@wp@line{\@@wp@line &\textbf{\the\local@count}}
\if@RAM
\global\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num\@@wp\@site{RAM}}%
\xdef\@@wp@line{\@@wp@line &\textbf{\the\local@count}}
\fi% if@sites
\else% if@sites
\xdef\@@wp@line{\@@wp@line&\wp@style{\pdataref@safe{wp}\@@wp{RM}}}
\if@RAM\xdef\@@wp@line{\@@wp@line&\wp@style{\pdataref@safe{wp}\@@wp{RAM}}}\fi
\fi% if@sites
\xdef\@wp@lines{\@wp@lines\@@wp@line\tabularnewline\hline}}
```

Now the case where we do not have work areas.

```
\else% ifwork@areas
\edef\@@wps{\pdataref@safe{all}{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over its work packages
```

```
\xdef\@@wp@line{\pdataRef{wp}\@@wp{label}%
&\@ifundefined{wp@\@@wp @short}{\pdataref{wp}\@@wp{title}}{\pdataref{wp}\@@wp{short}}
\ifx\wpfig@type\@true&\pdataref{wp}\@@wp{type}\fi%
\ifx\wpfig@pages\@true&\pdataref{wp}\@@wp{page}\fi%
\ifx\wpfig@start\@true&\pdataref{wp}\@@wp{start}\fi%
\ifx\wpfig@length\@true&\pdataref{wp}\@@wp{len}\fi%
\ifx\wpfig@end\@true&\pdataref{wp}\@@wp{end}\fi}
\if@sites
\@for\@site:=\prop@gen@sites\do{%
\edef\@@lead{\pdataref@safe{wp}\@@wp{lead}}
\edef\@@RM{\ifx\@@lead\@site\lead@style{\pdataref@safe\@@wp\@site{RM}}\else\wp@style{\pdataref@safe\@@wp\@sit
\xdef\@@wp@line{\@@wp@line&\@@RM}
\if@RAM
\edef\@@RAM{\ifx\@@lead\@site\lead@style{\pdataref@safe\@@wp\@site{RAM}}\else\wp@style{\pdataref@safe\@@wp\@s
\xdef\@@wp@line{\@@wp@line&\wp@style\@@RAM}
\fi}
\global\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num\@@wp\@site{RM}}%
\xdef\@@wp@line{\@@wp@line &\textbf{\the\local@count}}
\if@RAM
\global\local@count0\relax%
\@for\@site:=\prop@gen@sites\do{\global\advance\local@count by \pdataref@num{#1}\@site{RAM}}%
\xdef\@@wp@line{\@@wp@line &\textbf{\the\local@count}}
\fi
\else% if@sites
\xdef\@@wp@line{\@@wp@line&\wp@style{\pdataref@safe{wp}\@@wp{RM}}}
\if@RAM\xdef\@@wp@line{\@@wp@line&\wp@style{\pdataref@safe{wp}\@@wp{RAM}}\fi}
\fi% if@sites
\xdef\@wp@lines{\@wp@lines\@@wp@line\tabularnewline\hline}}
\fi%ifwork@areas
```

Now we compute the totals lines in the `\@totals` macros; again there are four cases to consider

```
\gdef\@totals{}
\ifwork@areas
\if@sites
\@for\@site:=\prop@gen@sites\do{% iterate over the sites
\@@@RM=0\if@RAM\@@@RAM=0\fi
\edef\@@was{\pdataref@safe{all}{wa}{ids}}%
\@for\@@wa:=\@@was\do{% iterate over the work areas
\edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over the work packages
\advance\@@@RM by \pdataref@num\@@wp\@site{RM}%
\if@RAM\advance\@@@RAM by \pdataref@num\@@wp\@site{RAM}\fi}}
\pdata@def{all}\@site{RM}{\the\@@@RM}\if@RAM\pdata@def{all}\@site{RAM}{\the\@@@RAM}\fi
\advance\all@@@RM by \the\@@@RM\if@RAM\advance\all@@@RAM by \the\@@@RAM\fi
\xdef\@totals{\@totals & \textbf{\the\@@@RM}\if@RAM& \textbf{\the\@@@RAM}\fi}}
\xdef\@totals{\@totals & \textbf{\the\all@@@RM}\if@RAM&\textbf{\the\all@@@RAM}\fi}
\pdata@def{all}{total}{RM}{\the\all@@@RM}\if@RAM\pdata@def{all}{total}{RAM}{\the\all@@@RAM}\fi
\else% if@sites
\@@@RM=0\if@RAM\@@@RAM=0\fi
\edef\@@was{\pdataref@safe{all}{wa}{ids}}%
\@for\@@wa:=\@@was\do{\edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over the work packages
\advance\@@@RM by \pdataref@num{wp}\@@wp{RM}%
\if@RAM\advance\@@@RAM by \pdataref@num{wp}\@@wp{RAM}\fi}}
\pdata@def{all}{total}{RM}{\the\@@@RM}\if@RAM\pdata@def{all}{total}{RAM}{\the\@@@RAM}\fi
\xdef\@totals{&\the\@@@RM\if@RAM &\the\@@@RAM\fi}
\fi% if@sites
\else%i.e. no work@areas
```

```
\if@sites
\@for\@site:=\prop@gen@sites\do{%iterate over the sites
\@@@RM=0\if@RAM\@@@RAM=0\fi%
\edef\@@wps{\pdataref@safe{all}{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over the work packages
\advance\@@@RM by \pdataref@num\@@wp\@site{RM}%
\if@RAM\advance\@@@RAM by \pdataref@num\@@wp\@site{RAM}\fi}
\pdata@def{all}\@site{RM}{\the\@@@RM}\if@RAM\pdata@def{all}\@site{RAM}{\the\@@@RAM}\fi
\xdef\@totals{\@totals & \textbf{\the\@@@RM}\if@RAM& \textbf{\the\@@@RAM}\fi}
\advance\all@@@RM by \the\@@@RM\if@RAM\advance\all@@@RAM by \the\@@@RAM\fi}
\xdef\@totals{\@totals &\textbf{\the\all@@@RM}\if@RAM&\textbf{\the\all@@@RAM}\fi}
\pdata@def{all}{total}{RM}{\the\all@@@RM}\if@RAM\pdata@def{all}{total}{RAM}{\the\all@@@RAM}\fi
\else% if@sites
\@@@RM=0\if@RAM\@@@RAM=0\fi
\edef\@@wps{\pdataref@safe{all}{wp}{ids}}%
\@for\@@wp:=\@@wps\do{% iterate over the work packages
\advance\@@@RM by \pdataref@num{wp}\@@wp{RM}%
\if@RAM\advance\@@@RAM by \pdataref@num{wp}\@@wp{RAM}\fi}
\pdata@def{all}{total}{RM}{\the\@@@RM}\if@RAM\pdata@def{all}{total}{RAM}{\the\@@@RAM}\fi
\xdef\@totals{&\the\@@@RM\if@RAM &\the\@@@RAM\fi}
\fi% if@sites
\fi
```

And we finally have a line for the intended totals which we use in draft mode.

```
\gdef\intended@totals{}\gdef\requested@totals{}
\if@sites
\@for\@site:=\prop@gen@sites\do{
\xdef\intended@totals{\intended@totals&\textbf{\pdataref@safe{site}\@site{intendedRM}}}
\xdef\requested@totals{\requested@totals&\pdataref@safe{site}\@site{reqPM}}
\if@RAM\xdef\intended@totals{\intended@totals&\textbf{\pdataref@safe{site}\@site{intendedRAM}}}\fi}
\if@RAM\xdef\intended@totals{\intended@totals&&}\else%
\xdef\intended@totals{\intended@totals&}%
\xdef\requested@totals{\requested@totals&}%
\fi
\else% if@sites
\xdef\intended@totals{\intended@totals&\textbf{\pdataref@safe{all}{intended}{RM}}}
\if@RAM\xdef\intended@totals{\intended@totals&\textbf{\pdataref@safe{all}{intended}{RAM}}}\fi
\fi}% if@sites
```

finally, we make all of this into a figure, computing the colspan of the the legend cells for the totals via `\local@count` from the optional columns.

```
\local@count\thewpfig@options\advance\local@count by 2
\begin{wp@figure}
\@wp@lines\hline%
\multicolumn{\the\local@count}{|c|}{\prop@legend@totals}\@totals\\\hline%
\ifsubmit\else%
\multicolumn{\the\local@count}{|c|}{\prop@legend@intendedtotals}\intended@totals\\\hline
\multicolumn{\the\local@count}{|c|}{\prop@legend@requestedtotals}\requested@totals\\\hline
\fi
\end{wp@figure}}
```

and now multilinguality support

```
\newcommand\prop@legend@totals{\textbf{totals}}
\newcommand\prop@legend@intendedtotals{\textbf{intended totals}}
\newcommand\prop@legend@requestedtotals{\textbf{requested totals}}
```

4.11 Gantt Charts

Gantt Charts are done with help of the the `tikz` package. The `gantt` environments pick up on the declared duration of the proposal in months stored in the `\prop@gen@months` macro.

We define the keys for Gantt tables

```
\newif\ifgantt@draft\gantt@draftfalse
\define@key{gantt}{xscale}{\def\gantt@xscale{#1}}
\define@key{gantt}{yscale}{\def\gantt@yscale{#1}}
\define@key{gantt}{step}{\def\gantt@step{#1}}
\define@key{gantt}{size}{\def\gantt@size{#1}}
\define@key{gantt}{draft}[true]{\ifsubmit\else\gantt@drafttrue\fi}
```

Then we define an auxiliary function that provides defaults for these keys and sets the internal macros.

```
\def\gantt@set#1{\gantt@draftfalse\def\gantt@xscale{1}\def\gantt@yscale{.35}\def\gantt@step{3}
\setkeys{gantt}{#1}}
```

Finally, the Gantt Chart environment itself.

`gantt` The `gantt[`⟨*keyvals*⟩`]{`⟨*height*⟩`}` environment sets up the grid and legend for a gantt chart. The grid is `\prop@gen@months` wide and ⟨*height*⟩ high.

```
\newenvironment{gantt}[2][]
{\gantt@set{#1}
\def\@test{\prop@gen@months@default}
\ifx\@test\prop@gen@months
\ClassError{proposal}{Need overall project months to draw gantt
    chart - expect trouble;\MessageBreak specify
    \protect\begin{proposal}[...,months=??,...] to fix}\fi
\@ifundefined{gantt@size}{}{\csname\gantt@size\endcsname}
\newdimen\gantt@ymonths
\gantt@ymonths=#2 cm
\advance\gantt@ymonths by .5cm
\begin{tikzpicture}[xscale=\gantt@xscale,yscale=\gantt@yscale]
\draw[xstep=\gantt@step,gray,very thin] (0,0) grid (\prop@gen@months,#2);
\foreach \x in {0,\gantt@step,...,\prop@gen@months} \node at (\x,\gantt@ymonths) {\x};}
{\end{tikzpicture}}
```

`\@action` In this we have used the macro that does the actual painting. `\@action{`⟨*name*⟩`}{`⟨*line*⟩`}{`⟨*start*⟩`}{`⟨*len*⟩`}{`⟨*force*⟩`}` creates a gantt node with name ⟨*name*⟩ in line ⟨*line*⟩ starting at month ⟨*month*⟩ with length ⟨*len*⟩ that is ⟨*force*⟩ thick.

```
\newdimen\gantt@ymid\newdimen\gantt@yinc\newdimen\gantt@xend
\newcommand{\@action}[5]{%
\gantt@ymid=#2 cm\gantt@yinc=\gantt@yscale cm
\gantt@xend=#3 cm\advance\gantt@xend by #4 cm
\advance\gantt@ymid by \gantt@yinc
\fill (#3,#2) rectangle +(#4,#5);
\node (#1@left) at (#3,\gantt@ymid) {};
\node (#1@right) at (\gantt@xend,\gantt@ymid) {};}
```

`\@dependency`

```
\def\@dependency#1#2{\draw[->,line width=2pt,color=red] (#1@right) -- (#2@left);}
```

`tt@compute@effort` A helper function that updates the dimension `\gantt@effort` according to whether the counter `\gantt@month` is in the range. It is used in `\gantt@chart`

```
\newcommand\gantt@compute@effort[3]{% start, len, force
  \@@e=#1\advance\@@e by #2
  \ifnum\thegantt@month<#1\else
  \ifnum\thegantt@month<\@@e
  \gantt@plus=#3cm\advance\gantt@effort by \gantt@plus\fi\fi}
```

`\ganttchart` This macro iterates over the work areas, their work packages, and finally their work phases to use the internal macro `\@action`. All of this in the gantt setting.

```
\newcommand{\ganttchart}[1][]{\begin{figure}[ht]\centering
\gantt@set{#1}
\def\gantt@wps{\pdataref@num{all}{wp}{count}}
\begin{gantt}[#1]{\gantt@wps}
 \newcounter{taskwps}\newcount\@@line
 \edef\@@was{\pdataref@safe{all}{wa}{ids}}
 \ifwork@areas
 \@for\@@wa:=\@@was\do{% iterate over work areas
   \edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}
   \@for\@@wp:=\@@wps\do{% iterate over work packages
     \stepcounter{taskwps}
     \@@line=\gantt@wps\advance\@@line by -\thetaskwps
     \edef\@@tasks{\pdataref@safe\@@wp{task}{ids}}
     \node at (-1/\gantt@xscale,\@@line) [above=-2pt] {\pdataRef{wp}\@@wp{label}};
     \edef\@@wphases{\pdataref@safe{wp}\@@wp{wphases}}
     \@for\@@ft:=\@@wphases\do{%wp-level work phases
       \decode@wphase\@@ft
       \@action\@@wp\@@line\wphase@start\wphase@len\wphase@force}
     \@for\@@task:=\@@tasks\do{% tasks
       \edef\@@wphases{\pdataref@safe{task}\@@task{wphases}}
       \@for\@@ft:=\@@wphases\do{%task-level work phases
         \decode@wphase\@@ft
         \@action\@@task\@@line\wphase@start\wphase@len\wphase@force}}}}
 \else% ifwork@areas false
 \edef\@@wps{\pdataref@safe{all}{wp}{ids}}
 \@for\@@wp:=\@@wps\do{% iterate over work packages
   \stepcounter{taskwps}
   \@@line=\gantt@wps\advance\@@line by -\thetaskwps
   \edef\@@tasks{\pdataref@safe\@@wp{task}{ids}}
   \node at (-1/\gantt@xscale,\@@line) [above=-2pt] {\pdataRef{wp}\@@wp{label}};
   \edef\@@wphases{\pdataref@safe{wp}\@@wp{wphases}}
   \@for\@@ft:=\@@wphases\do{%iterate over the wp-level work phases
     \decode@wphase\@@ft
     \@action\@@wp\@@line\wphase@start\wphase@len\wphase@force}
   \@for\@@task:=\@@tasks\do{% task-level work phases
     \edef\@@wphases{\pdataref@safe{task}\@@task{wphases}}
     \@for\@@ft:=\@@wphases\do{%iterate over the task-level work phases
       \decode@wphase\@@ft
       \@action\@@task\@@line\wphase@start\wphase@len\wphase@force}}}
 \fi% ifwork@areas end
 \edef\@@deps{\pdataref@safe{all}{task}{deps}}
 \@for\@@dep:=\@@deps\do{%
   \@dependency{\pdataref@safe{taskdep}\@@dep{from}}{\pdataref@safe{taskdep}\@@dep{to}}}
```

The next piece of code generates the effort sum table in draft mode

```
 \ifgantt@draft
    \newcounter{gantt@month}
    \newcount\@@e\newdimen\gantt@effort\newdimen\gantt@plus
    \@whilenum\thegantt@month<\prop@gen@months\do{% step over months
      \gantt@effort=0cm
      \ifwork@areas
      \edef\@@was{\pdataref@safe{all}{wa}{ids}}
      \@for\@@wa:=\@@was\do{% iterate over work areas
        \edef\@@wps{\pdataref@safe\@@wa{wp}{ids}}
        \@for\@@wp:=\@@wps\do{% iterate over work packages
          \edef\@@wphases{\pdataref@safe{wp}\@@wp{wphases}}
          \@for\@@ft:=\@@wphases\do{%iterate over the wp-level work phases
            \decode@wphase\@@ft
            \gantt@compute@effort\wphase@start\wphase@len\wphase@force}
        \edef\@@tasks{\pdataref@safe\@@wp{task}{ids}}
        \@for\@@task:=\@@tasks\do{% iterate over tasks
        \edef\@@wphases{\pdataref@safe{task}\@@task{wphases}}
        \@for\@@ft:=\@@wphases\do{%iterate over the wp-level work phases
          \decode@wphase\@@ft
          \gantt@compute@effort\wphase@start\wphase@len\wphase@force}}}}
    \fill  (\thegantt@month,-5) rectangle +(1,\gantt@effort);
    \else% ifwork@areas
    \edef\@@wps{\pdataref@safe{all}{wp}{ids}}
    \@for\@@wp:=\@@wps\do{% iterate over work packages
        \edef\@@wphases{\pdataref@safe{wp}\@@wp{wphases}}
        \@for\@@ft:=\@@wphases\do{%iterate over the wp-level work phases
          \decode@wphase\@@ft
          \gantt@compute@effort\wphase@start\wphase@len\wphase@force}
        \edef\@@tasks{\pdataref@safe\@@wp{task}{ids}}
        \@for\@@task:=\@@tasks\do{% iterate over tasks
        \edef\@@wphases{\pdataref@safe{task}\@@task{wphases}}
        \@for\@@ft:=\@@wphases\do{%iterate over the wp-level work phases
          \decode@wphase\@@ft
          \gantt@compute@effort\wphase@start\wphase@len\wphase@force}}}
    \fill  (\thegantt@month,-5) rectangle +(1,\gantt@effort);
    \fi% ifwork@areas
    \stepcounter{gantt@month}}
  \fi% ifgantt@draft
 \end{gantt}
 \caption{\gantt@caption}\label{fig:gantt}
\end{figure}}
```

now the multilingual support

```
\newcommand\gantt@caption@main{Overview Work Package Activities}
\newcommand\gantt@caption@lower{lower bar shows the overall effort \if@RAM (RAM only)\fi per month}
\newcommand\gantt@caption{\gantt@caption@main\ifgantt@draft\xspace (\gantt@caption@lower)\fi}
```

\gantttaskchart EdN:9

This macro is a variant of \ganttchart, but it shows the tasks consecutively, as is useful for EU projects[9]

```
\newcommand{\gantttaskchart}[1][]{\begin{figure}[ht]\centering\gantt@set{#1}
\def\gantt@tasks{\pdataref@num{all}{task}{count}}
\begin{gantt}[#1]{\gantt@tasks}
  \newcounter{gantt@tasks}\newcount\@@line
  \edef\@@wps{\pdataref@safe{all}{wp}{ids}}
   \@for\@@wp:=\@@wps\do{% iterate over work packages
     \edef\@@tasks{\pdataref@safe\@@wp{task}{ids}}
     \@for\@@task:=\@@tasks\do{% iterate over the tasks
       \stepcounter{gantt@tasks}
       \@@line=\gantt@tasks\advance\@@line by -\thegantt@tasks
       \node at (-1/\gantt@xscale,\@@line) [above=-2pt] {\taskreflong\@@wp\@@task};
       \edef\@@wphases{\pdataref@safe{task}\@@task{wphases}}
       \@for\@@ft:=\@@wphases\do{%iterate over the task-level work phases
         \decode@wphase\@@ft
         \@action\@@task\@@line\wphase@start\wphase@len\wphase@force
      }}}% end all iterations
    \end{gantt}
    \caption{\gantt@caption@main}\label{fig:gantt}
\end{figure}}
```

[9] EdNote: this should be incorporated with the gantt chart above, but I am currently to scared to do it so close to the deadline

4.12 Coherence

`\j*`

```
\newcommand\jpub{\textcolor{\prop@link@color}{\textbf{\large{$\star$}}}}
\newcommand\jpro{\textcolor{\prop@link@color}{\textbf{\large{$\bullet$}}}}
\newcommand\jorga{\textcolor{\prop@link@color}{\textbf{\large{$\circ$}}}}
```

`\add@joint` `\add@joint{`⟨*first*⟩`}{`⟨*second*⟩`}{`⟨*sym*⟩`}` adds ⟨*sym*⟩ to the the `\coherence@`⟨*first*⟩`@`⟨*second*⟩ macro for the coherence table.

```
\newcommand\add@joint[3]{\@ifundefined{coherence@#1@#2}%
{\@namedef{coherence@#1@#2}{#3}}%
{\expandafter\g@addto@macro\csname coherence@#1@#2\endcsname{#3}}}
```

`\prop@joint` This iterates over a comma-separated list of names and makes the necessary entries into the coherence table.

```
\newcommand\prop@joint[2]{\@for\@first:=#2\do{%
\@for\@second:=#2\do{\ifx\@first\@second\else\add@joint\@first\@second{#1}\fi}}}
```

`\joint*` Now, some instances that use these.

```
\newcommand\jointproj[1]{\prop@joint\jpro{#1}}
\newcommand\jointpub[1]{\prop@joint\jpro{#1}}
\newcommand\jointorga[1]{\prop@joint\jorga{#1}}
```

`\coherencematrix`

```
\newcommand{\coherencematrix}{
{\let\tabularnewline\relax\let\hline\relax\let\site\relax% so they do
 \let\@sw\relax\let\jpub\relax\let\jpro\relax\let\jorga\relax% not bother us
\gdef\@ct@head{}
\@for\@site:=\prop@gen@sites\do{\xdef\@ct@head{\@ct@head &\site{\@site}}}
\gdef\@ct@lines{\@ct@head\tabularnewline\hline\hline} %initialize with head line
\@for\@site:=\prop@gen@sites\do{\xdef\@ct@line{\site{\@site}}
  \@for\@@site:=\prop@gen@sites\do{
    \xdef\@ct@line{\@ct@line&\ifx\@site\@@site{X}\fi
      \@ifundefined{coherence@\@site @\@@site}{}{\@nameuse{coherence@\@site @\@@site}}}}
  \xdef\@ct@lines{\@ct@lines\@ct@line\tabularnewline\hline}}}
\begin{tabular}{|l||*{\the@site}{c|}}\hline
\@ct@lines\hline
joint&\multicolumn{\the@site}{l|}{\jpub $\hat=$ publication, \jpro $\hat=$ project,
  \jorga $\hat=$ organization}\\\hline
\end{tabular}}
```

`\coherencetable`

```
\newcommand\coherencetable{%
\begin{table}[ht]
\begin{center}\small\setlength{\tabcolsep}{.5em}
\renewcommand{\arraystretch}{.9}\coherencematrix
\end{center}
\caption{\coherence@caption}\label{tab:collaboration}
\end{table}}
```

now the multilinguality support

```
\newcommand\coherence@caption{Previous Collaboration between {\pn} members}
⟨/cls⟩
```

4.13 Relevant Papers & References

We first define a bibLaTeX bibliography heading that does not create headers, we need it somewhere.

```
⟨*cls | reporting⟩
\defbibheading{empty}{}
```

We define an internal macro that prints a publication list of a given bibTEX entry type and title for convenience. It also adds a `notype=` to the token register `\prop@rl` to deal with the unclassified entries from the list.

```
\newif\if@allpapers\@allpaperstrue
\newcommand\prop@ppl[3][]{\@allpapersfalse\message{ppl processing: #2}%
\printbibliography[heading=subbibliography,type=#2,title=#3#1]%
\@ifundefined{prop@rl}{\xdef\prop@rl{#2}}{\xdef\prop@rl{\prop@rl, #2}}}
```

The following code does not work yet, it would have been nice to be able to just add a key `unclassified` to catch the unclassified ones. I guess we just have to issue a warning instead.

```
\newcommand\prop@prl[1]{\message{unclassified: #1}%
\printbibliography[heading=subbibliography,title=Unclassified,#1]}%
\define@key{paperlist}{unclassified}[true]{\message{unclass: \prop@rl}\prop@prl\prop@rl}
```

with this, we define a couple of keys that generate

```
\define@key{paperlist}{articles}[true]{\prop@ppl{article}{Articles}}
\define@key{paperlist}{chapters}[true]{\prop@ppl{inbook}{Book Chapters}}
\define@key{paperlist}{confpapers}[true]{\prop@ppl[,keyword=conference]{inproceedings}{Conference Papers}}
\define@key{paperlist}{wspapers}[true]{\prop@ppl[,notkeyword=conference]{inproceedings}{Workshop Papers}}
\define@key{paperlist}{theses}[true]{\prop@ppl{thesis}{Theses}}
\define@key{paperlist}{submitted}[true]{\prop@ppl[,keyword=submitted]{unpublished}{Submitted}}
\define@key{paperlist}{books}[true]{\prop@ppl{book}{Monographs}}
\define@key{paperlist}{techreports}[true]{\prop@ppl{techreport}{Technical Reports}}
```

`featured` We introduce a new bibLaTeX category `featured` for those papers that were already mentioned in `\prop@paperlist` and the macros defined from it.

```
\DeclareBibliographyCategory{featured}
```

`\prop@paperlist` We generate a subsection with a `refsection` (this makes a separate bibliography for this section) and activate the keys via `\nocite`. Then we just print the bibliography with the empty header we created before.

```
\newcommand\prop@paperlist[2][]{%
\begin{refsection}%
\nocite{#2}\addtocategory{featured}{#2}%
\let\biboldfont\bibfont%
\renewcommand{\bibfont}{\footnotesize}%
\renewcommand{\baselinestretch}{.9}
\setkeys{paperlist}{#1}
\@ifundefined{prop@rl}{}{\@latex@warning{some papers are not classified!}}
\if@allpapers\printbibliography[heading=empty]\fi%
\let\bibfont\biboldfont%
\end{refsection}}
```

We only have to define the `warnpubs` and `empty` heading constructors

```
\def\prop@warnpubs@message{Many of the proposers' publications are online at one of the following URIs:}
\def\prop@warnpubs@title{References}
\defbibheading{warnpubs}{\section*{\prop@warnpubs@title}%
 \@ifundefined{prop@gen@pubspages}
 {\@latex@warning{No publication pages specified;
                  use the pubspage key in the proposal environment!}}
 {\prop@warnpubs@message%
 \@for\@I:=\prop@gen@pubspages\do{\par\noindent\csname\@I\endcsname}}}
```

```
\defbibheading{empty}{}
⟨/cls | reporting⟩
```

4.14 Miscellaneous

\signatures

```
⟨*pdata⟩
\newcommand{\signatures}[1]{\section{#1}
\qquad\number\day. \number\month. \number\year\\[6ex]
\strut\qquad Date\hfill\@for\@p:=\prop@gen@PIs\do{%
\wa@ref{person}\@p{personaltitle}~\wa@ref{person}\@p{name}\hfill}}
```

\@dmp The \@dmp macro shows metadata information about the keys in the margin if \keystrue is specified. This is a debugging tool.

```
\def\@dmp#1{\ifkeys\marginpar{#1}\fi}
```

\euro

```
\renewcommand\euro{\officialeuro\xspace}
⟨/pdata⟩
```

References

[Koh14a] Michael Kohlhase. *Editorial Notes for LaTeX*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2014.

[Koh14b] Michael Kohlhase. *Preparing DFG Proposals and Reports in LaTeX with `dfgproposal.cls`*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2014. URL: http://www.ctan.org/get/macros/latex/contrib/proposal/dfg/dfgproposal.pdf.

[Koh14c] Michael Kohlhase. *`workaddress.sty`: An Infrastructure for marking up Dublin Core Metadata in LaTeX documents*. Tech. rep. Comprehensive TeX Archive Network (CTAN), 2014. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/workaddress/workaddress.pdf.

[Lon] Brent Longborough. *gitinfo2.sty. A package for accessing metadata from the git dvcs.* URL: http://mirrors.ctan.org/macros/latex/contrib/gitinfo2/gitinfo2.pdf (visited on 10/26/2014).