TeXProject[1]: A Mini Manual

Stéphane Bressan Eitan M Gurari

12 May 1995

Contents

Chapter 1

Getting Started

TeXProject is a (La)TeX tool for managing project manuals. TeXProject offers:

- a declarative description language for your projects in terms of subprojects, tasks and milestones;
- a programmable formatted output of project components such as dates, manpower, descriptions or dependencies;
- an extendable graphical output of pert, schedule, tree and effort charts for visualizing the project data.

The advantage of TeXProject over other products is its integration within the (La)TeX environment: you can cross reference your project's items in your document and customize your graphics. Advanced users can create their own functions and their own graphics or extend the existing ones by using the advanced primitives of TeXProject and the primitives of (Al)DraTeX. (Commands with uppercase characters belong to (Al)DraTeX. Commands for drawing environments use parentheses, instead of braces, to delimit their arguments.)

DraTeX is a low-level system that includes commands for drawing primitive shapes like lines, ovals, curves and text as well as commands for combining such entities into sophisticated structures. AlDraTeX is an extension of DraTeX into a high-level system for specifying common entities like charts and diagrams. These systems are described in 'TeX and LaTeX: Drawing and Literate Programming' Gurari, E., McGraw-Hill, 1994.

To produce a management report with TeX or LaTeX using TeXProject you need to include the TeXProject.sty file in your document. TeXProject.sty requires the graphic libraries DraTex.sty and AlDraTex.sty. They are automatically loaded when loading TeXProject:

```
\input TeXProject.sty
```

Chapter 2

Project Definition

A project definition is a set of item definitions for milestones, tasks and subprojects.

Each project item has a reference and a name. Milestones have a date and tasks have begin and end dates. Tasks have resources. Tasks and milestones have dependencies. All items have a description.

Projects inherit their begin and end dates and their resources from the items they are composed of. Project definitions contain this list of items.

Projects, subprojects, milestones and tasks are respectively defined as follow:

- `\defproject{tag}{name}{list of members}`
  `{list of predecessors}{description}`
- `\defmilestone{tag}{name}{date}{list of predecessors}{description}`
- `\deftask{tag}{name}{start date}{end date}{list of contributions}`
  `{list of predecessors}{description}`

  Each contribution is a 'contributor:effort' pair.

The `\newproject` and `\endnewproject` delimiters provide the project definition environment.

The meaning of dates can be fixed using the command:

```
\minmaxdates{from_1/from_2/from_3//to_1/to_2/to_3}
```

The default setting is `\minmaxdates{1/1/00//31/12/99}`

Chapter 3

Debugging Tools

The following commands can be used to turn on and off the trace of input data and the warning messages on possibly inconsistent data:

- `\traceon`;
- `\traceoff` (default);
- `\warningson` (default);
- `\warningsoff`.

Chapter 4

Text-Producing Commands

All the elements of the tasks, milestones, subprojects and project definition are reusable in the text. To each element corresponds a primitive:

- `\description{tag}`;
- `\name{tag}`;
- `\resource{tag}`;
- `\effort{tag}`;
- `\effort{tag,contributor}`;
- `\contributions{tag}`;
- `\start{tag}`;
- `\finish{tag}`.

The output styles of some elements can be parameterized:

- `\datestyle{style}` determines how dates should be listed with a mapping `#1/#2/#3` to `style`. The default is `\datestyle{#1/#2/#3}`;
- `\contributionsstyle{part_1}{part_2}{separator}` determines how contributions should be listed by a mapping from `#1` (the contributor) `#2` (the effort) and a separator to `part_1part_2separator`. The default is `\contributionsstyle{#1}{ (#2)}{, }`.

Chapter 5

Diagrams and Charts Commands

From the project data you can produce several kinds of charts and diagrams:

- tree diagrams representing the project structure;
- pert charts visualizing the dependencies between the project items;
- effort charts visualizing the resources and the efforts for the project items;
- schedule charts visualizing your project schedule.

Each of these diagrams and charts is created by a single primitive in the `\Draw...\EndDraw` environment of (Al)DrateX. You get the default style (layout and output). The style can be changed by using the style commands or by using DrateX commands.

5.1 Tree Diagrams

`\treediagram(tag,depth)`

- `,depth:` : taken to be infinite, if omitted.

`\treestyle(m_command, p_command, t_command, e_command, width)`

- `m_command:` : command to generate milestone nodes;
- `p_command:` : command to generate project nodes;
- `t_command:` : command to generate task nodes;
- `e_command:` : command to generate edges;

- `width:` : required width of node, if 0 then the width is determined by the content.

Default: `\treestyle(\VRectNode, \RectNode, \SRectNode,\TreeEdge, 70)`
`\treenodestyle{style}` determines how content of nodes should be typeset with a mapping from `#1` (a tag) to `style`. The default is `\treenodestyle{\name{#1}}`.

5.2 Pert Diagrams

`\pertdiagram(tag)`
`\pertstyle(option,h_space,v_space,m_command,p_command,t_command,width)`

- `option:` : H (horizontal trees)
- `h-space, v_space:` : amount of space between nodes;
- `m_command:` : command to generate milestone nodes;
- `p_command:` : command to generate project nodes;
- `t_command:` : command to generate task nodes;
- `width:` : required width of node, if 0 then the width is determined by the content.

Default: `\pertstyle(,30,20,\RectNode,\VRectNode,\SRectNode,70)`
`\adjustpert(tag_1,distance_1 & ... & tag_n,distance_n )`
User adjustments to positions of specified nodes. The adjustments are perpendicular to the direction of the diagram.
`\pertnodestyle(style}` determines how content of nodes should be typeset wit a mapping from `#1` (a tag) to `style`. The default is `\treenodestyle{\name{#1}}`.

5.3 Effort Charts

`\effortchart(tag, start date, end date, excluded items, depth)`
Cutoff parameters:

- `excluded items:` : can be any combination of M (milestones), P (projects), T (tasks);
- `start date, end date, excluded items, depth` may be omitted;
- `depth`: : 0 (root level), 1 (children level; default), 2 (grandchildren level), etc.
- trailing commas may be omitted.

`\effortstyle(options, from, to, color, width)`

- `options:` : can be any combination of the following requests: A (default axes), T (names within bars), E (efforts within bars), 3 (3 dimensional bars);
- `from, to:` : location of bar in its slot: `0 <= from < to <= 10` ;
- `color:` : color of bars: 0 (white), 1 (black), 3 (lines);
- `width:` : of names, if 0 then the width is determined by the content

Default: `\effortstyle(A,2.5,7.5,1,70)`

5.4 Schedule Charts

`\schedulechart(tag, start date, end date, excluded items, depth)`
Cutoff parameters:

- `excluded items:` : any combination of: M (milestones), P (projects), T (tasks);
- `start date, end date, excluded items, depth:` : may be omitted
- trailing commas may be omitted.

`\schedulestyle(options, from, to, p_color, t_color, symbol, width)`

- `options:` : can be any combination of the following requests: A (default axes), T (names within bars), 3 (3 dimensional bars);
- `from, to:` : location of bar in its slot: `0 <= from < to <= 10` ;
- `p_color, t_color:` : colors of project and task bars: 0 (white), 1 (black), 3 (lines);
- `symbol:` : for representing milestones;
- `width:` : of names, if 0 then the width is determined by the content.

Default: `\schedulestyle(A,2.5,7.5,3,0,$\otimes$,70)`

5.5 User-Defined Axes

- \datesaxis(loc_1,loc_2)(style) ;
- \effortaxis(loc_1,loc_2)(style) ;
- \namesaxis(loc_1,loc_2)(style) ;
- \nonamesaxis(loc_1,loc_2)(style) .

Variants of the \Axis command of AlDraTeX which get their labels from \schedulechart and \effortchart

5.6 (Al)DraTeX Commands of Immediate Interest

- \Draw ... \EndDraw: : sets a drawing environment;
- \TreeAlign(H,-1,0)(0,0,0): : asks for horizontal trees;
- \TreeSpace(style, h_space, v_space): : style: : C (compact trees);
- \ArrowHeads(1): : asks for arrow heads on edges of diagrams.
- \Scale(h_scale, v_scale): asks for scaled effort and schedule charts.

5.7 Comments

The 3D option is problematic in effort and schedule charts. It should be either removed or fixed, depending on its usefulness.

Cutoff information is not supported for pert diagrams. Such a feature can be implemented if it is determined to be useful.

Chapter 6

Commands for Restricted Environments

The commands `\description`, `\name`, `\resource`, `\effort`, `\contributions`, `\start`, `\finish`, `\apply`, and `\rapply` are not permitted within environments that cannot process definitions of (La)TEX. However, the restriction does not apply to `\description`, `\name`, `\start`, and `\finish` within the drawing structures of TEXProject, and for these latter commands one can also use the following indirect processing approach.

6.1 Preprocessing Within Unrestricted Environment

A `\filtercommand` prefix to the commands `\description`, `\name`, `\start`, and `\finish` provides preprocessing for the commands without producing any output.

6.2 Processing Within Restricted Environment

The command `\putcommand` requests the output of the most recently preprocessed command.

Example. The `\section` command of LATEX can't process definitions that appear in its arguments. Hence, instead of using a command of the form `\section{\name{m1}}`, one should introduce the code

```
\filtercommand\name{m1} \section{\putcommand}.
```

Chapter 7

Programming Tools

The following commands allow the advanced programming of primitives filtering or recursively exploring the structure of a project. An example of their use could be the definition of a standard *company style* report that inputs the project definition and with one command produces the project document.

`\apply{tag, milestone command, project command, task command}`

- `milestone command`, `project command`, `task command` are user defined commands with parameters similar to those offered for `\defmilestone`, `\defproject`, and `\deftask`, respectively;
- `\apply` executes on `tag` the command that applies to the item type.

`\rapply{tag, milestone command, project command, task command}`

- A variant of `\apply` that recursively applies to the given item and its direct and indirect dependents;
- `\depth` records the depth of recursion.

Chapter 8

An Example

8.1 Project Definition

```
\newproject

     \defmilestone{m1}{Announcement}
{27/02/95}
{}
{You read the WWW announcement.}

      \deftask{t1.1}{FTP}
{28/02/95}{01/03/95}
{you:10}
{}
{You ftp the .sty files, amnuals and examples.}

      \deftask{t1.2}{Test}
{01/03/95}{02/03/95}
{you:60}
{t1.1}
{You test the system locally.}

      \deftask{t1.3}{Installation}
{02/03/95}{03/03/95}
{system~engineer:30}
{t1.2}
{You ask your system engineer to install the package
for your centre.}
```

```
      \defproject{t1}{Package}
{t1.1,t1.2,t1.3}
{m1}
{You get and install the package.}

      \defmilestone{m2}{Usage}
{04/03/95}
{t1}
{You can use TeXProject.}

    \defproject{texproject}{TeXProject}
{m1,t1,m2}
{}
{Welcome to TeXProject.}

\endnewproject
```

8.2 Text Output

The following text is an example of output using the project definition of section 8.1. In the text, we have emphasized all the TeXProject outputs and shown the corresponding commands between parentheses.

The project is called TeXProject *(*`\name{texproject}`*).*

It starts on the 27/02/95 *(*`\start{texproject}`*) and ends on the* 04/03/95 *(*`\finish{texproject}`*).*

It involves the following people: you, system engineer *(*`\resource{texproject}`*), and a total amount of man-minutes of* 100 *(*`\effort{texproject}`*). You will be involved for* 70 *(*`\effort{texproject,you}`*) minutes.*

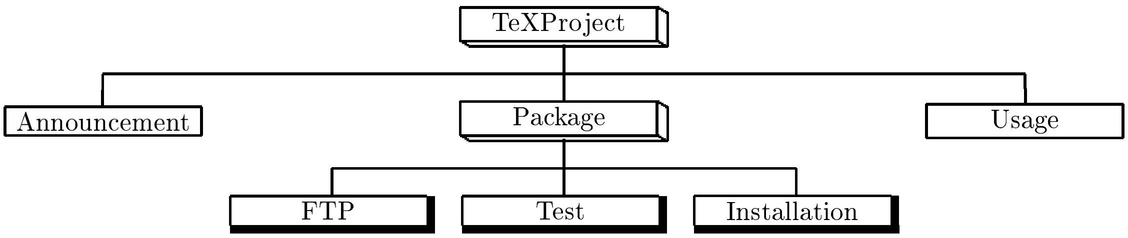
The task Installation *(*`\name{t1.3}`*) has the following description:* You ask your system engineer to install the package for your centre. *(*`\description{t1.3}`*).*

8.3 Graphical Output

The following charts and diagrams is an example of graphical output using the project definition of section 8.1. The Charts and diagrams follow their generating commands.

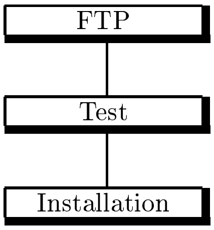
8.3.1 Tree Diagram

```
\Draw
\treediagram(texproject)
\EndDraw
```



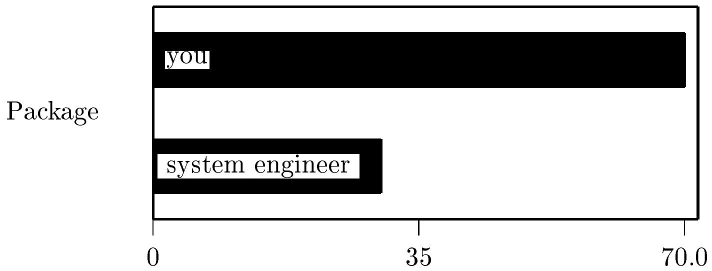
8.3.2 Pert Diagram

```
\Draw
\pertdiagram(t1)
\EndDraw
```

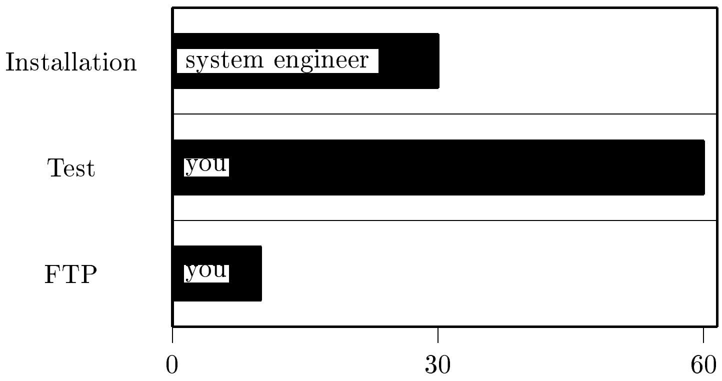


8.3.3 Effort Chart

```
\Draw
\effortchart(texproject)
\EndDraw
```

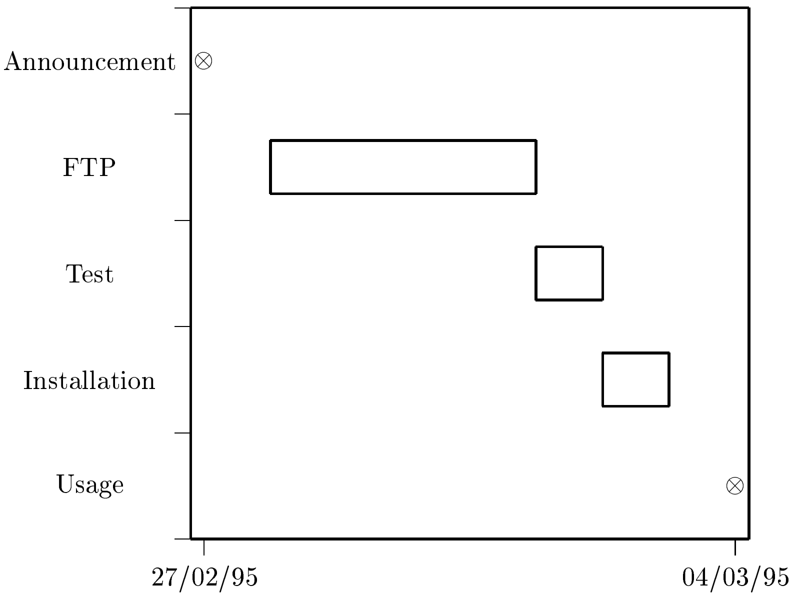



```
\Draw
\effortchart(texproject,,,,2)
\EndDraw
```



8.3.4 Schedule Chart

```
\Draw
\schedulechart(texproject,,,,99)
\EndDraw
```



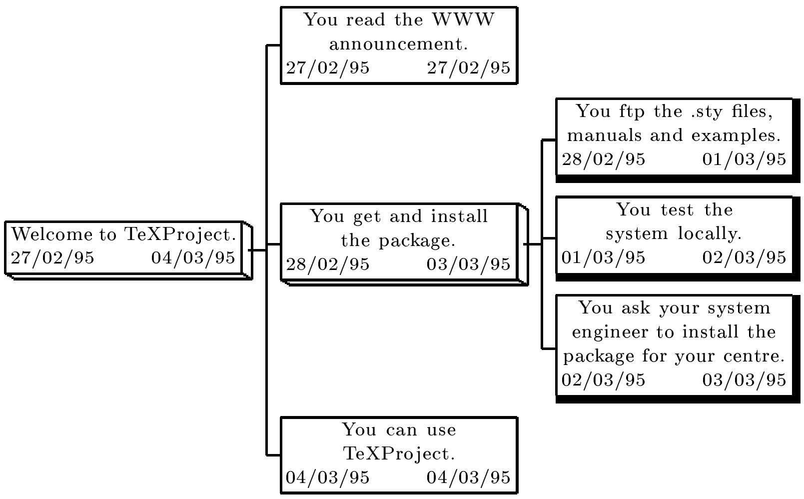
8.4 Beyond the Default Setting

```
\def\members#1{Members of ‘\name{#1}’:
   \rapply{#1,\milestonechild,\projectchild,\taskchild}}
```

```
\def\milestonechild#1#2#3#4#5{\child{#2}}
\let\projectchild=\milestonechild
\def\taskchild#1#2#3#4#5#6#7{\child{#2}}
\def\child#1{\ifnum 1=\depth \hfill\break\qquad #1 \fi }
\members{t1}
```

Members of ‘Package’:
FTP
Test
Installation

```
\Draw
\treenodestyle{\description{#1}\break  \start{#1}\quad\hfill\finish{#1}}
\font\sevenrm=cmr7 \sevenrm
\baselineskip=9pt
\treestyle(,,,,85)
\TreeAlign(H,-1,0)(0,0,0)
\TreeSpace(,5,10)            \treediagram(texproject)
\EndDraw
```




```
\newproject     \minmaxdates{0/0/0//100/0/0}
\defproject{tasks}{}{a,b,c,d}{}{}
\deftask{a}{task 1}{0/0/0}{3/0/0}{}{}{}
\deftask{b}{task 2}{2/0/0}{6/0/0}{}{}{}
\deftask{c}{task 3}{1/0/0}{2/0/0}{}{}{}
\deftask{d}{task 4}{5/0/0}{7/0/0}{}{}{}
\endnewproject
\Draw   \Scale(1,0.5)  \schedulestyle(,,,,3,,)
    \schedulechart(tasks)
```

```
    \MoveToLoc(SW)  \Move(-5pt\du,0)  \MarkLoc(sw)
    \MoveToLoc(NE)  \Move( 5pt\du,0)  \MarkLoc(ne)  \CSeg\DrawRect(ne,sw)
    \namesaxis(sw,ne)(W-0)
    \nonamesaxis(sw,ne)(W-1)
    \font\sevenrm=cmr7      \sevenrm
    \Axis(SW,NE)(N0,& week~1 && week~2 && week~3 && week~4 &&
                      week~5 && week~6 && week~7 & )
    \Axis(SW,NE)(N-1,&&&&&&& )
\EndDraw
```