

# TeXProject<sup>1</sup>: A Mini Manual

Stéphane Bressan      Eitan M Gurari

12 May 1995  
Last revised, February 2001

<sup>1</sup>©Stéphane Bressan (Stephane.Bressan@ecrc.de) and Eitan Gurari (gurari@cis.ohio-state.edu)

<b>1</b>	<b>Getting Started</b>	<b>3</b>
<b>2</b>	<b>Project Definition</b>	<b>4</b>
<b>3</b>	<b>Debugging Tools</b>	<b>5</b>
<b>4</b>	<b>Text-Producing Commands</b>	<b>6</b>
<b>5</b>	<b>Diagrams and Charts Commands</b>	<b>7</b>
5.1	Tree Diagrams . . . . .	7
5.2	Pert Diagrams . . . . .	8
5.3	Effort Charts . . . . .	9
5.4	Schedule Charts . . . . .	10
5.5	User-Defined Axes . . . . .	10
5.6	(A)DraTeX Commands of Immediate Interest . . . . .	11
<b>6</b>	<b>Data Views</b>	<b>12</b>
6.1	Split Preprocessing . . . . .	12
6.2	Processing Within Restricted Environments . . . . .	12
6.3	Dates . . . . .	12
6.4	Dates Projects . . . . .	13
6.5	Contributors Projects . . . . .	13
<b>7</b>	<b>Programming Tools</b>	<b>14</b>
7.1	Tag Operations . . . . .	14
7.2	Set Operations . . . . .	14
<b>8</b>	<b>An Example</b>	<b>16</b>
8.1	Project Definition . . . . .	16
8.2	Text Output . . . . .	17
8.3	Graphical Output . . . . .	18
8.3.1	Tree Diagram . . . . .	18
8.3.2	Pert Diagram . . . . .	19
8.3.3	Effort Chart . . . . .	20
8.3.4	Schedule Chart . . . . .	23
8.4	Data Views . . . . .	25
8.4.1	Dates . . . . .	25

8.4.2	Contributors . . . . .	27
8.4.3	Tag and Set Operations . . . . .	28
8.5	Beyond the TeXProject Native Features . . . . .	32
8.5.1	Tree Diagrams . . . . .	32
8.5.2	Schedule Charts . . . . .	33
<b>9</b>	<b>Known Problems</b>	<b>35</b>

# 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 extendible 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 of a project definition are reusable in the text. To each element corresponds a primitive:

- `\annotation{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 through a mapping from the contributor (`#1`) and the effort (`#2`), and how they should be separated. 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 (A)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}`

- This instruction determines how content of nodes should be typeset with a mapping from #1 (a tag) to `style`.

Default: `\treenodestyle{\name{#1}}`

## 5.2 Pert Diagrams

`\pertdiagram(tag)`

- The command draws a diagram which takes into account both predecessors and time constraints.
- A start ‘\*’ following the control sequence asks for diagrams which take into account only predecessors constraints.

`\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)`  
`\pertlimits(rows, columns, boundary-mark, code)`

- `rows`, `columns`: : integer numbers requesting the decomposition of the diagrams into subdiagrams, when the estimated number of rows or columns is greater than the given limits. (The numbers are estimates, because rows and columns are only vaguely defined in, respectively, vertical and horizontal diagrams.)
- `boundary-mark`: : a command for typesetting nodes referenced out of the subdiagrams; it is given a parenthesized tag for an argument. Two variants, named `\ExtNode` and `\PutNode`, are predefined in the package.
- `code`: : a contribution to be inserted between the subdiagrams of a decomposed diagram.
- empty arguments ask to preserve the status quo.

Default: `\pertlimits(, , \ExtNode, \par)`  
`\adjustpert(tag_1, distance_1 & ... & tag_n, distance_n )`

- This command allows for user adjustments to the positions of specified nodes. The adjustments assume point units of measurements, and they are perpendicular to the direction of the diagram: a negative value to move to the left (top), and a positive value to move to the right (bottom).

`\pertnodestyle(style)`

- Determines how content of nodes should be typeset, with a mapping from #1 (a tag) to `style`.

Default: `\pertnodestyle{\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:  $0 \leq \text{from} < \text{to} \leq 10$ ;
- `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)`

`\effortlimits(limit, code)`

- `limit`: : an integer number requesting the decomposition of the chart into sub-charts, when the number of bars is greater than the given limit.
- `code`: : an optional contribution to be inserted between the subcharts of a decomposed chart.

Default: `\effortlimits(, \par)`

## 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:  $0 \leq \text{from} < \text{to} \leq 10$ ;
- `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)`

`\schedulelimits(limit, code)`

- `limit`: : an integer number requesting the decomposition of the chart into subcharts, when the number of bars is greater than the given limit.
- `code`: : an optional contribution to be inserted between the subcharts of a decomposed chart.

Default: `\schedulelimits(, \par)`

## 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 AIDraTeX 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.

# Chapter 6

## Data Views

### 6.1 Split Preprocessing

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

The `\putcommand` instruction may be used for extracting the outcome of the command most recently preprocessed by `\filtercommand`.

### 6.2 Processing Within Restricted Environments

The `\putcommand` instruction may be used for importing data into immediately processed environments.

**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}.
```

The commands `\annotation`, `\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 `\annotation`, `\name`, `\start`, and `\finish` within the drawing structures of  $TeXProject$ , and to the `\putcommand` command.

### 6.3 Dates

```
\setNumericDate\cs{date,annotation}
```

- `\cs` : a user-provided control sequence to be assigned a numeric value.

- `data` : : a tag or a date constant.
- `annotation` : : ‘start’, ‘finish’, or an empty string. The annotation is optional for milestone tags and date constants.

Examples:

**Example.**

```
\setNumericDate\Start{t1,start}
\setNumericDate\Finish{t1,finish}
\setNumericDate\Notice{m1}
\setNumericDate\Current{01/03/95}
```

## 6.4 Dates Projects

`\datesproject{tag}{name}{list of members}{description}`

- Creates a three-levels project hierarchy: the tag root, a level of dates, and a level of the given members.
- Each date-node has under it exactly one of the members.
- A milestone member ‘m’ has an ancestor date-node labeled `tag.m`.
- A non-milestone member ‘m’ has two date ancestor nodes named ‘`tag.m.1`’ and ‘`tag.m.2`’ representing, respectively, the start and end dates of the given member.
- The `\name{...}` values of the date nodes are numeral encodings of the dates
- The ‘`\annotation{...}`’ values are ‘start’, ‘finish’, or an empty string.

## 6.5 Contributors Projects

`\contributorsproject{tag}{name}{list of members}{description}`

- Creates a three-levels project hierarchy: the tag root, a level of contributors, and a level of the given members.
- Each contributor has under it the tasks which it supports.

## Chapter 7

# Programming Tools

The following commands allow the advanced programming of primitives' filtering, and 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 documentation.

### 7.1 Tag Operations

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

- `milestone command`, `project command`, and `task command` are user defined commands with parameters similar to those offered for `\defmilestone`, `\defproject`, and `\deftask`, respectively. The only exception is for the trailing description parameters, which are provided indirectly through names of macros that hold the parameters.
- `\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.

### 7.2 Set Operations

```
\sapply{tag set, general command}
```

- A variant of the `\apply` command in which the first parameter should be given a set of tags.

- A tag set is specified by a comma-separated list of tags delimited within braces, or by an identifier defined through the following commands.

```

\sapply{identifier, merge, set, set}
\sapply{identifier, intersect, set, set}
\sapply{identifier, subtract, set, set}
\sapply{identifier, milestones, tag set}
\sapply{identifier, projects, tag set}
\sapply{identifier, tasks, tag set}
\sapply{identifier, before, tag set}
\sapply{identifier, after, tag set}
\sapply{identifier, up, tag set}
\sapply{identifier, down, tag set}
\sapply{identifier, sort, tag set}
\sapply{tag, defproject, tag set, name, description}

```

The first argument is assigned the result of applying the specified operation. The operation is given by the second argument, and the operands by the remaining arguments.

- The operands of the `merge`, `intersect`, and `subtract` operations may be consisted of general sets of keywords, not just of tags.
- The following commands might become useful during debugging.

```

\sapply{traceon}
\sapply{traceoff}

```

- `\breadth` records the breadth of the set.

# 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, manuals 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 center.}

\defproject{t1}{Package}
  {t1.1,t1.2,t1.3}
```

```

    {m1}
    {You get and install the package.}

\deftask{t2.1}{Manual}
  {01/03/95}{04/03/95}
  {you:10}
  {t1.1}
  {You review the manual and examples.}

\deftask{t2.2}{Ask}
  {27/02/95}{04/03/95}
  {you:5,someone:5}
  {}
  {You communicate with others.}

\deftask{t2.3}{Experiment}
  {03/03/95}{04/03/95}
  {you:30}
  {t1.2}
  {You set up a small example.}

\defproject{t2}{Help}
  {t2.1,t2.2,t2.3}
  {m1}
  {You get to know the system.}

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

\defproject{texproject}{TeXProject}
  {m1,t1,t2,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, someone (\resource{texproject}), and a total amount of man-minutes of 150 (\effort{texproject}). You will be involved for 115 (\effort{texproject,you}) minutes.

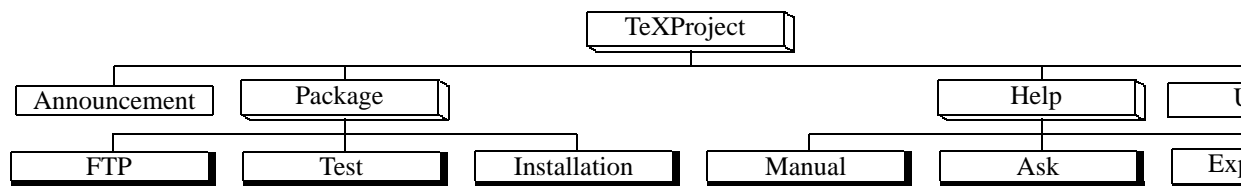
*The task Installation (\name{t1.3}) has the following description:* You ask your system engineer to install the package for your center. (\annotation{t1.3}).

### 8.3 Graphical Output

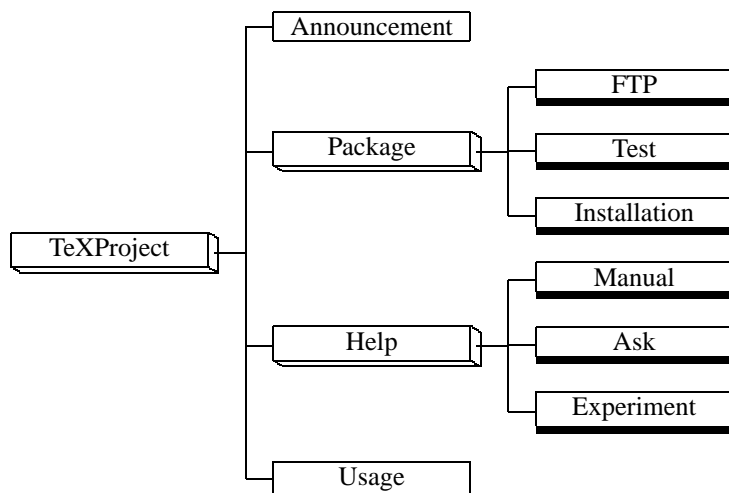
The following charts and diagrams is an example of graphical output using the project definition of section 8.1. The Charts and diagrams follow their generating commands.

#### 8.3.1 Tree Diagram

```
\Draw
\TreeSpace(C,10,10)
\treedigraph(texproject)
\EndDraw
```

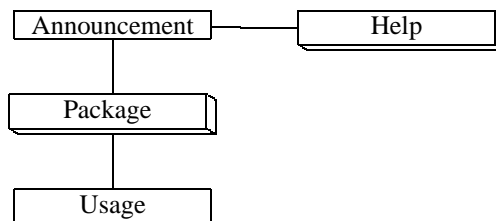


```
\Draw
\TreeAlign(H,-1,0)(0,0,0)
\treedigraph(texproject)
\EndDraw
```

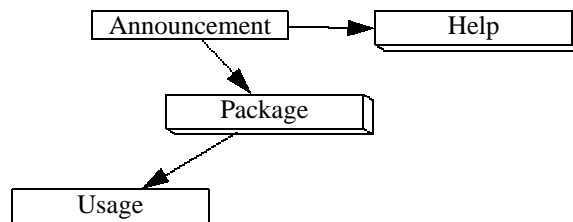


### 8.3.2 Pert Diagram

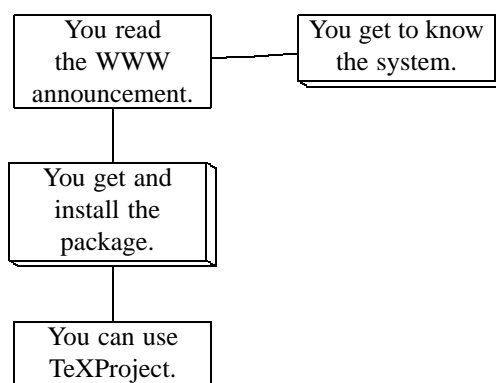
```
\Draw
\pertdiagram(texproject)
\EndDraw
```



```
\Draw
\ArrowHeads(1)
\adjustpert(t1,30 & m2,-30 )
\pertdiagram(texproject)
\EndDraw
```



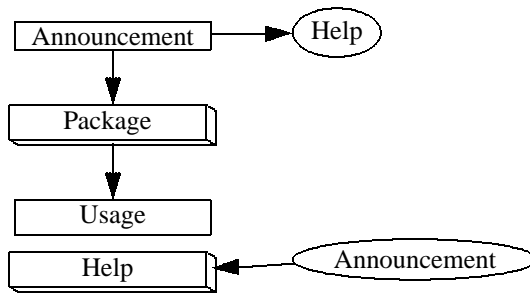
```
\Draw
\pertnodestyle{\annotation{#1}}
\pertdiagram(texproject)
\EndDraw
```



```

\Draw
  \ArrowHeads(1)
  \pertlimits(,1)
  \pertdiagram(texproject)
\EndDraw

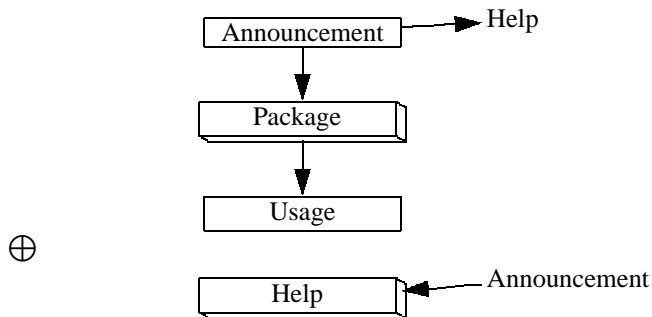
```



```

\Draw
  \ArrowHeads(1)
  \Define\externalnode(1){
    \filtercommand\name{#1}
    \Node(#1)(--\putcommand--)}
  \pertlimits(,1,\externalnode,\par$\bigoplus$\par)
  \pertdiagram(texproject)
\EndDraw

```

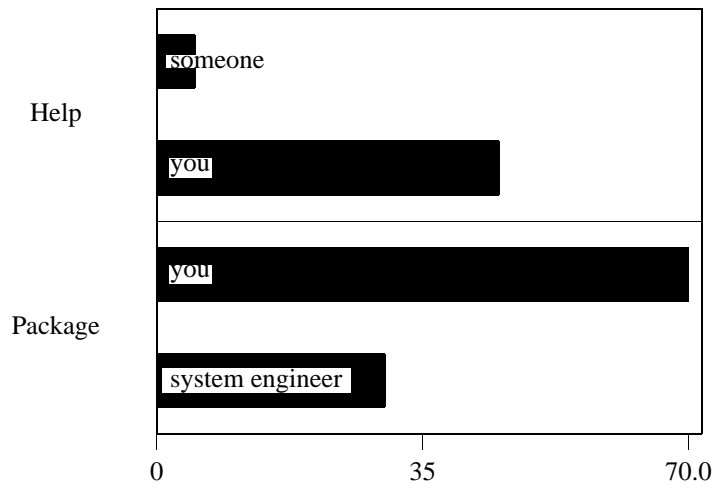


### 8.3.3 Effort Chart

```

\Draw
\effortchart(texproject)
\EndDraw

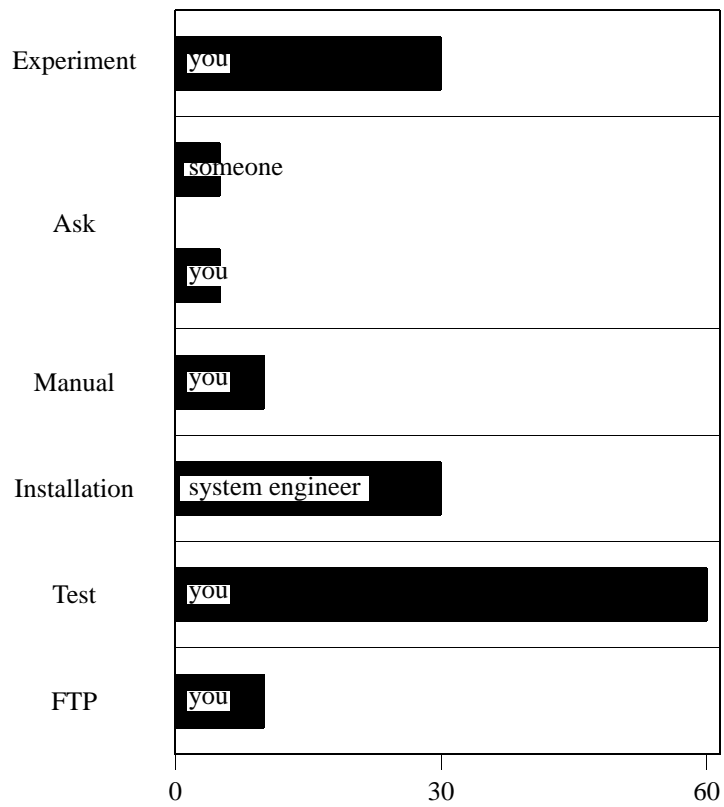
```



```

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

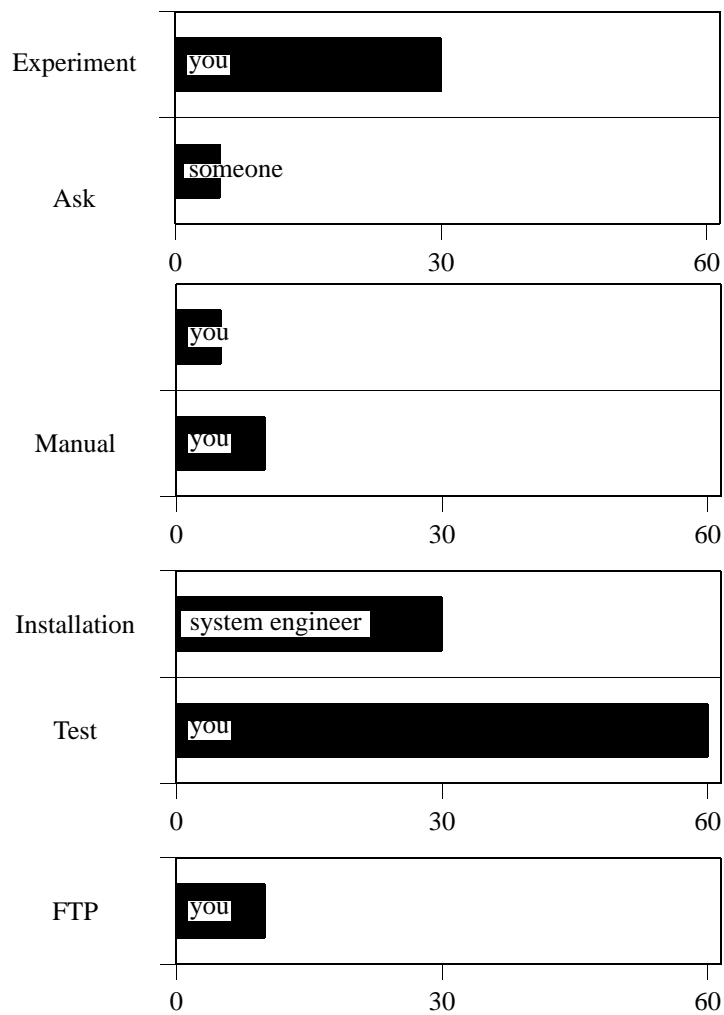
```



```

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

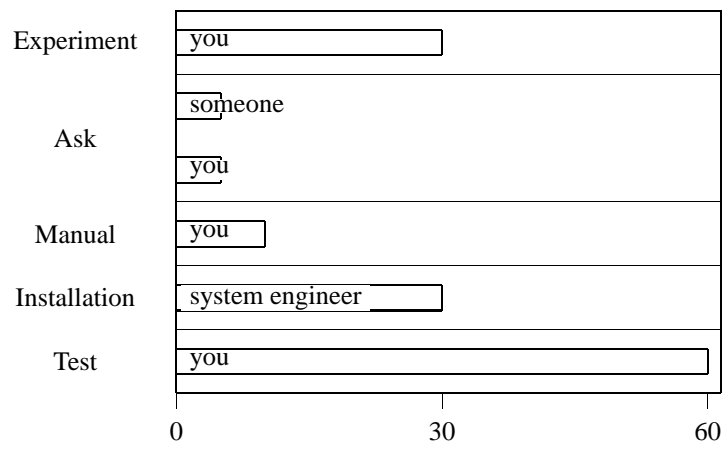
```



```

\Draw
\Scale(1,0.6)
\effortstyle(A,3,7,0,0,12)
\effortchart(texproject,02/03/95,03/03/95,,3)
\EndDraw

```

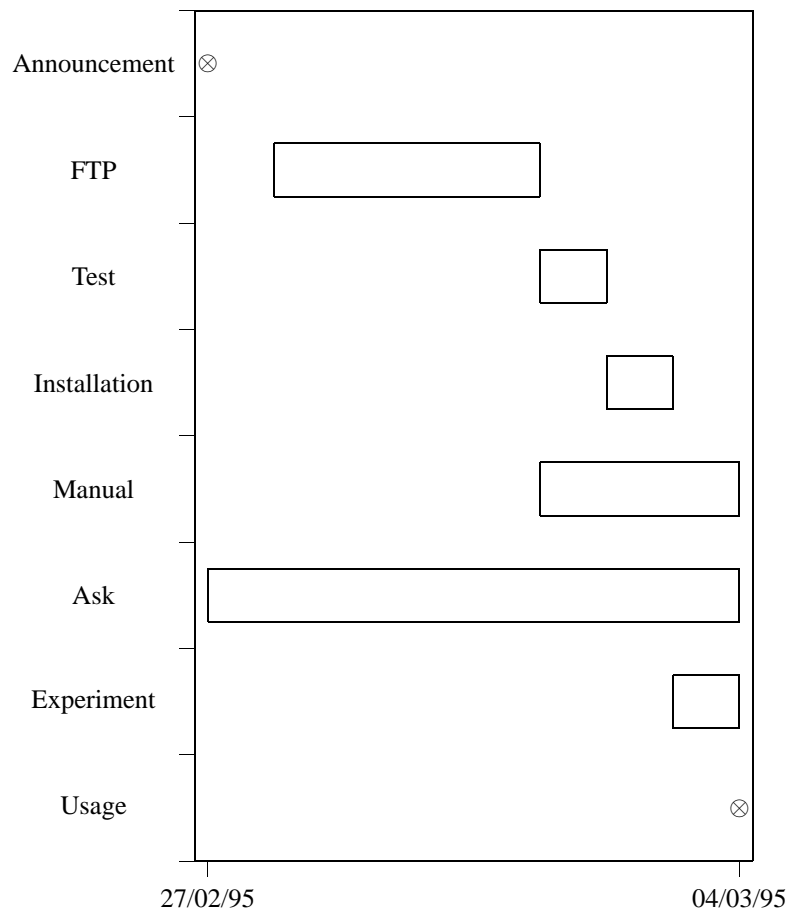


### 8.3.4 Schedule Chart

```

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

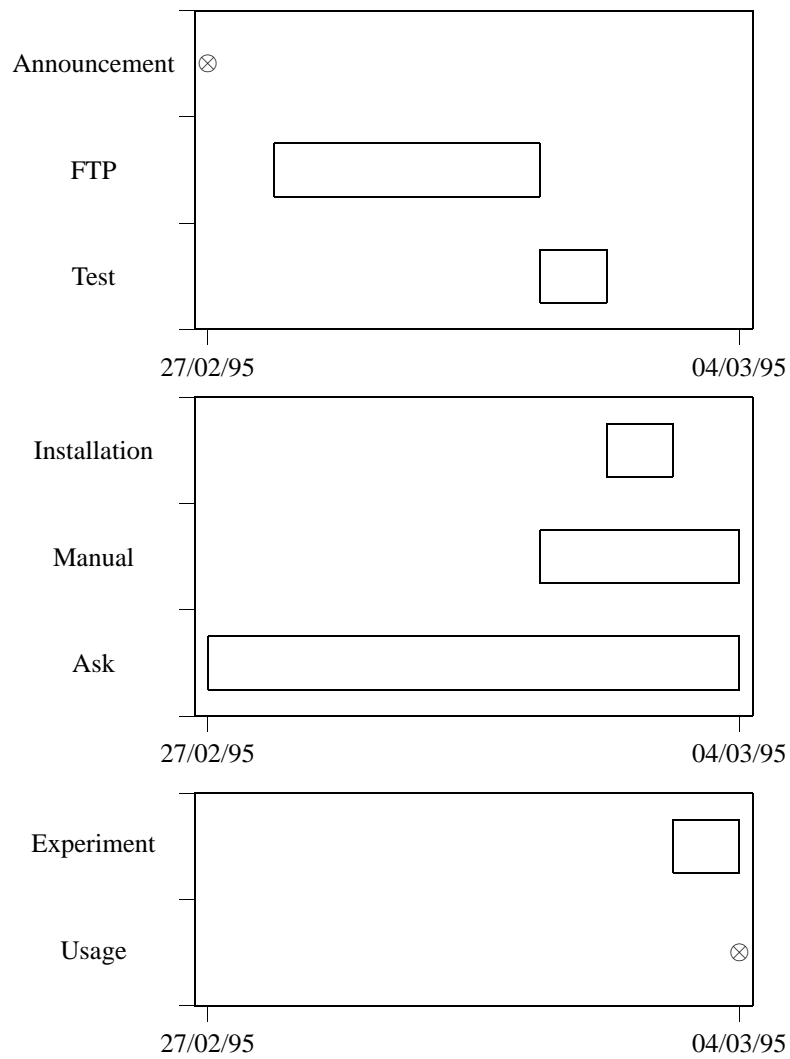
```



```

\Draw
\schedulelimits(3)
\schedulechart(texproject,,,,99)
\EndDraw

```



## 8.4 Data Views

### 8.4.1 Dates

```

\Draw
  \setnumericdate\Current{02/03/95}

  \pertenodestyle{%
    \setnumericdate\Finish{#1,finish}%
    \ifnum\Finish<\Current \Huge \bf
      \smallskip \name{#1} \smallskip
  }

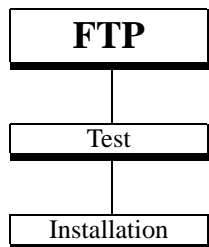
```

```

\else
  \name{#1}
\fi }

\pertdiagram(t1)
\EndDraw

```



```

\newcommand\showentry[1]{
  \sapply{xx,down,{#1}}
  \sapply{xx,tasks,xx}
  \filtercommand\start{#1}
  \sapply{xx,\item[\putcommand] \annotation{#1} \getname}
}
\newcommand\getname[1]{ \sapply{xx,\name} }

\datesproject{cal}{calendar}{texproject}{}
\sapply{xx,down,{cal}}
\begin{description}
  \sapply{xx,\showentry}
\end{description}

```

**27/02/95** start Ask

**28/02/95** start FTP

**01/03/95** finish FTP

**01/03/95** start Test

**01/03/95** start Manual

**02/03/95** finish Test

**02/03/95** start Installation

**03/03/95** finish Installation

**03/03/95** start Experiment

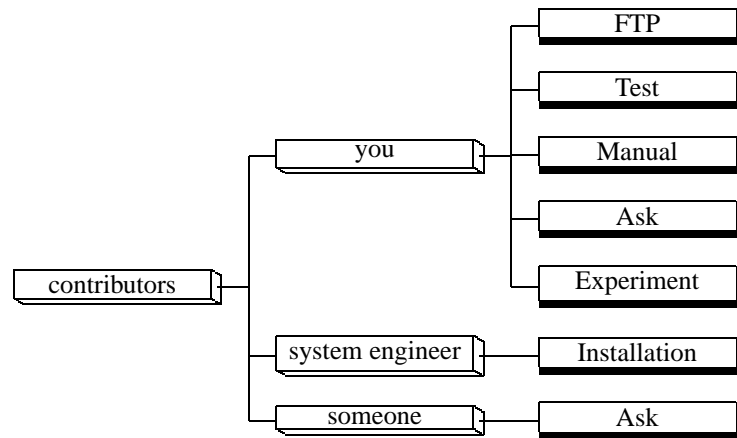
**04/03/95** finish Manual

04/03/95 finish Ask

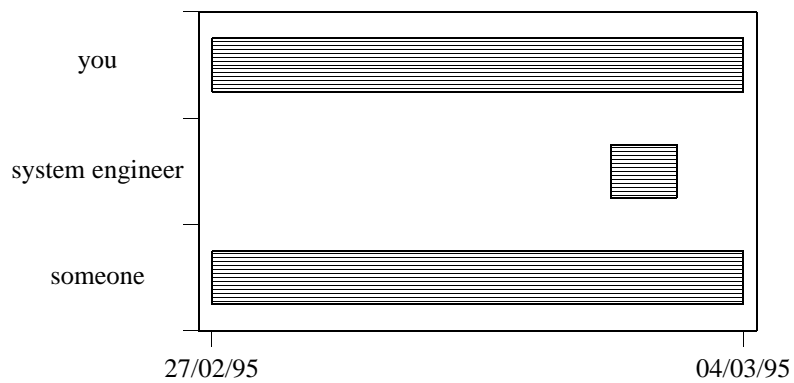
04/03/95 finish Experiment

### 8.4.2 Contributors

```
\contributorsproject{c}{contributors}{texproject}{description}  
\Draw  
\TreeAlign(H,-1,0)(0,0,0)  
\treediagram(c)  
\EndDraw
```



```
\Draw  
\schedulechart(c)  
\EndDraw
```



```
\newcommand\info[1]{  
  {\bf \name{#1}}: effort \effort{#1}, \start{#1}--\finish{#1}. }  
\newcommand\cinfo[1]{
```

```

\info{#1}
\sapply{tasks,down,{#1}}
\begin{enumerate}
\sapply{tasks,\tinfo}
\end{enumerate} }
\newcommand\tinfo[1]{ \item \info{#1} }

```

```

\sapply{contributors,down,{c}}
\sapply{contributors,\cinfo}

```

**you:** effort 120, 27/02/95–04/03/95.

1. **FTP:** effort 10, 28/02/95–01/03/95.

2. **Test:** effort 60, 01/03/95–02/03/95.

3. **Manual:** effort 10, 01/03/95–04/03/95.

4. **Ask:** effort 10, 27/02/95–04/03/95.

5. **Experiment:** effort 30, 03/03/95–04/03/95.

**system engineer:** effort 30, 02/03/95–03/03/95.

1. **Installation:** effort 30, 02/03/95–03/03/95.

**someone:** effort 10, 27/02/95–04/03/95.

1. **Ask:** effort 10, 27/02/95–04/03/95.

### 8.4.3 Tag and Set Operations

```

\newcommand\getparent[1]{
\sapply{p,up,{#1}}
\sapply{p,\name}
}
\newcommand\V[5]{\sapply{all,merge,all,{#1}}}
\newcommand\Vi[7]{\sapply{all,merge,all,{#1}}}
\sapply{all,merge,{},{}}
\rapply{texproject,\V,\V,\Vi}
\sapply{all,subtract,all,{texproject}}
\sapply{foo,defproject,all,,}

```

```

\Draw
\TreeAlign(H,-1,0)(0,0,0)
\treediagram(foo,1)
\EndDraw

```

```

\Draw

```

```

\ArrowHeads(1)
\pertnodestyle{\getparent{#1}:\break\hfil\name{#1}}
\pertdiagram *(foo)
\EndDraw

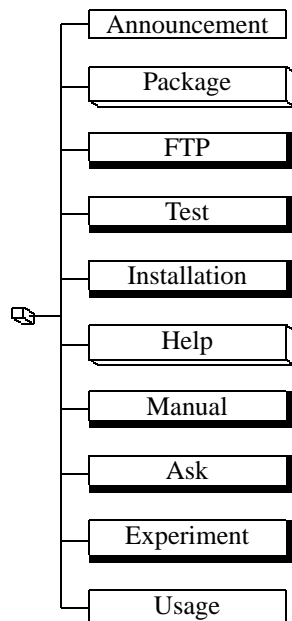
\smallskip \hrule\smallskip

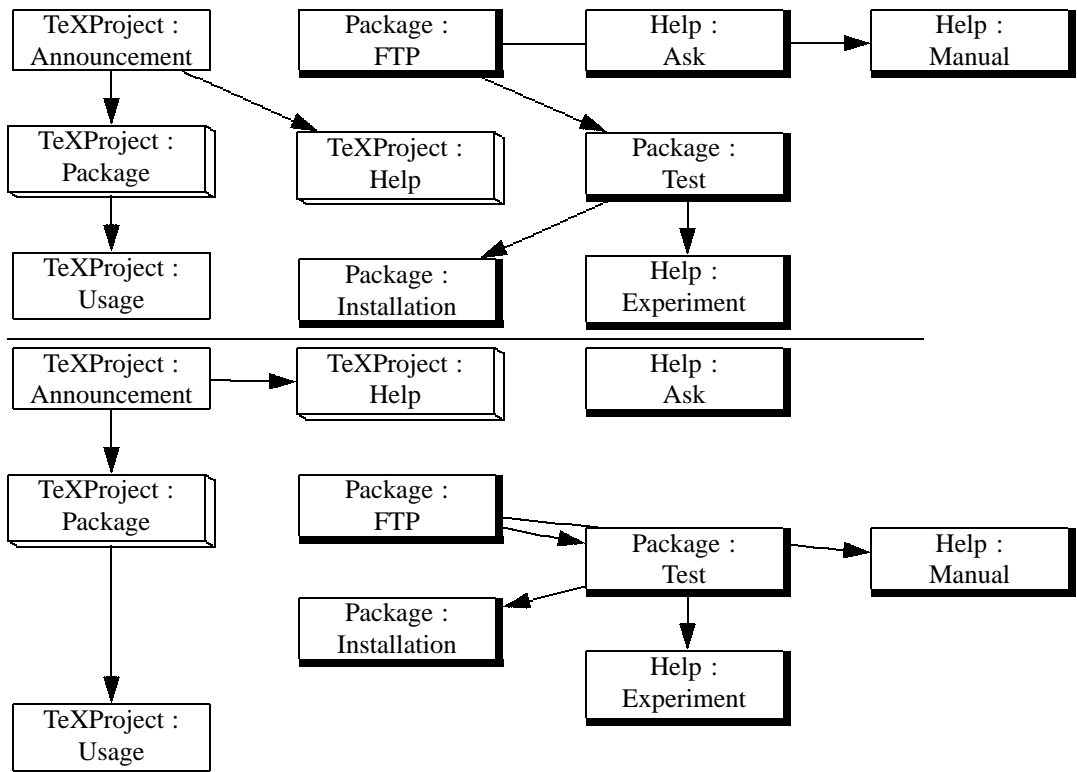
\Draw
\ArrowHeads(1)
\pertnodestyle{\getparent{#1}:\break\hfil\name{#1}}
\pertdiagram(foo)
\EndDraw

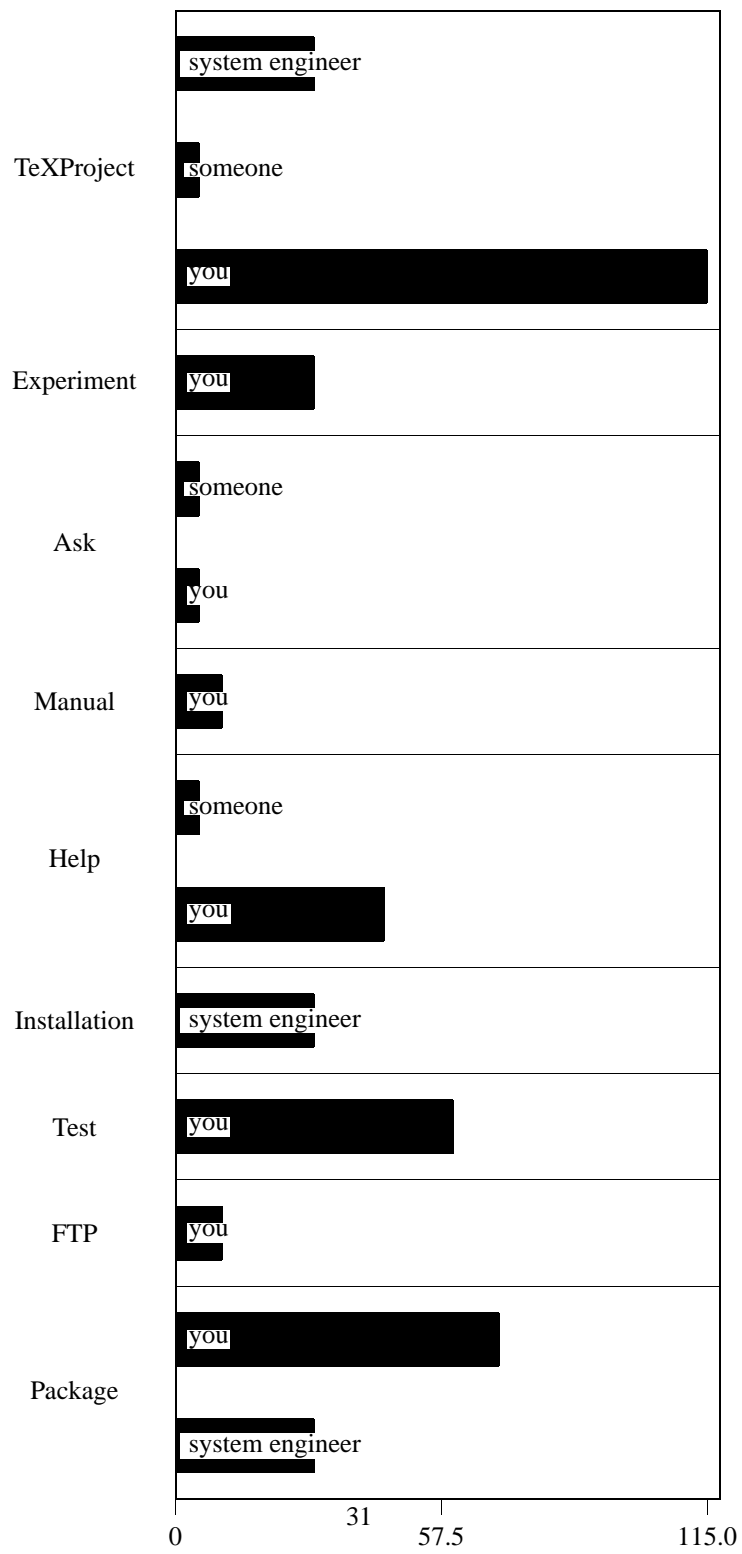
\Draw
\sapply{all,merge,all,{texproject}}
\sapply{foo,defproject,all,,}
\effortchart(foo)
\EndDraw

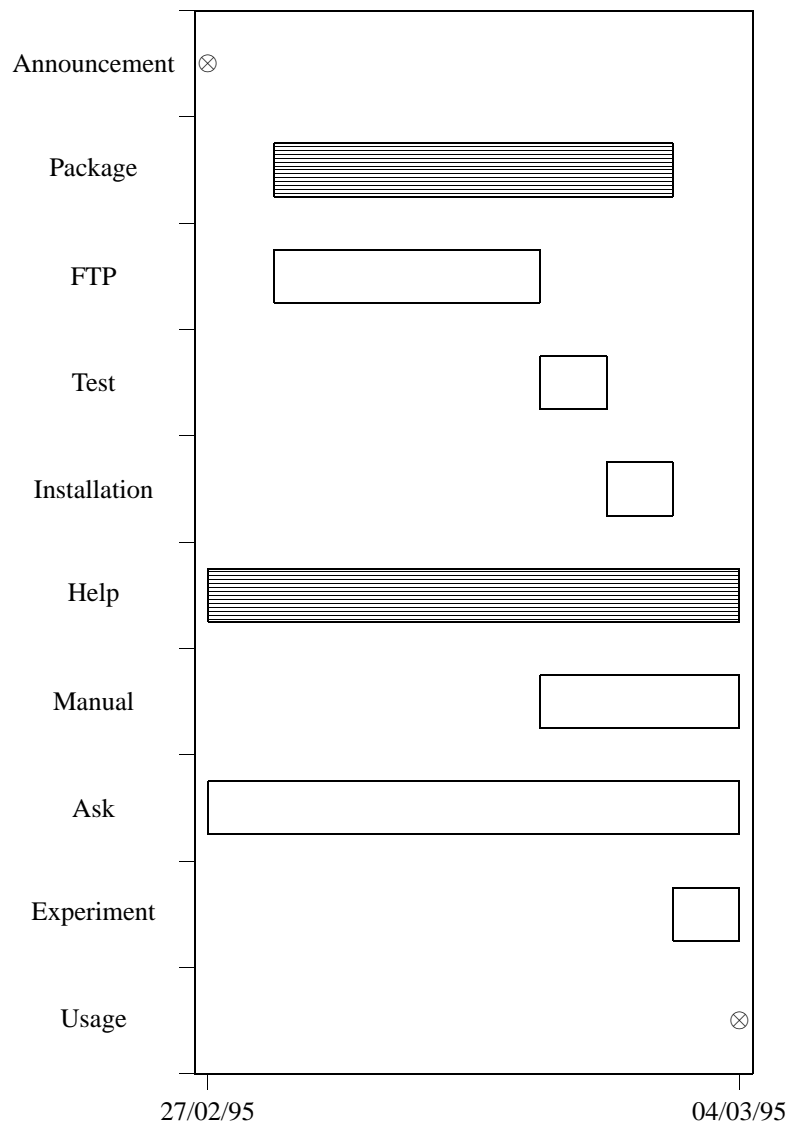
\Draw
\schedulechart(foo)
\EndDraw

```









## 8.5 Beyond the $\TeX$ Project Native Features

### 8.5.1 Tree Diagrams

```

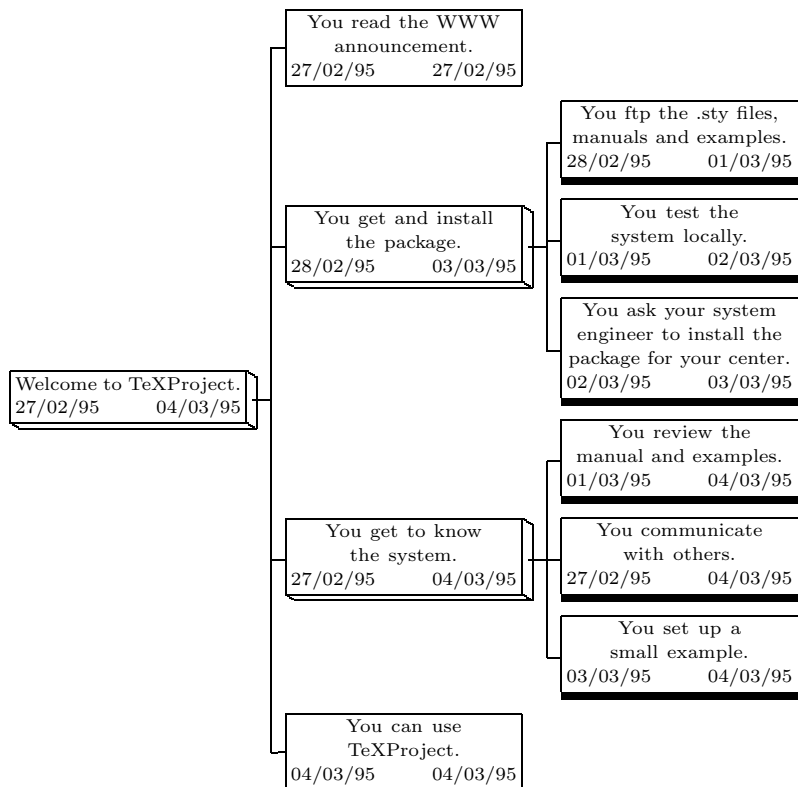
\Draw
\treenodestyle{\annotation{#1}\break \start{#1}\quad\hfill\finish{#1}}
\font\sevenrm=cmr7 \sevenrm
\baselineskip=9pt
\treestyle(,,,85)

```

```

\TreeAlign(H,-1,0)(0,0,0)
\TreeSpace(,5,10) \treediagram(texproject)
\EndDraw

```



## 8.5.2 Schedule Charts

```

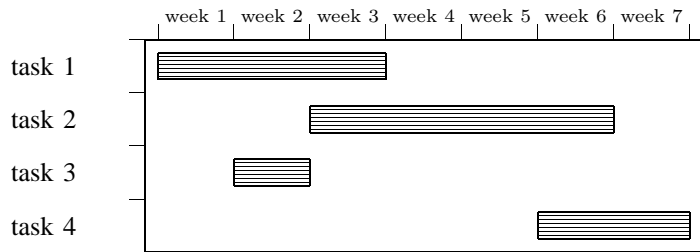
\newproject \minmaxdates{0/0/0//100/0/0}
\defproject{tasks}{{a,b,c,d}}{}
\deftask{a}{task 1}{0/0/0}{3/0/0}{}{}{}
\deftask{b}{task 2}{2/0/0}{6/0/0}{}{}{}
\deftask{c}{task 3}{1/0/0}{2/0/0}{}{}{}
\deftask{d}{task 4}{5/0/0}{7/0/0}{}{}{}
\endnewproject
\Draw \Scale(1,0.5) \schedulestyle(,,,3,,)
\schedulechart(tasks)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

```



## Chapter 9

# Known Problems

- The `\description` command got renamed to `\annotation` in January 2001.
- 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.
- The options parameter of the `\effortstyle` command needs fixing.
- The `\schedulestyle` needs more sophisticated labels. In particular, it should allow to move the left-side labels to the right, and put at the left group labels. (A possible application is a contributors' schedule chart which is divided by resources (persons), and for each person subdivided by tasks.)
- Checking for validity of a project data is a memory consuming process. If your environment is limited in memory, place the instructions `\warningsoff` and `\warningson` before and after, respectively, the `\newproject` and `\endnewproject` commands.

# Acknowledgment

We are very grateful to Sven Utcke for many valuable comments and suggestions.