The reqdoc package[*]

Thai Son Hoang
ETH-Zurich
<htson at inf dot ethz dot ch>

June 29, 2012

Abstract

This package provides macros for typesetting requirements documents. It was developed at the Swiss Federal Institute of Technology Zurich (ETH-Zurich).

Contents

1 Introduction

This package was developed in order to ease the typesetting of requirements documents in LaTeX.

2 Usage

Just like any other package, you need to request this package with a `\usepackage` command in the preamble. This package take some options on how requirements are presented, either normally (within some bounding box) or compactly (as a compact description list).

So in the simpler case, one just types

`\usepackage{reqdoc}`

to load the package with the default option (i.e. display requirements in normal size), or uses

`\usepackage[compact]{reqdoc}`

when the user wants to present the requirements documents in the compact form.

The rest of this section is to give descriptions of the main environment `requirements` and macros for typesetting different types of requirements.

`requirements` The requirements should be put within the `requirements` environment. The

[*]This document corresponds to reqdoc v1.2, dated 2012/06/27.

environment will be adapted accordingly to the package option, e.g. `compact`.

`\req` `\eqp` `\fun` `\sys` `\saf` `\env` `\asm` `\sched` `\alg`

To typeset a requirement, several macros are provided for different type of requirements. The signature of the requirement macros are as follows.

`\macroname[⟨width⟩]{⟨ref-label⟩}{⟨requirement⟩}`

The optional argument `[⟨width⟩]` defines the width for the requirement box in the default normal mode. This option is ignored in the compact mode. The default value for this optional argument is `0.6\textwidth`. The argument `{⟨ref-label⟩}` defines the reference label, by which the requirement can be referenced later using `\ref{ref-label}`. Note that `{⟨ref-label⟩}` is the LaTeX label for referencing, which is different from the automatically generated requirement label (discussed below). Finally the argument `\marg{requirement}` is some concise description test.

`\requirement@counter`

Each requirement are labelled automatically, where the labels consist of some prefix (depending on the type of the requirement), and some sequence number. The requirement sequence number are controlled by a counter `requirement@counter` which is advanced automatically after each requirement declaration.

Table 1 lists the macros, their intended purpose, and the prefixes. The convention for different types of requirements does not mean to be exhaustive or necessarily disjoint.

Macro	Purpose	Prefix
`\req`	General requirements	`REQ`
`\eqp`	Equipment requirements	`EQP`
`\fun`	Functional requirements	`FUN`
`\sys`	System requirements	`SYS`
`\saf`	Safety requirements	`SAF`
`\env`	Environment requirements	`ENV`
`\asm`	Assumptions	`ASM`
`\sched`	Scheduling requirements	`SCHED`
`\alg`	Algorithmic requirements	`ALG`

Table 1: List of macros for requirements

`\ReqSpacing`

An useful macro is `\ReqSpacing` to give some spacing between requirements within the `requirements` environment. This spacing will be ignored when the package option `compact` is enabled. The signature of the command is as follows.

`\ReqSpacing[⟨height⟩]`

where the optional argument `[⟨height⟩]` define the spacing gap (default `2ex`).

Below is a some sample requirements typeset using the newly defined environment and macros.

```
\begin{requirements}
  \asm[0.5\textwidth]{asm:instructors}
  {Instructors are members of the club.}
  \ReqSpacing
  \asm[0.5\textwidth]{asm:participants}
  {Participants are members of the club.}
  \ReqSpacing[4ex]
  \req{req:course-status}
  {A course is either \emph{opened} or \emph{closed}.}
  \ReqSpacing
  \req{req:open-course}{The system allows to open a closed course.}
  \ReqSpacing
  \req{req:close-course}{The system allows to close an opened course.}
\end{requirements}
```

In the default normal mode, the result looks like the following

ASM 1	Instructors are members of the club.

ASM 2	Participants are members of the club.

REQ 3	A course is either *opened* or *closed*.

REQ 4	The system allows to open a closed course.

REQ 5	The system allows to close an opened course.

When the package option `compact` is enabled. the result looks as follows.

ASM 1 Instructors are members of the club.

ASM 2 Participants are members of the club.

REQ 3 A course is either *opened* or *closed*.

REQ 4 The system allows to open a closed course.

REQ 5 The system allows to close an opened course.

The requirements can be referenced using the previously defined labels. An example is as follows.

```
There are two assumptions
\ref{asm:instructors}, \ref{asm:participants},
and three requirements \ref{req:course-status},
\ref{req:open-course}, \ref{req:close-course}.
```

There are two assumptions ASM 1, ASM 2, and three requirements REQ 3, REQ 4, REQ 5.

3 Implementation

The implementation is quite straightforward. We first request the `varioref` package for referencing the requirements later.

```
\RequirePackage{varioref}
```

Subsequently, we declare the option `[`⟨*compact*⟩`]` and redefine the requirements environment accordingly. . For the environment and macros that depending on the package options, we define different auxiliary versions of the environment and macros, and use the appropriate version depending on the package option. In our case, the environment `requirements`, macros `\requirement` and `\ReqSpacing` are depended on the package option.

`requirements` By default `requirements` environment is the same as `requirementsbox` environment (to be defined later).

```
\newenvironment{requirements}
  {\begin{requirementsbox}}
  {\end{requirementsbox}}
```

`\ReqSpacing` By default `\ReqSpacing` is the same as `\ReqSpacingBox`.

```
\newcommand{\ReqSpacing}[1][2ex]{\ReqSpacingBox[#1]}
```

`\requirement` By default `\requirement` is the same as `\requirementbox`.

```
\newcommand{\requirement}[3][0.6\textwidth]{
  \requirementbox[#1]{#2}{#3}
}
```

By declaring option `compact`, the previously defined environment and macros are redefined accordingly. In particular `\ReqSpacing` is simply ignored. These environment and macros are for implementation purpose only. The user should not use them directly.

```
\DeclareOption{compact}{
  \renewenvironment{requirements}
    {\begin{requirementscompact}}
    {\end{requirementscompact}}

  \renewcommand{\requirement}[3][0.6\textwidth]{
    \requirementcompact{#2}{#3}
  }

  \renewcommand{\ReqSpacing}[1][]{}
}
```

Afterwards, we process the options accordingly

```
\ProcessOptions
```

In the subsequent, we define the environment and macro for different package options.

`requirementsbox` We define environment `requirementsbox` is the same as environment `center`.

```
\newenvironment{requirementsbox}
  {\begin{center}}
  {\end{center}}
```

`requirementscompact` We define environment `requirementcompact` is the same as environment `description` with font `\small`

```
\newenvironment{requirementscompact}
  {\begin{description}\small}
  {\end{description}}
```

`\ReqSpacingBox` We define `\ReqSpacingBox` by passing the optional argument `[`⟨*height*⟩`]` to `\\`.

```
\newcommand{\ReqSpacingBox}[1][2ex]{\\[#1]}
```

`\requirementbox` The macro for typesetting a requirement in a box is as follows.

```
\newcommand{\requirementbox}[3][0.6\textwidth]{
  \medskip
  \refstepcounter{requirement@counter}
  \begin{tabular}{|@{\quad}c@{\quad}|@{\quad}c@{\quad}|}
    \hline
    & \\
    \textsf{#2~\arabic{requirement@counter}} &
    \begin{minipage}[c]{#1}
      \begin{center}
        #3
      \end{center}
    \end{minipage} \\
    & \\
    \hline
  \end{tabular}
}
```

`\requirementcompact` The macro for typesetting a requirement as a compact item is as follows.

```
\newcommand{\requirementcompact}[2]{
  \medskip
  \refstepcounter{requirement@counter}
  \item[\textsf{#1~\arabic{requirement@counter}}] #2
}
```

`requirement@counter` We define a counter for requirements, which will be used within the the labels of the requirements.

```
\newcounter{requirement@counter}
\labelformat{requirement@counter}{REQ~#1}
```

We define some specific commands for different type of requirements. These commands are those that the user should use to typeset the requirements. Each command associated with a new counter for each type of requirements and a unique label format. Each type of requirement use the underlying macro `\requirement` as defined earlier.

`reqc` `\req` Command for typesetting REQUIREMENT (typically prefixed with REQ).

```
\newcounter{reqc}
\labelformat{reqc}{REQ~#1}
\newcommand{\req}[3][0.6\textwidth]{
  \setcounter{reqc}{\value{requirement@counter}}
  \requirement[#1]{REQ}{#3}
  \refstepcounter{reqc}
  \label{#2}
}
```

eqpc \eqp Command for typesetting EQUIPMENT requirements (typically prefixed with EQP).

```
\newcounter{eqpc}
\labelformat{eqpc}{EQP~#1}
\newcommand{\eqp}[3][0.6\textwidth]{
  \setcounter{eqpc}{\value{requirement@counter}}
  \requirement[#1]{EQP}{#3}
  \refstepcounter{eqpc}
  \label{#2}
}
```

func \fun Command for typesetting FUNCTIONAL requirements (typically prefixed with FUN).

```
\newcounter{func}
\labelformat{func}{FUN~#1}
\newcommand{\fun}[3][0.6\textwidth]{
  \setcounter{func}{\value{requirement@counter}}
  \requirement[#1]{FUN}{#3}
  \refstepcounter{func}
  \label{#2}
}
```

sysc \sys Command for typesetting SYSTEM requirements (typically prefixed with SYS).

```
\newcounter{sysc}
\labelformat{sysc}{SYS~#1}
\newcommand{\sys}[3][0.6\textwidth]{
  \setcounter{sysc}{\value{requirement@counter}}
  \requirement[#1]{SYS}{#3}
  \refstepcounter{sysc}
  \label{#2}
}
```

safc \saf Command for typesetting SAFETY requirements (typically prefixed with SAF).

```
\newcounter{safc}
\labelformat{safc}{SAF~#1}
\newcommand{\saf}[3][0.6\textwidth]{
  \setcounter{safc}{\value{requirement@counter}}
  \requirement[#1]{SAF}{#3}
  \refstepcounter{safc}
  \label{#2}
}
```

envc \env Command for typesetting ENVIRONMENT requirements (typically prefixed with ENV).

```
\newcounter{envc}
\labelformat{envc}{ENV~#1}
\newcommand{\env}[3][0.6\textwidth]{
  \setcounter{envc}{\value{requirement@counter}}
  \requirement[#1]{ENV}{#3}
  \refstepcounter{envc}
  \label{#2}
}
```

`asmc` `\asm` Command for typesetting ASSUMPTIONS (typically prefixed with ASM).

```
\newcounter{asmc}
\labelformat{asmc}{ASM~#1}
\newcommand{\asm}[3][0.6\textwidth]{
  \setcounter{asmc}{\value{requirement@counter}}
  \requirement[#1]{ASM}{#3}
  \refstepcounter{asmc}
  \label{#2}
}
```

`schedc` `\sched` Command for typesetting SCHEDULING requirements (typically prefixed with SCHED).

```
\newcounter{schedc}
\labelformat{schedc}{SCHED~#1}
\newcommand{\sched}[3][0.6\textwidth]{
  \setcounter{schedc}{\value{requirement@counter}}
  \requirement[#1]{SCHED}{#3}
  \refstepcounter{schedc}
  \label{#2}
}
```

`algc` `\alg` Command for typesetting ALGORITHM requirements (typically prefixed with ALG).

```
\newcounter{algc}
\labelformat{algc}{ALG~#1}
\newcommand{\alg}[3][0.6\textwidth]{
  \setcounter{algc}{\value{requirement@counter}}
  \requirement[#1]{ALG}{#3}
  \refstepcounter{algc}
  \label{#2}
}
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.