

---

# New Control System for the IRAM 30m WG Software Engineering - Project ProjectDoc

---

**Title:** Project ProjectDoc: documentation of projects

**Identifier—MasterURL:** <http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/>

**File:** projectDoc

**Revision:** draft, 1.3

**Revision Date:** 2001-10-19

**Expiration Date:**

**Supersedes:** not applicable

**Is superseded by:** not applicable

**Authors:** W. Brunswig (brunswig@iram.es), A. Sievers (sievers@iram.es)

**Contributors:** Hans Ungerechts

**Affiliations:**

**Addresses:**

**Audience:** technical staff, astronomers

**Publisher:** IRAM, Granada, Spain

**Subject:** NCS 30m: Documentation Standard for Projects

**Keywords:** Project Documents, XML, PDF

## **Description - about this document:**

Describe a standard of project documentation. This standard will be used in the development of the IRAM 30m telescope New Control System. This document describes iteration 0 of the project.

We consider requirements and design to be done. The requirements that are open will not be implemented in this iteration. The format as described in the DTD file should be fixed now, but the XSL scripts to transform to PDF and HTML are not widely tested yet.

*Note the list of pending items.*

**References:**

See the list in Appendix [B](#).

**References and Related Documents—Short List:**

1. [the projectDoc DTD file](#)
2. [template of projectDoc files](#)
3. [template make file for projectDoc](#)

**History of this Document:**

1. v 1.1, 2001-04-04 wb: describe ideas about project documentation
2. v 1.2, 2001-08-08 wb: review of concepts, note possible extension, modifications
3. v 1.3, 2001-10-17 wb: adapted to agreements in meeting (WB,HU,AS), related to v1.0 of dtd, document iteration 0 of project.

**Pending Items:**

1. work out user guide

## Contents

<b>1</b>	<b>Project Plan ( 2001-10-19)</b>	<b>5</b>
<b>2</b>	<b>Introduction ( )</b>	<b>5</b>
<b>3</b>	<b>Requirements</b>	<b>5</b>
3.1	* sections ( <i>state: done 2001-09-17</i> ) . . . . .	5
3.2	* subSections ( <i>state: done 2001-09-17</i> ) . . . . .	6
3.3	* identifiers ( <i>state: done 2001-09-17</i> ) . . . . .	6
3.4	* format ( <i>state: done 2001-08-08</i> ) . . . . .	6
3.5	attributes ( <i>state: done 2001-04-04</i> ) . . . . .	6
3.6	swConcepts ( <i>state: done 2001-04-04</i> ) . . . . .	6
3.7	textStructure ( <i>state: done 2001-04-04</i> ) . . . . .	7
3.8	* acronyms ( <i>state: open 2001-09-17</i> ) . . . . .	7
3.9	* index ( <i>state: open 2001-09-17</i> ) . . . . .	7
3.10	hyperText ( <i>state: done 2001-04-04</i> ) . . . . .	7
3.11	* requirementGroups ( <i>state: open 2001-08-09</i> ) . . . . .	7
3.12	* template ( <i>state: done 2001-09-17</i> ) . . . . .	7
3.13	* outputPDF ( <i>state: done 2001-08-08</i> ) . . . . .	7
3.14	* outputHTML ( <i>state: done 2001-08-08</i> ) . . . . .	7
3.15	extractUserDoc ( <i>state: open 2001-04-04</i> ) . . . . .	8
3.16	* extractModifications ( <i>state: open 2001-09-17</i> ) . . . . .	8
3.17	emacsSupport ( <i>state: open 2001-04-04</i> ) . . . . .	8
3.18	* verification ( <i>state: done 2001-09-19</i> ) . . . . .	8
<b>4</b>	<b>Design</b>	<b>8</b>
4.1	* sections ( <i>state: done 2001-09-17</i> ) . . . . .	8
4.2	format ( <i>state: done 2001-04-04</i> ) . . . . .	9
4.3	* subSections ( <i>state: done 2001-09-18</i> ) . . . . .	9
4.4	* attributes ( <i>state: done 2001-09-17</i> ) . . . . .	10
4.5	* specifications ( <i>state: done 2001-09-17</i> ) . . . . .	10
4.6	* requirementAttributes ( <i>state: done 2001-09-17</i> ) . . . . .	10
4.7	* swConcepts ( <i>state: done 2001-09-18</i> ) . . . . .	11
4.8	output ( <i>state: done 2001-04-04</i> ) . . . . .	11
4.9	* outputPDF ( <i>state: done 2001-09-17</i> ) . . . . .	11
4.10	* outputHTML ( <i>state: done 2001-09-17</i> ) . . . . .	11
4.11	* tests ( <i>state: done 2001-08-09</i> ) . . . . .	12
4.12	* verification ( <i>state: done 2001-09-19</i> ) . . . . .	12
<b>5</b>	<b>Implementation</b>	<b>12</b>
5.1	* format ( <i>state: open 2001-09-18</i> ) . . . . .	12
5.2	* sections ( <i>state: open 2001-09-18</i> ) . . . . .	12
5.3	* subSections ( <i>state: open 2001-09-18</i> ) . . . . .	12

5.4	* attributes ( <i>state: open 2001-09-18</i> ) . . . . .	12
5.5	* swConcepts ( <i>state: open 2001-09-18</i> ) . . . . .	13
5.6	* textStructure ( <i>state: open 2001-09-18</i> ) . . . . .	13
5.7	* hyperText ( <i>state: done 2001-09-18</i> ) . . . . .	13
5.8	* template ( <i>state: done 2001-09-17</i> ) . . . . .	13
5.9	* outputPDF ( <i>state: done 2001-09-17</i> ) . . . . .	13
5.10	* outputHTML ( <i>state: done 2001-09-17</i> ) . . . . .	13
<b>6</b>	<b>Installation</b>	<b>13</b>
6.1	* hosts ( <i>state: done 2001-10-19</i> ) . . . . .	13
6.2	* requiredSw ( <i>state: done 2001-10-19</i> ) . . . . .	13
6.3	* homeMadeSw ( <i>state: done 2001-10-19</i> ) . . . . .	14
6.4	* verification ( <i>state: done 2001-09-19</i> ) . . . . .	14
<b>7</b>	<b>User Guide (2001-09-19)</b>	<b>14</b>
<b>A</b>	<b>List of Requirements/Specifications and Descendants</b>	<b>15</b>
<b>B</b>	<b>References</b>	<b>17</b>

## 1 Project Plan ( 2001-10-19)

1. 2001-10-19: distribute documentation of iteration 0 and request comments. (wb)
2. 2001-10-31: release iteration 0. (wb)
3. in 2002: review project, based on experience with iteration 0 and produce iteration 1. (wb, ?)

## 2 Introduction ( )

A project typically goes through a sequence of phases:

1. first ideas and top-level requirements
2. detailed specifications
3. design
4. implementation
5. module tests
6. installation (deployment)
7. integration tests
8. maintenance
9. out-phasing

The sequence might be iterated several times (except for the last phase). Although the sequence of phases can be found in many areas we here concentrate on software projects. Many software projects at the IRAM, Spain, are of a small scale of a few Man\*Weeks. These smaller projects are often not well documented. The scheme presented could be use to document the various development steps.

This version of this document relates to v1.0 of the projectDoc format. The syntax of projectDoc files is defined in the projectDoc DTD file: <http://www.iram.es/IRAMES/documents/ncs30mProj>. Its semantic is described below.

Project documents might not handle all details. For small projects most of the details could be included in the project document (to avoid several small or tiny documents). However, for larger project details should be handled in separate documents.

## 3 Requirements

### 3.1 \* sections (*state: done 2001-09-17*)

A project documents shall consists of a list of sections:

1. standard document "meta" information
2. document history

3. one chapter for each of the software development phases: requirements, design, implementation, installation
4. a project plan
5. a project logbook: lists important events
6. an operation manual (user guide)

### **3.2 \* subSections (*state: done 2001-09-17*)**

Each section is related to a specific software development phase. It shall contain a list of items ("subsections") describing details, e.g. each requirement shall be written into one requirement subsection.

### **3.3 \* identifiers (*state: done 2001-09-17*)**

Each subsection shall have an identifier. If a specific requirement is covered by just one subsection of the following phases (design, implementation, installation) then those subsection shall have the same identifier.

### **3.4 \* format (*state: done 2001-08-08*)**

The overview document shall be formatted such that different items and concepts are identified by tags.

Reason: Using tags allows to extract in a systematic way information. E.g. Requirements are enclosed in tags: <req> and </req>. Although project documents will contain chapters on all steps of development, extracting a listing of requirements can easily done by using the tags.

### **3.5 attributes (*state: done 2001-04-04*)**

Items like requirements or tests shall have attributes that indicate important information. The attributes of the tags depend on the type of tag.

Example:

1. Requirement attributes: state (open, accepted,...), implementation (done, started,...)
2. Implementation attributes: date, platform (host)

The details shall be defined in the design.

### **3.6 swConcepts (*state: done 2001-04-04*)**

The language shall include a set of tags for the standard concepts of software: file, host, port, socket, message, error, ... .

### 3.7 textStructure (*state: done 2001-04-04*)

The language shall include a set of tags to define the text structure: paragraphs, lists, emphasized text, ... .

### 3.8 \* acronyms (*state: open 2001-09-17*)

The language shall have a tag to define acronyms. The definition of acronyms can be done in the same file or in different document. The "output" versions of the document shall contain lists of acronyms. **Note:** *Not implemented in iteration 0.*

### 3.9 \* index (*state: open 2001-09-17*)

The language shall have a tag to register document positions in an index. **Note:** *Not implemented in iteration 0.*

### 3.10 hyperText (*state: done 2001-04-04*)

The language shall include tags for hypertext references to external documents.

### 3.11 \* requirementGroups (*state: open 2001-08-09*)

Allow to structure larger sets of requirements by defining groups (subsections) of requirements, e.g., group of requirements related to monitor and control. **Note:** *Not implemented in iteration 0.*

### 3.12 \* template (*state: done 2001-09-17*)

Provide a file that contains a template of a project document.

### 3.13 \* outputPDF (*state: done 2001-08-08*)

The original document shall be transformed to PDF.

Reason: The original version of the document uses tags and is therefore not "nice" to read. PDF documents produce an acceptable output and have also basic hypertext features.

### 3.14 \* outputHTML (*state: done 2001-08-08*)

The original document shall be transformed to HTML.

Reason: As HTML has more advanced hypertext features it might be produced as well if these features are wanted.

**Note:** *Should we maintain HTML besides PDF? HU: Maybe not in this case, but for "Help"/User Documentation.*

### 3.15 **extractUserDoc** (*state: open 2001-04-04*)

A tool shall allow to extract information for a user. This shall consist of sections:

1. Introduction
2. Requirements
3. UserGuide

**Note:** *Not implemented in iteration 0.*

### 3.16 \* **extractModifications** (*state: open 2001-09-17*)

A tools shall allow to extract those subsections (or sections) that have been modified since a given date. The standard output versions of the document shall also indicate modifications from the last to the actual version of the document.

**Note:** *Not implemented in iteration 0.*

### 3.17 **emacsSupport** (*state: open 2001-04-04*)

Editing project documents shall be supported by a particular emacs mode.

### 3.18 \* **verification** (*state: done 2001-09-19*)

A tool shall be provided that check if a projectDoc document is in conformance with the DTD defining the language syntax.

## 4 Design

### 4.1 \* **sections** (*state: done 2001-09-17*)

A project document will have the following sections:

1. meta: information about the document. It consists of most of the items that are used for standard documents of the IRAM new control system.
2. pending: important information about pending items (e.g. unfinished installation tests (ok, this should not happen))
3. history: document history
4. introduction: project background, basic ideas, etc.
5. requirements: list project requirements (what shall be done)
6. specifications: list specifications
7. design: list design decision (how things will be done)
8. implementation: information about implementation, links to further documentation (e.g. code documentation) (what has been done)

9. installation:

- (a) can give details on installed requirements (in particular, if not all requirements are implemented/installed at the same time.
- (b) describe how to install the software (e.g. which configuration files have to be modified).
- (c) list on which systems the software is installed

10. project log: a logbook of important project events

11. userGuide: can contain information about

- (a) how to use the software or link to an external user manual.
- (b) which maintenance is needed.
- (c) which health-checks can be done by users

12. miscellaneous: can contain what does not fit somewhere else, like future ideas and plans.

## 4.2 format (*state: done 2001-04-04*)

The language to describe the software and software engineering concepts is based on XML. It defines the tags

1. to identify software engineering concepts: requirements, design, ... ,
2. to identify standard software/computer concepts: file, host, ... ,
3. to reference external documents, and
4. to do basic text structuring.

It also defines tag attributes and which type of tags can be "inside" other tags. E.g. tag <requirements> can be followed by a tag <req> but not by a tag <test>.

Why use XML ? XML allows to define a structured text-tagging language (give some items a meaning). It is becoming an important standard for web documents and tools are available (many as public-domain). Libraries for many languages exist: Java, C, TCL, Python. We do not know if Fortran utilities exists (besides using C libraries) but some basic routines have been written by ourselves.

## 4.3 \* subSections (*state: done 2001-09-18*)

Subsections are available for the sections requirements (tag: req), specifications (spec), implementation (impl), installation (inst).

#### 4.4 \* attributes (*state: done 2001-09-17*)

All top-level items (requirements, designs, ...) shall have the following attributes:

1. id: identifier of requirement, design, ...
2. state: state of item (open,done).
3. date: date of last modification: the date shall refer to last modification of requirement or any stage with same identifier

The state of all items (requirements, design, ...) is described by an attribute state with can take one of these values:

1. notStarted: work has not yet started
2. open: work has started, but not yet finished
3. done: this is the "final" version (of this "iteration")

**Note:** *Usage of state attribute is different for requirements, see requirementAttributes*

#### 4.5 \* specifications (*state: done 2001-09-17*)

The first phase of a project consists in defining the requirements of the project. This normally starts with a list a unprecise requirements. Reviews and discussions between "customer" and "provider" shall lead to a list of precise requirements that both parts agree upon.

The following steps in the software development can also reveal deficiencies of the list of requirements and thus lead to new and modified requirements.

We can also distinguish between "user" requirements and those requirements that are derived by an analysis of the "user" requirements. Requirements that result from the analysis of the "user" requirements are also called specifications.

**Note:** *Note: specifications shall not use identifiers used for a requirement*

#### 4.6 \* requirementAttributes (*state: done 2001-09-17*)

For the requirements, we use the state attribute also to indicate the state of the "realization" of a requirement:

1. notStarted, open: same meaning as above
2. fixed: the requirement has been fixed (for this "iteration")
3. ...ing: current state of "realization" (designing, implementing,installing,testing)
4. ...ed: the state has been finished, next state not yet started (designed,implemented,installed)
5. done: the requirement has been "realized": designed, implemented, installed, and tested.

Requirements can have the following additional attribute:

1. iter: indicate in which software process iteration the requirement shall be implemented.
2. obsolete="yes": indicate that requirements is not valid anymore, but keep it, possibly with an explanation why it has been dropped.
3. prio essential, high, low: indicate priority of requirement.
  - (a) essential: requirement has to be implemented in first iteration.
  - (b) high: requirement shall be implement after all essential requirements have been implemented.
  - (c) low: requirement is considered to be useful, but implementation shall be done after implementing all requirements with high priority.
4. from: who made the requirement

**Note:** *Question: Shall we use different states/dates referring to the requirement itself and any other stage with same identifier ?*

#### 4.7 \* swConcepts (*state: done 2001-09-18*)

Tags will be defined for the most common concepts: file, host, process. Other concepts can be specified by a general tag swItem. See DTD for defined attributes.

#### 4.8 output (*state: done 2001-04-04*)

Document transformation will be developed from projectDoc language (in XML) to HTML and TeX/PDF in the XSL language.

Reason: Why use XSL ? For the translation several possible standards exist: XSL, DSSSL, CSS. XSL is a standard set up by the www consortium (w3c.org) and is the most powerful language for this purpose. However, support of XSL by web browsers is currently still limited.

#### 4.9 \* outputPDF (*state: done 2001-09-17*)

Project documents will be translated into PDF. This is done by

1. transform the documents from XML to PDFLaTeX (using XSL), and then
2. use PDFLaTeX to create a PDF version (using macros developed by HU)

#### 4.10 \* outputHTML (*state: done 2001-09-17*)

A script will be written in XSL to translate the projectDoc language into HML. The generated document will contain:

1. all sections. for those items that have a date attribute it will be indicated if the item has been modified or added since the last revision of the document
2. a contents table consisting of sections and subsections

3. a list of requirements, their attributes and other items with the same identifier (descendants)
4. a list of references: hyperText and swConcept elements

#### **4.11 \* tests (*state: done 2001-08-09*)**

Testing can be done at various stages of the development. The documentation of those tests depends on the development stage:

1. module tests are done during the implementation phase. They are described either in the project doc or as part of the source code documentation.
2. integration tests check the interface of and cooperation between modules. They are described in the implementation section.
3. system tests check the complete system after installation. They are also described in the installation section.
4. health check tests and tests to identify errors are described in the user guide.

#### **4.12 \* verification (*state: done 2001-09-19*)**

a (pd) tool xmlv check conformance of documents with regard to a DTD. this tools will be provided on gra-lx1 .

## **5 Implementation**

### **5.1 \* format (*state: open 2001-09-18*)**

The projectDoc language is based on XML. Its syntax is defined by the general XML rules and the specific language defined in :

<http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/project.dtd> .

### **5.2 \* sections (*state: open 2001-09-18*)**

The required sections are defined in the DTD referenced above.

### **5.3 \* subSections (*state: open 2001-09-18*)**

The required sections are defined in the DTD referenced above.

### **5.4 \* attributes (*state: open 2001-09-18*)**

The required attributes are defined in the DTD referenced above.

### 5.5 \* swConcepts (*state: open 2001-09-18*)

The required tags are defined in the DTD.

### 5.6 \* textStructure (*state: open 2001-09-18*)

The required tags (ol, ul, li, p, pre) are defined in the DTD referenced above.

### 5.7 \* hyperText (*state: done 2001-09-18*)

The projectDoc DTD contains a tag to define references to other documents.

### 5.8 \* template (*state: done 2001-09-17*)

For a template of the current project "language" look at:

<http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/template.xml> .

### 5.9 \* outputPDF (*state: done 2001-09-17*)

A transformation script is defined produce a PDF document. The transformation script is written in XSL.

### 5.10 \* outputHTML (*state: done 2001-09-17*)

A transformation script is defined produce a PDF document. The transformation script is written in XSL.

## 6 Installation

### 6.1 \* hosts (*state: done 2001-10-19*)

All tools needed are available on [gra-lx1.iram.es](http://gra-lx1.iram.es) .

### 6.2 \* requiredSw (*state: done 2001-10-19*)

The following sw packages are used:

1. com.jelark.xml.sax: xml parser (pd: public domain)
2. com.jelark.xsl.sax: xsl transformer (pd)
3. pdflatex: LaTeX wit PDF extensions (pd)

### 6.3 \* homeMadeSw (*state: done 2001-10-19*)

1. LaTeX macros for projectDoc style (hu): in /usr/local/projectDoc
2. XSL scripts (wb): in /usr/local/projectDoc

### 6.4 \* verification (*state: done 2001-09-19*)

tools xmlv (xml verification) is available on gra-lx1  
in directory /usr/local/bin .

## 7 User Guide (2001-09-19)

How to write projectDoc documents:

1. get a basic understanding of xml type documents: syntax rules, tags, attributes, special characters
2. copy file <http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/template.xml> to your new document
3. check for allowed attributes in file <http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/projectTo.make>
4. set a link:project.dtd to /usr/local/projectDoc/projectDoc.dtd  
with command:

```
ln -s /usr/local/projectDoc/projectDoc.dtd projectDoc.dtd
```

5. insert file <http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/projectTo.make> into your make file and edit it.

Please note also:

1. if <ref> tags reference external documents giving a relative name are these names are preceded by the "masterURL" specified with tag <doc:masterURL> .
2. xml files can be printed "prettily" with:

```
a2ps -1 --delegate=no --pretty-print=html your-xml-file
```

## A List of Requirements/Specifications and Descendants

1.
  - req sections done 2001-09-17
  - des sections done 2001-09-17
  - impl sections open 2001-09-18
2.
  - req subSections done 2001-09-17
  - des subSections done 2001-09-18
  - impl subSections open 2001-09-18
3.
  - req identifiers done 2001-09-17
4.
  - req format done 2001-08-08
  - des format done 2001-04-04
  - impl format open 2001-09-18
5.
  - req attributes done 2001-04-04
  - des attributes done 2001-09-17
  - impl attributes open 2001-09-18
6.
  - req swConcepts done 2001-04-04
  - des swConcepts done 2001-09-18
  - impl swConcepts open 2001-09-18
7.
  - req textStructure done 2001-04-04
  - impl textStructure open 2001-09-18
8.
  - req acronyms open 2001-09-17
9.
  - req index open 2001-09-17
10.
  - req hyperText done 2001-04-04
  - impl hyperText done 2001-09-18
11.
  - req requirementGroups open 2001-08-09
12.
  - req template done 2001-09-17
  - impl template done 2001-09-17
13.
  - req outputPDF done 2001-08-08
  - des outputPDF done 2001-09-17
  - impl outputPDF done 2001-09-17
14.
  - req outputHTML done 2001-08-08
  - des outputHTML done 2001-09-17
  - impl outputHTML done 2001-09-17

- 15.   • req extractUserDoc open 2001-04-04
- 16.   • req extractModifications open 2001-09-17
- 17.   • req emacsSupport open 2001-04-04
- 18.   • req verification done 2001-09-19
- des verification done 2001-09-19
- inst verification done 2001-09-19

## B References

1. the projectDoc DTD file  
<http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/project.dtd>
2. template of projectDoc files  
<http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/template.xml>
3. template make file for projectDoc  
<http://www.iram.es/IRAMES/documents/ncs30mProjectDoc/OTHERS/projectTo.make>