SLOVENSKI STANDARD

SIST ES 59011:2004

julij 2004

Specification for the representation of quality rules and metrics for hardware and software design languages

ICS 35.060

Referenčna številka
SIST ES 59011:2004(en)

EUROPEAN SPECIFICATION **ES 59011**

SPÉCIFICATION EUROPÉENNE

EUROPÄISCHE SPEZIFIKATION March 2001

ICS 35.060

English version

Specification for the representation of Quality rules and metrics for Hardware and Software Design Languages

This European Specification was approved by CENELEC on 2000-10-16.

CENELEC members are required to announce the existence of this ES in the same way as for an EN and to make the ES available promptly at national level in an appropriate form. It is permissible to keep conflicting national standards in force.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Ref. No. ES 59011:2001 E

Foreword

This European Specification was prepared by the Technical Committee CENELEC TC 217, Electronic Design Automation (EDA).

The text of the draft was submitted to the National Committees members of CENELEC for comments. It was voted upon during the meeting of CLC/TC 217 and approved by CENELEC as ES 59011 on 2000-10-16.

The following date was fixed:

- latest date by which the existence of the ES has to be announced at national level (doa) 2001-05-15

Contents

1 Scope

The quality or methodology departments of all major European automotive, electronic, telecom and aerospace companies try to ensure that code developed within the company adheres to certain coding guidelines. These rules cover aspects of programming style that relate to, for example, the reusability, maintainability, portability and documentation of the code. The coding guidelines are either industry standards or rules that have been specified within the company, and typically exist in the form of written documents accessible by all programmers or designers.

The purpose of this document is to define a specification for the presentation of quality rules and metrics.

2 Definitions

The following terms that are used in this document are defined below in subclauses 2.1 to 2.4:

- classifications;
- quality characteristics (and sub-characteristics);
- rulesets;
- policy;
- level of severity;
- rules;
- metrics;
- rules and metrics representation template.

2.1 Classification

2.1.1 QA point of view

For the Quality Assurance department, an outstanding report must indicate which impact on quality have been evaluated (how much the code is portable, maintainable, usable,…), so that they can qualify the code during design reviews according to the projects they are reviewing (re-usable macros, specific designs,...). Thus

- coding rules should be classified according to **impact on quality characteristics**, e.g. portability, maintainability, usability or else.
- the level of severity of the rule should depend on the project e.g. when one rule impacting portability fails for re-usable macro it has to output a fatal error.

To achieve this, they need

- to be able to bundle rules into "**rulesets**" according to their impacts on quality,
- and to bundle "rulesets" into "**policies**" according to the type of designs (re-usable macro, specific designs,...), to the tools used (for simulation and synthesis efficiency), to the technology (Actel, Altera, Xilinx,…) and to assign each ruleset with a **level of severity** (fatal, error, warning, note) in the ruleset/policy link.

2.1.2 Designer point of view

For the designer an outstanding report must indicate which rules fail, why and eventually which chapter he has to read in the **Language Reference Manual**, to fix it.

2.1.3 Managing two classifications

To satisfy these two needs,

- the chapters of the **Language Reference Manual** are coded in the identification of the rule (**rule id**)
- and the **impact on quality** in a field named **impact on quality characteristics** of the template.

2.2 Rules definition

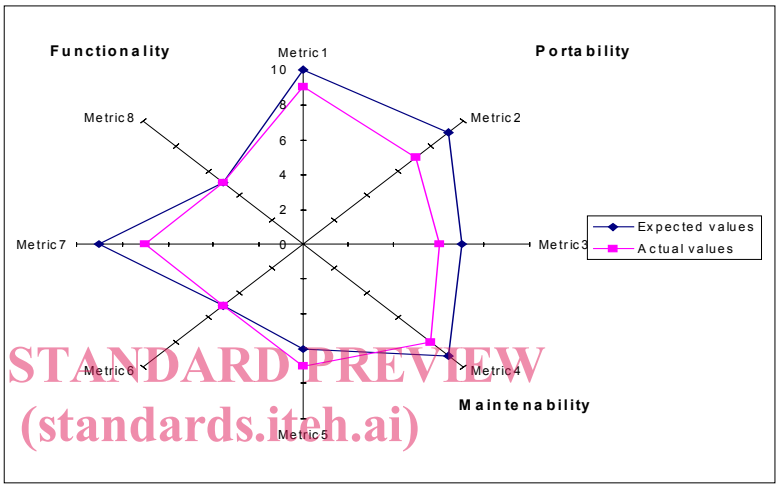
Rules are used to check the code against quality requirements. For each violated rules, explanations and recommendations are provided to improve the source code.

2.3 Metrics definition

Metrics return systematically values which are used to analyse the code against quality models.

Quality models can be predefined or project specific.

Graphs (such as Kiviat graphs) are used to visualise the level of compliance with the selected quality model.



2.4 Rules and metrics representation template definition

A unique and language independent pattern used to describe the rules according to given fields.

The OMI guidelines have been taken into account as the starting point. They have the following description fields:

- Rule ID : Unique code
- Rule Name : Short name for rule
- Rule Description : Description on what the rule seeks to achieve
- Rule Category: General categories of rules
- Justification : Why is this necessary
- Area of Impact : Where does this benefit e.g. synthesis, maintainability etc.
- Related Rules : Any related rule ID's
- Conflicting Rules : Any conflicting rules
- Reference : Where did this rule come from
- Automatic Check: Can this rule be checked automatically

They have been tested on real cases and improved as described in this specification.

3 Acronyms and references

3.1 Acronyms

EDA : Electronic Design Automation

IEEE : Institute of Electrical and Electronics Engineers

VHSIC : Very High Scale Integrated Circuit

VHDL : VHSIC Hardware Description Language

O-VHDL : Objective VHDL

OMI : Open Microprocessor systems Initiative

ANSI : American National Standard Institute

LRM : Language Reference Manual

ISO : International Standard Organisation

3.2 References

EN 61691-1:1997, *IEEE Standard VHDL Language Reference Manual* (IEEE Standard STD 1076-1993)

VHDL Coding Standard OMI Draft Standard for Open Review

ISO/IEC 9899:1999, *Programming Language C*

ISO/IEC 14882:1998, *Programming Language C++*

4 Rules and metrics representation templates

Rules are described according to the following fields of the template.

Field	Rules	Metrics
Id	M	M
Version	M	M
Language	M	M
Name	M	M
Specification	M	M
Description	O	NA
Level of description	M	M
Report	O	NA
Justification	M	M
Impact on quality characteristics	M	M
Related rules and metrics	O	O
Conflicting rules and metrics	O	O
Reference (rule programmer)	M	M
Origin of the rule, metrics (example: OMI, ESA, TCC, TCO, TTM, synthesis level 0, level 1)	M	M
Rule automatic check capability	M	NA
Metric measurability	NA	M

M = Mandatory, O = Optional, NA = Non Applicable.

These items shall be completed, in English language, as defined in the 4.1 to 4.16.

4.1 Rule or metrics Id

Rules and metrics will be identified by a block of 8 letters which makes a significant mnemonic code. This block is made up of the following five fields:

HW/SW	Rule/Metrics	Category Code	Sub-Category Code	Mnemonic
1 char	1 char	1 char	2 char	3 char

The code for the first field is **H** for hardware (VHDL or O-VHDL) or **S** for Software (C and C++) or **C** for HW/SW Co-design.

The code for the second field is **R** for a Rule and **M** for a Metric.

The code for the third field is the letter of the category code (according to the **Language Reference Manual)** in which the rule or metric is classified.

The code for the fourth field is the two letters of the sub-category code (according to the **Language Reference Manual)** in which the rule or metric is classified.

The mnemonics of the Id is meant to ease the search of the description of the rule by the designer for messages issued by the checkers.

The last field contains a 3 digits number from 000 to 999.

Example: **HRCNAXXX** will be the **H**ardware **R**ule **XXX** within the category **C**ode Layout and sub-category **Na**mes.

4.2 Version

This field gives the version number of the rule. As rules can be updated, the QA and end-user need to know which version of the rules have been checked.

4.3 Language

This field indicates the language the rule is applied to (C, C++, VHDL, O-VHDL).

4.4 Rule or metric name

The name is a representative short name for the rule or metric.

4.5 Specification

This item defines the behaviour of the rule or metric.

4.6 Description

This item describes the rule or metric by giving either an example of what should not be done or an example of good implementation or both or none.

4.7 Level of description

It describes the abstraction level of the description, e.g. algorithmic, behavioural, RTL. It is not used for software applications.

4.8 Report

This item describes the report message which is to be given to the user in case of failure.

4.9 Justification

This item explains why this rule or metric is necessary.

4.10 Impact on quality characteristics

In this field are listed the quality characteristics and sub-characteristics that are impacted by the rule. The table of quality characteristics and sub-characteristics according to ISO 9126 is given later in this document.

The related ISO characteristics and sub-characteristics will also be listed by using the code letters. Example: MA (Maintainability/Analysability).

4.11 Related rules and metrics

This item gives all the rules and metrics ID of the related rules and metrics.

4.12 Conflicting rules and metrics

This item gives the rules and metrics ID for conflicting rules and metrics.

4.13 Reference

This item points out the name of the programmer who has developed the rule.

4.14 Origin of the rule or metric

This field indicates the document or source the rule comes from.

4.15 Rule automatic check capability

This item points out if the rule can be checked automatically or not.

4.16 Metric measurability

This item points out if the metric can be computed.

4.17 Example of use of the template

Rule ID :HRGDC007
Version :1.0
Language :VHDL, O-VHDL
Rule Name :GenericExistence
Specification :The presence of GENERIC is checked.
Description :
An example of use of GENERIC within a VHDL entity:

```
Entity adder is
 Generic (WIDTH: natural);
 Port (
       a,b: in std_logic_vector(WIDTH-1 downto 0);
       cin: in std_logic;
       s: out std_logic_vector(0 to WIDTH-1);
       cout: out std_logic
 );
end adder;
```


Level of description :Behavioural and RTL code
Report :


```
You never used GENERIC
The GENERIC usage could improve the reusability of your design
```

Justification :GENERIC is one of the main tools to design parametrizable devices. Its usage is recommended to improve the reusability of the design.
Impact on quality :PA
Related rules :HRRTY001, HRGDC009, HRGDC008
Conflicting rules :None
Reference :Confidential
Origin :Confidential
Automatic Check :yes