

This is an **Archive IEEE Standard**. It has been withdrawn or superseded by a later version of the standard.

ARCHIVE STANDARDS MAY HAVE VALUE AS HISTORICAL DOCUMENTS, BUT THEY ARE NOT VALID OR APPROVED IEEE STANDARDS. THE IEEE SHALL NOT BE LIABLE FOR ANY DAMAGES RESULTING FROM THE SUBSCRIBER'S OR USERS' PRACTICE OF THIS UNAPPROVED, ARCHIVE IEEE STANDARD.

For more information about this or other current, revised or withdrawn standards, please consult the IEEE Standards Status report on-line at <http://standards.ieee.org/db/status/>.

IEEE Recommended Practice for Software Requirements Specifications

Sponsor

**Software Engineering Standards Committee
of the
IEEE Computer Society**

Approved December 2, 1993

IEEE Standards Board

Approved June 3, 1994

American National Standards

Abstract: The content and qualities of a good software requirements specification (SRS) are described and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products.

Keywords: contract, customer, prototyping, software requirements specification, supplier, system requirements specifications

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1994 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1994. Printed in the United States of America.

ISBN 1-55937-395-4

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Introduction

(This introduction is not a part of IEEE Std 830-1993, IEEE Recommended Practice for Software Requirements Specifications.)

This recommended practice describes recommended approaches for the specification of software requirements. It is based on a model in which the result of the software requirements specification process is an unambiguous and complete specification document. It should help

- a) Software customers to accurately describe what they wish to obtain
- b) Software suppliers to understand exactly what the customer wants
- c) Individuals to accomplish the following goals:
 - 1) Develop a standard software requirements specification (SRS) outline for their own organizations
 - 2) Define the format and content of their specific software requirements specifications
 - 3) Develop additional local supporting items such as an SRS quality checklist, or an SRS writer's handbook

To the customers, suppliers, and other individuals, a good SRS should provide several specific benefits, such as:

- a) *Establish the basis for agreement between the customers and the suppliers on what the software product is to do.* The complete description of the functions to be performed by the software specified in the SRS will assist the potential user to determine if the software specified meets their needs or how the software must be modified to meet their needs.
- b) *Reduce the development effort.* The preparation of the SRS forces the various concerned groups in the customer's organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting. Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early in the development cycle when these problems are easier to correct.
- c) *Provide a basis for estimating costs and schedules.* The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates.
- d) *Provide a baseline for validation and verification.* Organizations can develop their validation and verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured.
- e) *Facilitate transfer.* The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organization, and suppliers find it easier to transfer it to new customers.
- f) *Serve as a basis for enhancement.* Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation.

At the time this recommended practice was completed, the Software Requirements Specifications Working Group had the following membership:

Edward R. Byrne, Chair

Fletcher J. Buckley
A. M. Davis
J. W. Horch
Motti Klein

Thomas M. Kurihara
John R. Matras
Robert Nation
William C. Sasso
Carl S. Seddio

Carl A. Singer
Richard H. Thayer
Leonard L. Tripp
A. Yonda

The following persons were on the balloting committee:

D. Avery
H. R. Berlack
W. J. Boll, Jr.
F. Buckley
G. Cozens
S. Crawford
P. I. Davis
B. K. Derganc
C. L. Evans
J. W. Fendrich
K. Fortenberry
D. Gelperin
Y. Gershkovich
J. Gonzalez
D. A. Gustafson

J. Harauz
W. Hefley
J. W. Horch
P. L. Hung
M. S. Karasik
R. Kosinski
T. Kurihara
R. Lane
B. Lau
B. Livson
J. Maayan
J. Marijarvi
R. Martin
S. D. Matthews
I. Mazza

S. McGrath
D. E. Nickle
G. D. Schumacher
R. W. Shillato
D. M. Siefert
H. M. Sneed
V. Srivastava
R. H. Thayer
G. D. Tice
L. L. Tripp
D. Wallace
W. M. Walsh
P. R. Work
N. C. Yopconka
J. Zalewski

When the IEEE Standards Board approved this standard on December 2, 1993, it had the following membership:

Wallace S. Read, *Chair*

Donald C. Loughry, *Vice Chair*

Andrew G. Salem, *Secretary*

Gilles A. Baril
José A. Berrios de la Paz
Clyde R. Camp
Donald C. Fleckenstein
Jay Forster*
David F. Franklin
Ramiro Garcia
Donald N. Heirman

Jim Isaak
Ben C. Johnson
Walter J. Karplus
Lorraine C. Kevra
E. G. "Al" Kiener
Ivor N. Knight
Joseph L. Koepfinger*
D. N. "Jim" Logothetis

Don T. Michael*
Marco W. Migliaro
L. John Rankine
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp
Donald W. Zipse

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
David E. Soffrin
Stanley I. Warshaw

Rachel A. Meisel
IEEE Standards Project Editor

Contents

CLAUSE	PAGE
1. Overview.....	1
1.1 Scope.....	1
2. References.....	2
3. Definitions.....	3
4. Considerations for producing a good SRS.....	4
4.1 Nature of the SRS	4
4.2 Environment of the SRS	4
4.3 Characteristics of a good SRS.....	5
4.4 Joint preparation of the SRS	9
4.5 SRS evolution	9
4.6 Prototyping.....	9
4.7 Embedding design in the SRS.....	10
4.8 Embedding project requirements in the SRS	10
5. The parts of an SRS	11
5.1 Introduction (Section 1 of the SRS).....	11
5.2 Overall description (Section 2 of the SRS).....	12
5.3 Specific requirements (Section 3 of the SRS).....	15
5.4 Supporting information.....	20
Annex A	21
A.1 Template of SRS section 3 organized by mode: Version 1	21
A.2 Template of SRS section 3 organized by mode: Version 2	21
A.3 Template of SRS section 3 organized by user class	22
A.4 Template of SRS section 3 organized by object	22
A.5 Template of SRS section 3 organized by feature.....	23
A.6 Template of SRS section 3 organized by stimulus	24
A.7 Template of SRS section 3 organized by functional hierarchy.....	24
A.8 Template of SRS section 3 showing multiple organizations	26

IEEE Recommended Practice for Software Requirements Specifications

1. Overview

This recommended practice describes recommended approaches for the specification of software requirements. It is divided into five clauses. Clause 1 explains the scope of this recommended practice. Clause 2 lists the references made to other standards. Clause 3 provides definitions of specific terms used. Clause 4 provides background information for writing a good SRS. Clause 5 discusses each of the essential parts of an SRS. This recommended practice also has an annex, which provides alternate format templates.

1.1 Scope

This is a recommended practice for writing software requirements specifications. It describes the content and qualities of a good software requirements specification (SRS) and presents several sample SRS outlines.

This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products. However, application to already-developed software could be counterproductive.

When software is embedded in some larger system, such as medical equipment, then issues beyond those identified in this standard may have to be addressed.

This recommended practice describes the process of creating a product and the content of the product. The product is a software requirements specification. This recommended practice can be used to create such software requirements specification directly or can be used as a model for a more specific standard.

This recommended practice does not identify any specific method, nomenclature, or tool for preparing an SRS.

2. References

- ASTM 1340-90, Standard Guide for Rapid Prototyping of Computerized Systems.¹
- IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI).²
- IEEE Std 730-1989, IEEE Standard for Software Quality Assurance Plans (ANSI).
- IEEE Std 828-1990, IEEE Standard for Software Configuration Management Plans (ANSI).
- IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI).
- IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI).
- IEEE Std 983-1986, IEEE Guide to Software Quality Assurance Planning.³
- IEEE Std 1002-1987, IEEE Standard Taxonomy for Software Engineering Standards (ANSI).
- IEEE Std 1012-1986, IEEE Standard for Software Verification and Validation Plans (ANSI).
- IEEE Std 1016-1987, IEEE Recommended Practice for Software Design Descriptions (ANSI).
- IEEE Std 1028-1988, IEEE Standard for Software Reviews and Audits (ANSI).
- IEEE Std 1042-1987, IEEE Guide to Software Configuration Management (ANSI).
- IEEE Std 1058.1-1987, IEEE Standard for Project Management Plans (ANSI).
- IEEE Std 1074-1991, IEEE Standard for Developing Software Life Cycle Processes (ANSI).
- IEEE P1233, October 1993, Draft Guide to Developing System Requirements Specifications.⁴

¹ASTM publications are available from the Customer Service Department, American Society for Testing and Materials, 1916 Race Street, Philadelphia, PA 19103, USA.

²IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

³This standard has been withdrawn; however, copies can be obtained from the IEEE Standards Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

⁴This authorized standards project was not approved by the IEEE Standards Board at the time this went to press. It is available from the IEEE Standards Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

3. Definitions

In general the definitions of terms used in this recommended practice conform to the definitions provided in IEEE Std 610.12-1990.⁵ The definitions below are key terms as they are used in this recommended practice.

3.1 contract: A legally binding document agreed upon by the customer and supplier. This includes the technical and organizational requirements, cost, and schedule for a product. A contract may also contain informal but useful information such as the commitments or expectations of the parties involved.

3.2 customer: The person, or persons, who pay for the product and usually (but not necessarily) decide the requirements. In the context of this recommended practice the customer and the supplier may be members of the same organization.

3.3 supplier: The person, or persons, who produce a product for a customer. In the context of this document, the customer and the supplier may be members of the same organization.

3.4 user: The person, or persons, who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).

⁵Information on references can be found in clause 2.

4. Considerations for producing a good SRS

This clause provides background information that should be considered in writing an SRS. This includes the following:

- a) Nature of the SRS
- b) Environment of the SRS
- c) Characteristics of a good SRS
- d) Joint preparation of the SRS
- e) SRS evolution
- f) Prototyping
- g) Embedding design in the SRS
- h) Embedding project requirements in the SRS

4.1 Nature of the SRS

The SRS is a specification for a particular software product, program, or set of programs that performs certain functions in a specific environment. The SRS may be written by one or more representatives of the supplier, one or more representatives of the customer, or by both. Subclause 4.4 recommends both.

The basic issues that the SRS writer(s) shall address are the following:

- a) *Functionality.* What is the software supposed to do?
- b) *External interfaces.* How does the software interact with people, the system's hardware, other hardware, and other software?
- c) *Performance.* What is the speed, availability, response time, recovery time of various software functions, etc.?
- d) *Attributes.* What are the portability, correctness, maintainability, security, etc. considerations?
- e) *Design constraints imposed on an implementation.* Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

The SRS writer(s) should avoid placing either design or project requirements in the SRS.

For recommended contents of an SRS see clause 5.

4.2 Environment of the SRS

It is important to consider the part that the SRS plays in the total project plan, which is defined in IEEE Std 610.12-1990. The software may contain essentially all the functionality of the project or it may be part of a larger system. In the latter case typically there will be an SRS that will state the interfaces between the system and its software portion, and will place external performance and functionality requirements upon the software portion. Of course the SRS should then agree with and expand upon these system requirements.

IEEE Std 1074-1991 describes the steps in the software life cycle and the applicable inputs for each step. Other standards, such as those listed in clause 2, relate to other parts of the software life cycle and so may complement software requirements.

Since the SRS has a specific role to play in the software development process, SRS writer(s) should be careful not to go beyond the bounds of that role. This means the SRS

- a) Should correctly define all of the software requirements. A software requirement may exist because of the nature of the task to be solved or because of a special characteristic of the project.