SRS TEMPLATE WITH EXPLANATION

5.1 Introduction (Section 1 of the SRS)

The introduction of the SRS should provide an overview of the entire SRS. It should contain the following

SUBSECTIONS:

a) Purpose;

b) Scope;

c) Definitions, acronyms, and abbreviations;

d) References;

e) Overview.

5.1.1 Purpose (1.1 of the SRS)

This subsection should

a) Delineate the purpose of the SRS;

b) Specify the intended audience for the SRS.

5.1.2 Scope (1.2 of the SRS)

This subsection should

a) Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);

b) Explain what the software product(s) will, and, if necessary, will not do;

c) Describe the application of the software being specified, including relevant benefits, objectives, and goals;

d) Be consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.

TABLE OF CONTENTS

2.4 Constraints

2.5 Assumptions and dependencies

3. Specific requirements (See 5.3.1 through 5.3.8 for explanations of possible specific requirements. See also Annex A for several different ways of organizing this section of the SRS.)

5.1.3 Definitions, acronyms, and abbreviations (1.3 of the SRS)

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

5.1.4 References (1.4 of the SRS)

This subsection should

a) Provide a complete list of all documents referenced elsewhere in the SRS;

b) Identify each document by title, report number (if applicable), date, and publishing organization;

c) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

5.1.5 Overview (1.5 of the SRS)

This subsection should

a) Describe what the rest of the SRS contains;

b) Explain how the SRS is organized.

5.2 Overall description (Section 2 of the SRS)

This section of the SRS should describe the general factors that affect the product and its requirements. This section does not state specific Requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

This section usually consists of six subsections, as follows:

a) **Product perspective;**

b) Product functions;

c) User characteristics;

d) Constraints;

e) Assumptions and dependencies;

f) Apportioning of requirements.

5.2.1 Product perspective (2.1 of the SRS)

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS deÞnes a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful.

This subsection should also describe how the software operates inside various constraints. For example,

These constraints could include

a) System interfaces;

b) User interfaces;

c) Hardware interfaces;

d) Software interfaces;

e) Communications interfaces;

f) Memory;

g) Operations;

h) Site adaptation requirements.

5.2.1.1 System interfaces

This should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

5.2.1.2 User interfaces

This should specify the following:

a) *The logical characteristics of each interface between the software product and its users.*

This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements.

b) *All the aspects of optimizing the interface with the person who must use the system.*

This may simply comprise a list of dos and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, e.g., A clerk typist grade 4 can do function *X* in *Z* min after 1 h of training rather than a typist can do function *X*.Ó (This may also be specified in the Software System Attributes under a section titled Ease of Use.)

5.2.1.3 Hardware interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how

they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.

5.2.1.4 Software interfaces

This should specify the use of other required software products (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

Ñ Name;

Ñ Mnemonic;

Ñ Specification number;

Ñ Version number;

Ñ Source.

For each interface, the following should be provided:

Ñ Discussion of the purpose of the interfacing software as related to this software product.

Ñ Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

5.2.1.5 Communications interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

5.2.1.6 Memory constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

5.2.1.7 Operations

This should specify the normal and special operations required by the user such as

a) The various modes of operations in the user organization (e.g., user-initiated operations);

b) Periods of interactive operations and periods of unattended operations;

c) Data processing support functions;

d) Backup and recovery operations.

NOTE this is sometimes specified as part of the User Interfaces section.

5.2.1.8 Site adaptation requirements

This should

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode (e.g., grid values, safety limits, etc.);

b) Specify the site or mission-related features that should be modified to adapt the software to a particular installation.

5.2.2 Product functions (2.2 of the SRS)

This subsection of the SRS should provide a summary of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.

Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity

a) The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.

b) Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables.

5.2.3 User characteristics (2.3 of the SRS)

This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements, but rather should provide the reasons why certain specific requirements are later specified in Section 3 of the SRS.

5.2.4 Constraints (2.4 of the SRS)

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies;

b) Hardware limitations (e.g., signal timing requirements);

c) Interfaces to other applications;

d) Parallel operation;

e) Audit functions;

f) Control functions;

g) Higher-order language requirements;

h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);

i) Reliability requirements;

j) Criticality of the application;

k) Safety and security considerations.

5.2.5 Assumptions and dependencies (2.5 of the SRS)

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS.

These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific

operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

5.2.6 Apportioning of requirements (2.6 of the SRS)

This subsection of the SRS should identify requirements that may be delayed until future versions of the system.

5.3 Specific requirements (Section 3 of the SRS)

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should

include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be stated in conformance with all the characteristics described in 4.3.

b) Specific requirements should be cross-referenced to earlier documents that relate.

c) All requirements should be uniquely identifiable.

d) Careful attention should be given to organizing the requirements to maximize readability.

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in 5.3.1 through 5.3.7.

5.3.1 External interfaces

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in 5.2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;

b) Description of purpose;

c) Source of input or destination of output;

d) Valid range, accuracy, and/or tolerance;

e) Units of measure;

f) Timing;

g) Relationships to other inputs/outputs;

h) Screen formats/organization;

i) Window formats/organization;

j) Data formats;

k) Command formats;

l) End messages.

5.3.2 Functions

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as ÒshallÓ statements starting with the system shallÉÓ

These include

a) Validity checks on the inputs

b) Exact sequence of operations

c) Responses to abnormal situations, including

1) Overflow

2) Communication facilities

3) Error handling and recovery

d) Effect of parameters

e) Relationship of outputs to inputs, including

1) Input/output sequences

2) Formulas for input to output conversion

It may be appropriate to partition the functional requirements into sub functions or sub processes. This does not imply that the software design will also be partitioned that way.

5.3.3 Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

a) The number of terminals to be supported;

b) The number of simultaneous users to be supported;

c) Amount and type of information to be handled.

Static numerical requirements are sometimes identified under a separate section entitled Capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 s.

rather than,

An operator shall not have to wait for the transaction to complete.

NOTE Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

5.3.4 Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

a) Types of information used by various functions;

b) Frequency of use;

c) Accessing capabilities;

d) Data entities and their relationships;

e) Integrity constraints;

f) Data retention requirements.

5.3.5 Design constraints

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

5.3.5.1 Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

a) Report format;

b) Data naming;

c) Accounting procedures;

d) Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

5.3.6 Software system attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. Sub clauses 5.3.6.1 through 5.3.6.5 provide a partial list of examples.

5.3.6.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

5.3.6.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

5.3.6.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to a) Utilize certain cryptographically techniques;

b) Keep specific log or history data sets;

c) Assign certain functions to different modules;

d) Restrict communications between some areas of the program;

e) Check data integrity for critical variables.

5.3.6.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

5.3.6.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following: a) Percentage of components with host-dependent code;

b) Percentage of code that is host dependent;

c) Use of a proven portable language;

d) Use of a particular compiler or language subset;

e) Use of a particular operating system.

5.3.7 Organizing the specific requirements

For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in Section 3 of the SRS. Some of these organizations are described in 5.3.7.1 through 5.3.7.7.

5.3.7.1 System mode

Some systems behave quite differently depending on the mode of operation. For example, a control system may have different sets of functions depending on its mode: training, normal, or emergency. When organizing this section by mode, the outline in A.1 or A.2 should be used. The choice depends on whether interfaces and performance are dependent on mode.

5.3.7.2 User class

Some systems provide different sets of functions to different classes of users. For example, an elevator control system presents different capabilities to passengers, maintenance workers, and fire fighters. When organizing this section by user class, the outline in A.3 should be used.

5.3.7.3 Objects

Objects are real-world entities that have a counterpart within the system. For example, in a patient monitoring system, objects include patients, sensors, nurses, rooms, physicians, medicines, etc. Associated with each object is a set of attributes (of that object) and functions (performed by that object). These functions are also called services, methods, or processes. When organizing this section by object, the outline in A.4 should be used. Note that sets of objects may share attributes and services. These are grouped together as classes.

5.3.7.4 Feature

A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. For example, in a telephone system, features include local call, call forwarding, and conference call. Each feature is generally described in a sequence of stimulus-response pairs. When organizing this section by feature, the outline in A.5 should be used.

5.3.7.5 Stimulus

Some systems can be best organized by describing their functions in terms of stimuli. For example, the functions of an automatic aircraft landing system may be organized into sections for loss of power, wind shear, sudden change in roll, vertical velocity excessive, etc. When organizing this section by stimulus, the outline in A.6 should be used.

5.3.7.6 Response

Some systems can be best organized by describing all the functions in support of the generation of a response. For example, the functions of a personnel system may be organized into sections corresponding to all functions associated with generating paychecks, all functions associated with generating a current list of employees, etc. The outline in A.6 (with all occurrences of stimulus replaced with response) should be used.

5.3.7.7 Functional hierarchy

When none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the

relationships between and among the functions and data. When organizing this section by functional hierarchy, the outline in A.7 should be used.

5.3.8 Additional comments

Whenever a new SRS is contemplated, more than one of the organizational techniques given in 5.3.7.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification. For example, see A.8 for an organization-combining user class and feature. Any additional requirements may be put in a separate section at the end of the SRS.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization.

For example, when organizing by mode, Unite state machines or state charts may prove helpful; when organizing by object, object-oriented

analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful;

and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.

In any of the outlines given in A.1 through A.8, those sections called Functional Requirement may be described in native language (e.g., English), in pseudo code, in a system dentition language, or in four subsections titled: Introduction, Inputs, Processing, and Outputs.

5.4 Supporting information

The supporting information makes the SRS easier to use. It includes the following:

a) Table of contents;

b) Index;

c) Appendixes.

5.4.1 Table of contents and index

The table of contents and index are quite important and should follow general compositional practices.

5.4.2 Appendixes

The appendixes are not always considered part of the actual SRS and are not always necessary. They may include

a) Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;

b) Supporting or background information that can help the readers of the SRS;

c) A description of the problems to be solved by the software;

d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

When appendixes are included, the SRS should explicitly state whether or not the appendixes are to be considered part of the requirements.

A.1 Template of SRS Section 3 organized by mode: Version 1

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

3.2.1. *n* Functional requirement 1.*n*

3.2.2 Mode 2

3.2. *m* Mode *m*

3.2. *m.* 1 Functional requirement *m.*1

3.2. *m.n* Functional requirement *m.n*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

A.2 Template of SRS Section 3 organized by mode: Version 2

3. Specific requirements

3.1. Functional requirements

3.1.1 Mode 1

3.1.1.1 External interfaces

3.1.1.1.1 User interfaces

3.1.1.1.2 Hardware interfaces

3.1.1.1.3 Software interfaces

3.1.1.1.4 Communications interfaces

3.1.1.2 Functional requirements

3.1.1.2.1 Functional requirement 1

IEEE

Std 830-1998 IEEE RECOMMENDED PRACTICE FOR

3.1.1.2.*n* Functional requirement *n*

3.1.1.3 Performance

3.1.2 Mode 2

3.1.*m* Mode *m*

3.2 Design constraints

3.3 Software system attributes

3.4 Other requirements

A.3 Template of SRS Section 3 organized by user class

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

3.2.1.*n* Functional requirement 1.*n*

3.2.2 User class 2

3.2.*m* User class *m*

3.2.*m*.1 Functional requirement *m*.1

3.2.*m*.*n* Functional requirement *m*.*n*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

A.4 Template of SRS Section 3 organized by object

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Classes/Objects

3.2.1 Class/Object 1

IEEE

SOFTWARE REQUIREMENTS SPECIFICATIONS Std 830-1998

3.2.1.1 Attributes (direct or inherited)

3.2.1.1.1 Attribute 1

3.2.1.1.*n* Attribute *n*

3.2.1.2 Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

3.2.1.2.*m* Functional requirement 1.*m*

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

3.2.*p* Class/Object *p*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

A.5 Template of SRS Section 3 organized by feature

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

3.2.1.3.*n* Functional requirement *n*

3.2.2 System feature 2

3.2.*m* System feature *m*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

IEEE

Std 830-1998 IEEE RECOMMENDED PRACTICE FOR

A.6 Template of SRS Section 3 organized by stimulus

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Stimulus 2

3.2.*m* Stimulus m

3.2.*m*.1 Functional requirement *m*.1

3.2.*m.n* Functional requirement *m.n*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

A.7 Template of SRS Section 3 organized by functional hierarchy

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

3.2.1.1.1 Data entities

3.2.1.1.2 Pertinent processes

3.2.1.1.3 Topology

3.2.1.2 Data flow diagram 2

3.2.1.2.1 Data entities

3.2.1.2.2 Pertinent processes

3.2.1.2.3 Topology

3.2.1.*n* Data flow diagram *n*

IEEE

SOFTWARE REQUIREMENTS SPECIFICATIONS Std 830-1998

3.2.1.*n*.1 Data entities

3.2.1.*n*.2 Pertinent processes

3.2.1.*n*.3 Topology

3.2.2 Process descriptions

3.2.2.1 Process 1

3.2.2.1.1 Input data entities

3.2.2.1.2 Algorithm or formula of process

3.2.2.1.3 Affected data entities

3.2.2.2 Process 2

3.2.2.2.1 Input data entities

3.2.2.2.2 Algorithm or formula of process

3.2.2.2.3 Affected data entities

3.2.2.*m* Process *m*

3.2.2.*m*.1 Input data entities

3.2.2.*m*.2 Algorithm or formula of process

3.2.2.*m*.3 Affected data entities

3.2.3 Data construct specifications

3.2.3.1 Construct 1

3.2.3.1.1 Record type

3.2.3.1.2 Constituent fields

3.2.3.2 Construct 2

3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields

3.2.3.*p* Construct *p*

3.2.3.*p*.1 Record type

3.2.3.*p*.2 Constituent fields

3.2.4 Data dictionary

3.2.4.1 Data element 1

3.2.4.1.1 Name

3.2.4.1.2 Representation

3.2.4.1.3 Units/Format

3.2.4.1.4 Precision/Accuracy

3.2.4.1.5 Range

3.2.4.2 Data element 2

3.2.4.2.1 Name

3.2.4.2.2 Representation

3.2.4.2.3 Units/Format

3.2.4.2.4 Precision/Accuracy

3.2.4.2.5 Range

3.2.4.*q* Data element *q*

3.2.4.*q*.1 Name

3.2.4.*q*.2 Representation

3.2.4.*q*.3 Units/Format

3.2.4.*q*.4 Precision/Accuracy

3.2.4.*q*.5 Range

IEEE

Std 830-1998 IEEE RECOMMENDED PRACTICE FOR

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

A.8 Template of SRS Section 3 showing multiple organizations

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

3.2.1.*m* Feature 1.*m*

3.2.1.*m*.1 Introduction/Purpose of feature

3.2.1.*m*.2 Stimulus/Response sequence

3.2.1.*m*.3 Associated functional requirements

3.2.2 User class 2

3.2.*n* User class *n*

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.6 Other requirements

IEEE

SOFTWARE REQUIREMENTS SPECIFICATIONS Std 830-1998

Annex B

(Informative)

Guidelines for compliance with IEEE/EIA 12207.1-1997

B.1 Overview

The Software Engineering Standards Committee (SESC) of the IEEE Computer Society has endorsed the policy of adopting international standards. In 1995, the international standard, ISO/IEC 12207, Information technology Software life cycle processes, was completed. The standard establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry.

In 1995 the SESC evaluated ISO/IEC 12207 and decided that the standard should be adopted and serve as the basis for life cycle processes within the IEEE Software Engineering Collection. The IEEE adaptation of ISO/IEC 12207 is IEEE/EIA 12207.0-1996. It contains ISO/IEC 12207 and the following additions:

improved compliance approach, life cycle process objectives, life cycle data objectives, and errata.

The implementation of ISO/IEC 12207 within the IEEE also includes the following:

Ñ IEEE/EIA 12207.1-1997, IEEE/EIA Guide for Information Technology and Software life cycle processes and Life cycle data;

Ñ IEEE/EIA 12207.2-1997, IEEE/EIA Guide for Information Technology and Software life cycle processes and Implementation considerations; and

Ñ Additions to 11 SESC standards (i.e., IEEE Stds 730, 828, 829, 830, 1012, 1016, 1058, 1062, 1219,

1233, 1362) to define the correlation between the data produced by existing SESC standards and the data produced by the application of IEEE/EIA 12207.1-1997.

NOTEÑ Although IEEE/EIA 12207.1-1997 is a guide, it also contains provisions for application as a standard with specific compliance requirements. This annex treats 12207.1-1997 as a standard.

B.1.1 Scope and purpose

Both IEEE Std 830-1998 and IEEE/EIA 12207.1-1997 place requirements on a Software Requirements

Description Document. The purpose of this annex is to explain the relationship between the two sets of requirements so that users producing documents intended to comply with both standards may do so.

B.2 Correlation

This clause explains the relationship between IEEE Std 830-1998 and IEEE/EIA 12207.0-1996 and IEEE/

EIA 12207.1-1997 in the following areas: terminology, process, and life cycle data.

B.2.1 Terminology correlation

Both this recommended practice and IEEE/EIA 12207.0-1996 have similar semantics for the key terms of software, requirements, specification, supplier, developer, and maintainer. This recommended practice uses IEEE

Std 830-1998 IEEE RECOMMENDED PRACTICE FOR

the term customer where IEEE/EIA 12207.0-1996 uses acquirer and this recommended practice uses user where IEEE/EIA 12207.0-1996 uses operator.

B.2.2 Process correlation

IEEE/EIA 12207.0-1996 uses a process-oriented approach for describing the definition of a set of requirements for software. This recommended practice uses a product-oriented approach, where the product is a Software Requirements Description (SRD). There are natural process steps, namely the steps to create each portion of the SRD. These may be correlated with the process requirements of IEEE/EIA 12207.0-1996. The difference is that this recommended practice is focused on the development of software requirements whereas IEEE/EIA 12207.0-1996 provides an overall life cycle view and mentions Software Requirements

Analysis as part of its Development Process. This recommended practice provides a greater level of detail on what is involved in the preparation of an SRD.

B.2.3 Life cycle data correlation

IEEE/EIA 12207.0-1996 takes the viewpoint that the software requirements are derived from the system requirements. Therefore, it uses the term, Description rather that specification to describe the software requirements. In a system in which software is a component, each requiring its own specification, there would be a System Requirements Specification (SRS) and one or more SRDs. If the term Software Requirements Specification had been used, there would be a confusion between an SRS referring to the system or software requirements. In the case where there is a stand-alone software system, IEEE/EIA 12207.1-1997

States If the software is a stand-alone system, then this document should be a specification.

B.3 Content mapping

This clause provides details bearing on a claim that an SRS complying with this recommended practice

would also achieve document compliance with the SRD described in IEEE/EIA 12207.1-1997. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA 12207.1-1997. That row is reproduced in Table B.1 of this recommended practice.

The requirements for document compliance are discussed in the following sub clauses:

Ñ B.3.1 discusses compliance with the information requirements noted in column 2 of Table B.1 as prescribed by 5.1.1.4, 5.3.4.1, and 5.3.4.2 of IEEE/EIA 12207.0-1996.

Table B.1ÑSummary of requirements for an SRD excerpted from Table 1 of IEEE/EIA 12207.1-1997

Information item IEEE/EIA 12207.0-1996

Clause

Kind

IEEE/EIA

12207.1-1997

Clause

References

Software

Requirements

Description

5.1.1.4, 5.3.4.1,

5.3.4.2

Description

(See note for 6.22.1of IEEE/EIA 12207.1-1997.)

6.22 IEEE Std 830-1998; EIA/IEEE

J-STD-016, F.2.3, F.2.4; MILSTD

961D. also see ISO/IEC

5806, 5807, 6593, 8631, 8790,

and 11411 for guidance on use of notations.

IEEE

SOFTWARE REQUIREMENTS SPECIFICATIONS Std 830-1998

Ñ B.3.2 discusses compliance with the generic content guideline (the kind of document) noted in column 3 of Table B.1 as a description. The generic content guidelines for a description appear in 5.1 of IEEE/EIA 12207.1-1997.

Ñ B.3.3 discusses compliance with the specific requirements for a Software Requirements Description noted in column 4 of Table B.1 as prescribed by 6.22 of IEEE/EIA 12207.1-1997.

Ñ B.3.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996 as described in 4.2 of IEEE/EIA 12207.1-1997.

B.3.1 Compliance with information requirements of IEEE/EIA 12207.0-1996

The information requirements for an SRD are those prescribed by 5.1.1.4, 5.3.4.1, and 5.3.4.2 of IEEE/EIA

12207.0-1996. the requirements are substantively identical to those considered in B.3.3 of this recommended practice.

B.3.2 Compliance with generic content guidelines of IEEE/EIA 12207.1-1997

According to IEEE/EIA 12207.1-1997, the generic content guideline for an SRD is generally a description, as prescribed by 5.1 of IEEE/EIA 12207.1-1997. A complying description shall achieve the purpose stated in 5.1.1 and include the information listed in 5.1.2 of IEEE/EIA 12207.1-1997.

The purpose of a description is:

IEEE/EIA 12207.1-1997, sub clause 5.1.1: Purpose: Describe a planned or actual function,

design, performance, or process.

An SRD complying with this recommended practice would achieve the stated purpose.

Any description or specification complying with IEEE/EIA 12207.1-1997 shall satisfy the generic content requirements provided in 5.1.2 of that standard. Table B.2 of this recommended practice lists the generic content items and, where appropriate, references the clause of this recommended practice that requires the same information.

Table B.2ÑCoverage of generic description requirements by IEEE Std 830-1998

IEEE/EIA 12207.1-1997 generic content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998

a) Date of issue and status Ñ Date of issue and status shall be provided.

b) Scope 5.1.1 Scope Ñ

c) Issuing organization Ñ Issuing organization shall be identified.

d) References 5.1.4 References Ñ

e) Context 5.1.2 Scope Ñ

f) Notation for description 4.3 Characteristics of a good SRS Ñ

g) Body 5. The parts of an SRS Ñ

h) Summary 5.1.1. Overview Ñ

i) Glossary 5.1.3 Definitions Ñ

j) Change history Ñ Change history for the SRD shall be provided or referenced.

IEEE

Std 830-1998 IEEE RECOMMENDED PRACTICE FOR

B.3.3 Compliance with specific content requirements of IEEE/EIA 12207.1-1997

The specific content requirements for an SRD in IEEE/EIA 12207.1-1997 are prescribed by 6.22 of IEEE/

EIA 12207.1-1997. A compliant SRD shall achieve the purpose stated in 6.22.1 of IEEE/EIA 12207.1-1997.

The purpose of the SRD is:

IEEE/EIA 12207.1-1997, sub clause 6.22.1: Purpose: Specify the requirements for a software

item and the methods to be used to ensure that each requirement has been met. Used as the basis for design and qualification testing of a software item.

An SRS complying with this recommended practice and meeting the additional requirements of Table B.3 of this recommended practice would achieve the stated purpose.

An SRD compliant with IEEE/EIA 12207.1-1997 shall satisfy the specific content requirements provided in 6.22.3 and 6.22.4 of that standard. Table B.3 of this recommended practice lists the specific content items and, where appropriate, references the clause of this recommended practice that requires the same information.

An SRD specified according the requirements stated or referenced in Table B.3 of this recommended practice shall be evaluated considering the criteria provided in 5.3.4.2 of IEEE/EIA 12207.0-1996.

Table B.3ÑCoverage of specific SRD requirements by IEEE Std 830-1998

IEEE/EIA 12207.1-1997 specific content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998
a) Generic description information See Table B.2 Ñ		
b) System identification and overview		

5.1.1 Scope Ñ

c) Functionality of the software item including:

Ð Performance requirements

Ð Physical characteristics

Ð Environmental conditions

5.3.2 Functions

5.3.3 Performance requirements

Physical characteristics and environmental conditions should be provided.

d) Requirements for interfaces external to software item

5.3.1 External interfaces Ñ

e) Qualification requirements Ñ The requirements to be used for qualification testing should be provided (or referenced).

f) Safety specifications 5.2.4 Constraints Ñ

g) Security and privacy specifications

5.3.6.3 Security Ñ

h) Human-factors engineering requirements

5.2.3 User characteristics

5.2.1.2 User interfaces

i) Data definition and database requirements

5.3.4 Logical data base requirements Ñ

j) Installation and acceptance requirements at operation site

5.2.1.8 Site adaptation requirements Installation and acceptance requirements at operation site

k) Installation and acceptance requirements at maintenance site

Ñ Installation and acceptance requirements at maintenance site

l) User documentation requirements Ñ User documentation requirements

m) User operation and execution requirements

5.2.1.7 Operations User execution requirements

IEEE

SOFTWARE REQUIREMENTS SPECIFICATIONS Std 830-1998

B.3.4 Compliance with life cycle data objectives

In addition to the content requirements, life cycle data shall be managed in accordance with the objectives provided in Annex H of IEEE/EIA 12207.0-1996.

B.4 Conclusion

The analysis suggests that any SRS complying with this recommended practice and the additions shown in

Table B.2 and Table B.3 also complies with the requirements of an SRD in IEEE/EIA 12207.1-1997. In addition, to comply with IEEE/EIA 12207.1-1997, an SRS shall support the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996.

n) User maintenance requirements 5.3.6.4 Maintainability Ñ

o) Software quality characteristics 5.3.6 Software system attributes Ñ

p) Design and implementation constraints

5.2.4 Constraints Ñ

q) Computer resource requirements 5.3.3 Performance requirements Computer resource requirements

r) Packaging requirements Ñ Packaging requirements

s) Precedence and criticality of requirements

5.2.6 Apportioning of requirements Ñ

t) Requirements traceability 4.3.8 Traceable Ñ

u) Rationale 5.2.5 Assumptions and dependencies

Ñ

Items a) through f) below are from

6.22.4

a) Support the life cycle data objectives of Annex H of IEEE/EIA

12207.0-1996

Ñ

Support the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996

b) Describe any function using well defined notation

4.3 Characteristics of a good SRS Ñ

c) Define no requirements that are in conflict

4.3 Characteristics of a good SRS Ñ

d) User standard terminology and definitions

5.1.3 Definition Ñ

e) Define each unique requirement one to prevent inconsistency

4.3 Characteristics of a good SRS Ñ

f) Uniquely identify each requirement

4.3 Characteristics of a good SRS Ñ

Table B.3ÑCoverage of specific SRD requirements by IEEE Std 830-1998 *(continued)*

IEEE/EIA 12207.1-1997 specific content	Corresponding clauses of IEEE Std 830-1998	Additions to requirements of IEEE Std 830-1998

.