INTERNATIONAL STANDARD

ISO/IEC/ IEEE 29148

Second edition
2018-11

Systems and software engineering — Life cycle processes — Requirements engineering

Ingénierie des systèmes et du logiciel — Processus du cycle de vie — Ingénierie des exigences

Reference number
ISO/IEC/IEEE 29148:2018(E)

8.4 System requirements specification

8.4.1 General

The System Requirements Specification (SyRS) identifies the technical requirements for the selected system-of-interest and usability for the envisaged human-system interaction. It defines the high-level system requirements from the domain perspective, along with background information about the overall objectives for the system, its target environment and a statement of the constraints, assumptions and non-functional requirements. It may include conceptual models designed to illustrate the system context, usage scenarios, the principal domain entities, data, information and workflows.

The purpose of the SyRS is to provide a description of what the system should do, in terms of the system's interactions or interfaces with its external environment. The SyRS should completely describe all inputs, outputs and required relationships between inputs and outputs. An SyRS has traditionally been viewed as a document that communicates the requirements of the acquirer to the technical community who will specify and build the system. The collection of requirements that constitutes the specification and its representation acts as the bridge between the two groups and needs to be understandable by both the acquirer and the technical community. One of the most difficult tasks in the creation of a system is that of communicating to all of the subgroups within both groups, especially in one document. This type of communication generally requires different formalisms and languages.

This document suggests a distinction between this structured collection of information and the way in which it is presented to its various audiences. The presentation of the SyRS should take a form that is appropriate for its intended use. This can be a paper document, models, prototypes, other non-paper document representations or any combination. All of these representations can be derived from this one SyRS to meet the needs of a specific audience. However, care should be taken to be certain that each of these presentations is traceable to a common source of system requirements information. The audience should be made aware that this structured collection of information remains the one definitive source for resolving ambiguities in the particular presentation chosen.

Generally, process requirements (how to develop or construct the system) should be contained in contract documentation such as a Statement of Work, not in a requirements specification. If included in a specification, they should be clearly identified as process requirements.

The SyRS presents the results of the definition of need, the system operational concept, the system architecture and the system requirements analysis tasks. As such, it is a description of what the system's acquirers expect it to do for them, the system's expected environment, the system's usage profile, performance parameters, expected quality and effectiveness and verification activities.

8.4.2 SyRS example outline

The specific requirements section of an SyRS should be organized such that a consensus of the stakeholders agrees that the organization method aids understanding of the requirements. There is no one optimal organization for all projects. An example outline of an SyRS is shown in Figure 7.

1. Introduction
- 1.1 System purpose
- 1.2 System scope
- 1.3 System overview
  - 1.3.1 System context
  - 1.3.2 System functions
  - 1.3.3 User characteristics
- 1.4 Definitions

2. References

3. System requirements
- 3.1 Functional requirements
- 3.2 Usability requirements
- 3.3 Performance requirements
- 3.4 Interface requirements
  - 3.4.1 External interface requirements
  - 3.4.2 Internal interface requirements
- 3.5 System operations
- 3.6 System modes and states
- 3.7 Physical characteristics
- 3.8 Environmental conditions
- 3.9 Security requirements
- 3.10 Information management requirements
- 3.11 Policy and regulation requirements
- 3.12 System life cycle sustainment requirements
- 3.13 Packaging, handling, shipping and transportation requirements

4. Verification

(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

NOTE This SyRS outline can be used, with tailoring, for subordinate specifications for system elements, even those that include software.

Figure 7 — Example SyRS Outline

NOTE Detailed content of the SyRS is found in 9.5.

8.5 Software requirements specification

8.5.1 General

The software requirements specification (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 acquirer, or by both.

It is important to consider the part that the SRS plays in the total project plan. 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 would be a requirement specification that states the interfaces between the system and its software portion, and places external performance and functionality requirements upon the software portion. Of course, the SRS should then agree with and expand upon these system requirements. The SRS indicates the precedence and criticality of requirements. The SRS defines all of the required capabilities of the specified software product to which it applies, as well as documenting the conditions and constraints under which the software has to perform, and the intended verification approaches for the requirements

8.5.2 SRS example outline

The specific requirements clause of the SRS should be organized such that a consensus of the system stakeholders agrees that the organization method aids understanding of the requirements. There is no one optimal organization for all systems. An example outline for an SRS is in Figure 8.

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Product overview
  - 1.3.1 Product perspective
  - 1.3.2 Product functions
  - 1.3.3 User characteristics
  - 1.3.4 Limitations
- 1.4 Definitions

2. References

3. Requirements
- 3.1 Functions
- 3.2 Performance requirements
- 3.3 Usability requirements
- 3.4 Interface requirements
- 3.5 Logical database requirements
- 3.6 Design constraints
- 3.7 Software system attributes
- 3.8 Supporting information

4. **Verification**
(parallel to subsections in Section 3)

5. Appendices
- 5.1 Assumptions and dependencies
- 5.2 Acronyms and abbreviations

Figure 8 — Example SRS Outline

NOTE 1 Detailed content of the SRS is found in 9.6.

Examples of organizational approaches to requirements in an SRS include:

— 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, degraded or emergency.

— 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 firefighters.

— 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.

— 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.

— 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.

— 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 pay checks, all functions associated with generating a current list of employees, etc.

— Functional hierarchy - when none of the above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by 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.

NOTE 2 There are many notations, methods and automated support tools available to aid in the documentation of SRS requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts can prove helpful; when organizing by object, object-oriented analysis can prove helpful; when organizing by feature, stimulus-response sequences can prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries can prove helpful.

9.6 Software requirements specification (SRS) content

9.6.1 SRS overview

This clause defines the normative content of the software requirements specification (SRS). The project shall produce the following information item content in accordance with the project's policies with respect to the software requirements specification. Organization of the content such as the order and section structure may be selected in accordance with the project's information management policies.

9.6.2 Purpose

Delineate the purpose of the software to be specified.

9.6.3 Scope

Describe the scope of the software under consideration by:

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

b) explaining what the software product(s) will do;

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

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

9.6.4 Product perspective

Define the system's relationship to other related products.

If the product is an element of a larger system, relate the requirements of that larger system to the functionality of the product covered by the SRS.

If the product is an element of a larger system, identify the interfaces between the product covered by the SRS and the larger system of which the product is an element.

Consider a block diagram showing the major elements of the larger system, interconnections and external interfaces.

Describe how the software operates within the following constraints:

a) system interfaces;

b) user interfaces;

c) hardware interfaces;

d) software interfaces;

e) communications interfaces;

f) memory;

g) operations;

h) site adaptation requirements; and

i) interfaces with services.

9.6.4.1 System interfaces

List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

9.6.4.2 User interfaces

Specify the logical characteristics of each interface between the software product and its users.

NOTE A style guide for the user interface can provide consistent rules for organization, coding and interaction of the user with the system.

9.6.4.3 Hardware interfaces

Specify the logical characteristics of each interface between the software product and the hardware elements 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.

9.6.4.4 Software interfaces

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, specify:

a) name;

b) mnemonic;

c) specification number;

d) version number; and

e) source.

NOTE It is acceptable to specify required platforms or operating systems, but rarely feasible to require a specific version. Typically, a version number most recent version or any currently maintain version can be specified for software.

For each interface, specify:

a) discussion of the purpose of the interfacing software as related to this software product;

b) 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.

9.6.4.5 Communications interfaces

Specify the various interfaces to communications such as local network protocols.

9.6.4.6 Memory constraints

Specify any applicable characteristics and limits on primary and secondary memory.

9.6.4.7 Operations

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; and

d) backup and recovery operations.

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

9.6.4.8 Site adaptation requirements

The site adaptation requirements include:

a) definition of 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) specification of the site or mission-related features that should be modified to adapt the software to a particular installation.

9.6.4.9 Interfaces with services

Specify interactions with services, e.g., Software as a Service (SaaS) or cloud services.

9.6.5 Product functions

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.

Use cases, user stories and scenarios are also used to describe product functions.

Note that for the sake of clarity:

a) the product functions should be organized in a way that makes the list of functions understandable to the acquirer 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.

9.6.6 User characteristics

Describe those general characteristics of the intended groups of users of the product including characteristics that may influence usability, such as educational level, experience, disabilities and technical expertise. This description should not state specific requirements, but rather should state the reasons why certain specific requirements are later specified in specific requirements in 9.6.9.

NOTE 1 Where appropriate, the user characteristics of the SyRS and SRS are consistent.

NOTE 2 For additional information on context of use and user needs, see ISO/IEC 25063 and ISO/IEC 25064.

9.6.7 Limitations

Provide a general description of any other items that will limit the supplier's options, including:

a) regulatory requirements and 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) quality requirements (e.g., reliability);

j) criticality of the application;

k) safety and security considerations;

l) physical/mental considerations; and

m) limitations that are sourced from other systems, including real-time requirements from the controlled system through interfaces.

9.6.8 Assumptions and dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors 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 have to change accordingly.

9.6.9 Apportioning of requirements

Apportion the software requirements to software elements. For requirements that will require implementation over multiple software elements, or when allocation to a software element is initially undefined, this should be so stated. A cross-reference table by function and software element should be used to summarize the apportionments.

Identify requirements that may be delayed until future versions of the system (e.g., blocks and/or increments).

9.6.10 Specified requirements

Specify the software system requirements to a level of detail sufficient for software design, development and verification of the software increment or release in process.

The requirements should:

a) be stated in conformance with all the characteristics described in 5.2 of this document;

b) be cross-referenced to earlier versions or related documents;

c) be uniquely identifiable;

d) describe every input (stimulus) into the software system, every output (response) from the software system, and all functions performed by the software system in response to an input or in support of an output.

9.6.11 External interfaces

Define all inputs into and outputs from the software system. The description should complement the interface descriptions in 9.6.4.1 through 9.6.4.5, and should not repeat information there.

Each interface defined should include the following content:

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) data formats;

i) command formats; and

j) data items or information included in the input and output.

9.6.12 Functions

Define the fundamental actions that have to take place in the software in accepting and processing the inputs and in processing and generating the outputs, including:

a) validity checks on the inputs;

b) exact sequence of operations;

c) responses to abnormal situations, including:

   1) overflow;

   2) communication facilities;

   3) hardware faults and failures; and

   4) error handling and recovery;

d) effect of parameters;

e) relationship of outputs to inputs, including:

   1) input/output sequences; and

   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.

9.6.13 Usability requirements

Define usability and quality in use requirements and objectives for the software system that can include measurable effectiveness, efficiency, satisfaction criteria and avoidance of harm that could arise from use in specific contexts of use.

NOTE Additional guidance on usability requirements can be found in ISO/IEC TR 25060.

9.6.14 Performance requirements

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; and

c) the 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.

The performance 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.

9.6.15 Logical database requirements

Specify the logical requirements for any information that is to be placed into a database, including:

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) security; and

g) data retention requirements.

9.6.16 Design constraints

Specify constraints on the system design imposed by external standards, regulatory requirements or project limitations.

9.6.17 Standards compliance

Specify the requirements derived from existing standards or regulations, including:

a) report format;

b) data naming;

c) accounting procedures; and

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 shall be recorded in a trace file with before and after values.

9.6.18 Software system attributes

Specify the required attributes of the software product. The following is a partial list of examples:

a) Reliability - specify the factors required to establish the required reliability of the software system at the time of delivery.

b) Availability - specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery and restart.

c) Security - specify the requirements to protect the software from accidental or malicious access, use modification, destruction or disclosure. Specific requirements in this area could include the need to:

    1) utilize certain cryptographic techniques;

    2) keep specific log or history data sets;

3) assign certain functions to different modules;

4) restrict communications between some areas of the programme;

5) check data integrity for critical variables; and

6) assure data privacy.

d) Maintainability - specify attributes of software that relate to the ease of maintenance of the software itself. These may include requirements for certain modularity, interfaces or complexity limitation. Requirements should not be placed here just because they are thought to be good design practices.

e) Portability - specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems, including:

1) percentage of elements with host-dependent code;

2) percentage of code that is host dependent;

3) use of a proven portable language;

4) use of a particular compiler or language subset; and

5) use of a particular operating system.

9.6.19 Verification

Provide the verification approaches and methods planned to qualify the software. The information items for verification are recommended to be given in a parallel manner with the information items in 9.6.10 to 9.6.18.

9.6.20 Supporting information

Additional supporting information to be considered includes:

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; and

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

The SRS should explicitly state whether or not these information items are to be considered part of the requirements.