MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National aerospace university n. a. N. E. Zhukovsky
“Kharkiv aviation institute”

O. A. Illiashenko, V. S. Kharchenko

SOFTWARE REQUIREMENTS SPECIFICATION FOR COMPUTER SYSTEMS AND NETWORKS

Part 1

Tutorial

Kharkiv "KhAI" 2014

UDC 004.4:004.7:004.01(075.8)
I54

Розглянуто основні етапи проектування технічного завдання для розроблення програмного забезпечення на основі міжнародних нормативних документів, що регулюють питання специфікування. Визначено, яке місце вимоги до програмного забезпечення посідають у загальному процесі розроблення вимог до програмного продукту. Викладено основні питання розроблення технічного завдання на програмне забезпечення, проаналізовано специфіку цього процесу. Наведено приклади технічних завдань на програмне забезпечення, а також приклади шаблонів та інструкцій щодо їх створення.

Для студентів університетів, які вивчають комп'ютерні науки (зокрема, комп'ютерну та програмну інженерію), аспірантів, викладачів, науковців, інженерів-розробників, тестувальників і співробітників служб забезпечення якості, а також для фахівців у галузі розроблення, верифікації й експертизи програмного забезпечення, експертів, які займаються проблемами якості та надійності програмного забезпечення і комп'ютерних систем.

Може використовуватися під час написання курсових і дипломних робіт, а також розроблення технічних завдань на програмне забезпечення у межах навчальної та позанавчальної роботи студентами та співробітниками кафедри комп'ютерних систем і мереж Національного аерокосмічного університету ім. М. Є. Жуковського "ХАІ".

Reviewers: DrS, Prof V. I. Hahanov (Kharkov National University of Radioelectronics, Kharkov, Ukraine),
PhD, Prof A. P. Plahteev (Kharkov National Automobile and Highway University, Kharkov, Ukraine)

Illiashenko, O.A.

I54 Software specification requirements for computer systems and networks. Part 1 [Text]: tutorial / O. A. Illiashenko, V. S. Kharchenko. – Kharkov: National aerospace university n. a. N. E. Zhukovsky "Kharkov aviation institute", 2014. – 40 p.

The main stages of requirements specification development, based on international regulation documents in the field of requirements specification for software development, are reviewed. The place of software requirements in overall requirements development process is determined. The general questions of software specification are considered and the features of the requirements specification process are analyzed. Examples of software specifications, as well as templates with instructions on how to create them, and are given.

The tutorial is intended for university students, studying computer science (especially, computer and software engineering), PhD students, teachers, professors, engineers, developers, testers and quality assurance services staff, as well as specialists in the design, verification, and expertise of software, and experts, dealing with quality and reliability of software and computer systems.

It can be used when writing of courseworks and dissertations, and developing of software specifications within the academic and extracurricular work of students and staff of National Aerospace University. Zhukovsky "KhAI".

Fig. 2. Bibliographic.: 23 nam.

UDC 004.4:004.7:004.01(075.8)

CONTENTS

ACRONYMS AND ABBREVIATIONS

ACK/NACK – Acknowledged / Not Acknowledged

ConOps – Concept of Operation

DBMS – Database Management System

IEEE – Institute of Electrical and Electronics Engineers

IEC – International Electrotechnical Commission

ISO – International Organization for Standardization

MTBF – Mean Time Between Failures

OpsCon –System Operational Concept

RAM – Random Access Memory

ROM – Read-only memory

SRS – Software Requirements Specification

StRS – Stakeholder Requirements Specification

SyRS – System Requirements Specification

TBD – To Be Determined

XON-XOFF – Software Flow Control

1 INTRODUCTION

This tutorial describes typical structure and contents of software requirements specification (SRS) in accordance with international standards, regulatory documents, guidelines, templates, industrial examples, and best practices**.** *Reference* section includes all the papers we used to prepare this tutorial.

Software requirements are based on the current international standard ISO/IEC/IEEE 29148-2011 “Systems and software engineering – Life cycle processes – Requirements engineering”. [1] This standard is a new version which replaces superseded standards IEEE 830-2011 “IEEE Recommendation practice for software requirements specifications”, IEEE 1233-1998 “IEEE Guide for developing system requirements specifications”, and IEEE 1362 – 1998 “IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document” and includes the revised provisions of these regulatory documents. [2 – 4] We used [2-10] to provide our readers with necessary clarifications. We included a number of other standards and guides to the *Reference* section of this tutorial as we wanted to be sure that everybody interested will get deep and profound understanding of requirements management process.

To make our readers acquainted with real examples of SRS industrial use, we referred to the regulatory documents delivered by the US Nuclear Regulatory Commission, namely NUREG/CR-6734 “Digital Systems Software Requirements Guidelines”. [11, 12] this profound piece of two volumes both reviews high-integrity SRSs for nuclear power plants and describes many failures which occur if these requirements are not properly met.

As an industrial example of requirements specification and guideline and its application the US Nuclear Regulatory Commission Regulation documents could be considered NUREG/CR-6734 “Digital Systems Software Requirements Guidelines” of two volumes [11, 12] first of which identifies guidelines that review and the second provide descriptions of 45 failures that are linked to software requirements review guidelines listed in the first volume. The failure descriptions are "lessons learned" illustrating why specific software requirements guidelines are needed.

This tutorial is provided with three appendices:

- Appendix A – *Software requirements specification outline*
- Appendix B – *Examples of software requirements specifications*
- Appendix C – *Examples of software requirements specifications templates and instructions:*

Appendix A gives an example of SRS outline that is based on Russian versions of standards on requirements specification, software lifecycle processes, etc. This SRS outline was described in detail in [13].

Appendices B and C provide our readers with different examples of software requirements specifications as well as SRS templates, guidelines, and instructions. These and other documents can be easily found in the Internet so we give them only for educational purposes. Authors of this tutorial are not responsible for their content and their further use.

This **tutorial is intended to consider main stages** and phases of SRS development. We also want to show our readers what **problems** and **inconsistences** may occur during the development process.

This tutorial does not give precise design rules, because they are defined by organizations and entities which develop SRS and, therefore, may vary greatly. We consider only several **most important issues in SRS development**.

Before turning to explanations, we want to remind our readers that, depending on the precise characteristics of software, SRS developers are allowed to **specify** the **content**, to **combine** or **omit** sections as well as **introduce** new ones, to **change** the section **names**. The final SRS content must be agreed with all stakeholders. All deviations from the standard SRS scheme must be discussed and agreed upon by the parties involved in the software development.

This tutorial describes the following SRS sections:

1.1. Purpose
1.2. Scope
1.3. Product perspective
    1.3.1. System interfaces
    1.3.2. User interfaces
    1.3.3. Hardware interfaces
    1.3.4. Software interfaces
    1.3.5. Communications interfaces
    1.3.6. Memory constraints
    1.3.7. Operations
    1.3.8. Site adaptation requirements
1.4. Product functions
1.5. User characteristics
1.6. Constraints and limitations
1.7. Assumptions and dependencies
1.8. Apportioning of requirements
1.9. Specific requirements
1.10. External interfaces
1.11. Functions

1.12. Usability requirements
1.13. Performance requirements
1.14. Logical database requirements
1.15. Design constraints
1.16. Standards compliance
1.17. Software system attributes
1.18. Verification
1.19. Supporting information

Numbering of sections, subsections, and paragraphs can be used hereinafter to prepare documents.

Note that **all the options we give here are only examples which means that they are neither exhaustive nor a dogma.**

When working on SRSs, we recommend that developers use numbered lists to allow convenient referring in the future. For example, if you have a list of 8 items in paragraph 4.3.5, you can refer to the 8th item in such a way: "*The validation technique of 4.3.5 (8)*", or "*The requirements of 4.3.5 (8) are met in full*".

When using specific, or uncommon, or both, terminology, we recommend that SRS is provided with a glossary of terms used in SRS (or document(s) referred to) together with their appropriate definitions.

2 GENERAL INFORMATION ABOUT SOFTWARE REQUIREMENTS SPECIFICATION

Requirement is a property which must be met in order to solve a problem. Software requirement is a property which software must meet to solve a particular problem it is developed or adapted for. Software is usually developed to automate certain tasks which stand before the software user; to support business processes of the organization which commissioned it; to correct shortcomings of already existing software, to control devices, etc. Tasks which should be solved by software are typically complex. Therefore, software requirements are typically complex combinations of requirements put forward by different people at different levels of organization proceeding from the environment in which the software will operate.

According to ISO/IEC/IEEE 29148:2011, **software requirements specification** is a *structured collection of the requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces*.

ISO/IEC/IEEE 29148 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Software & Systems Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

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. SRSs are written by one or more representatives of a supplier, a customer, or both. It is a two-way insurance policy that secures that both the client and the developer understand the each other's requirements.

SRS should be written in a precise, clear, and plain language so that it is understandable for business analysts, customer representatives or all other people with minimal technical expertise. However, it may also include analytical models (case diagrams, entity relationship diagrams, data dictionary, etc.) which may be used for detailed design and development of software systems. High-quality SRS is critical for software development since it acts as the bridge between software developers and business analysts. *Incomplete or incorrect SRS may have disastrous effects on a software project.*

Basic issues that SRS writer(s) will address are the following:

1. **Functionality.** What is the software supposed to do?
2. **External interfaces.** How should the software interact with people, system hardware, other hardware, and other software?
3. **Performance.** At what speed (availability, response time, recovery, etc.) should software work?
4. **Attributes.** What should the software's portability, correctness, maintainability, security, etc. be?
5. **Design constraints imposed on implementation.** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s), etc.?

SRS are intended to:

– *communicate understanding of requirements:* explains both the domain and the systems to be developed;

– *express agreement and commitment:* SRS is contractual and can be legally binding;

– *provide a baseline for evaluating subsequent products:* supports system testing, verification, and validation, should have enough data to verify whether the delivered system meets requirements;

– *provide a baseline for change control:* requirements change, software evolves.

We should note that SRSs include only functional and non-functional requirements: They do not offer development decisions, or possible solutions to any technology or business issues, or any other information other than what the development team understands the customer’s system requirements to be.

SRSs are typically developed at the stage of *requirements development* which is the initial product development phase at which information is gathered about what requirements must or must not be met. This data gathering stage may include onsite visits, questionnaires, surveys, interviews, and, perhaps, return-on-investment (ROI) analysis. It also requires analysing customer’s current business environment. The actual specification is written after the requirements have been gathered and thoroughly analysed.

3 SOFTWARE REQUIREMENTS SPECIFICATION vs. OVERALL REQUIREMENTS

It is important to consider the part that the SRS plays in the total project plan. Software may provide all the functionality of the project or it may be part of a larger system. In the latter case, there is typically a requirements specification for the interfaces between the system and its software portion which will specify external performance and functionality requirements for the software portion. Of course, the overall project SRS should then agree with and expand upon these system requirements. SRS indicates the precedence and criticality of requirements. It defines all capabilities required from the specified software product to which it applies as well as sets conditions and constraints under which the software has to perform. Finally, it describes intended verification approaches for the requirements.

This clause describes the relationship between the requirements processes and requirement information items by illustrating a typical application style in a project.

Requirements processes and their resultant specifications depend on the scope of the system for which the requirements are defined. Requirements for a system or system elements to be developed or changed are subject to organization level requirements for business or organizational operation. System requirements or requirements for system elements are defined for the lowest levels of system. A typical scope of a system and the corresponding requirement traced in it is given in Figure 3.1.

Stakeholder's requirements specification (StRS), **system requirements specification (SyRS)**, and **software requirements specification (SRS)** represent different sets of requirements.

These specifications correspond to the requirements shown in Figure 3.2 as follows:

– **StRS** – Stakeholder's Requirements (levels of business management and business operation);

– **SyRS** – System Requirements;

– **SRS** – Software Requirements.

These requirements may be applied to multiple specifications (instances) iteratively or recursively. An example of a sequence of requirements processes and specifications is given in Figure 3.2.

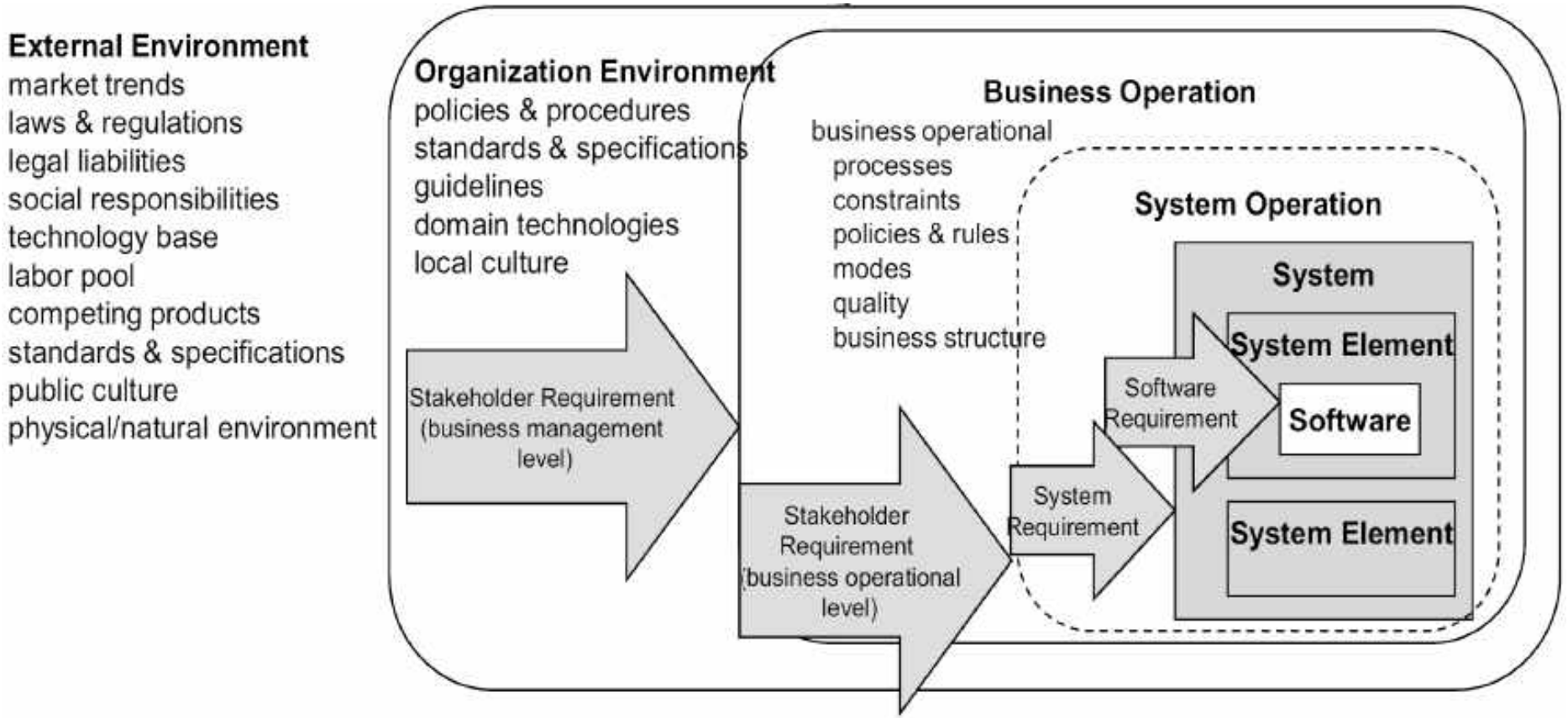

Figure 3.1 – Scope of requirement in business context

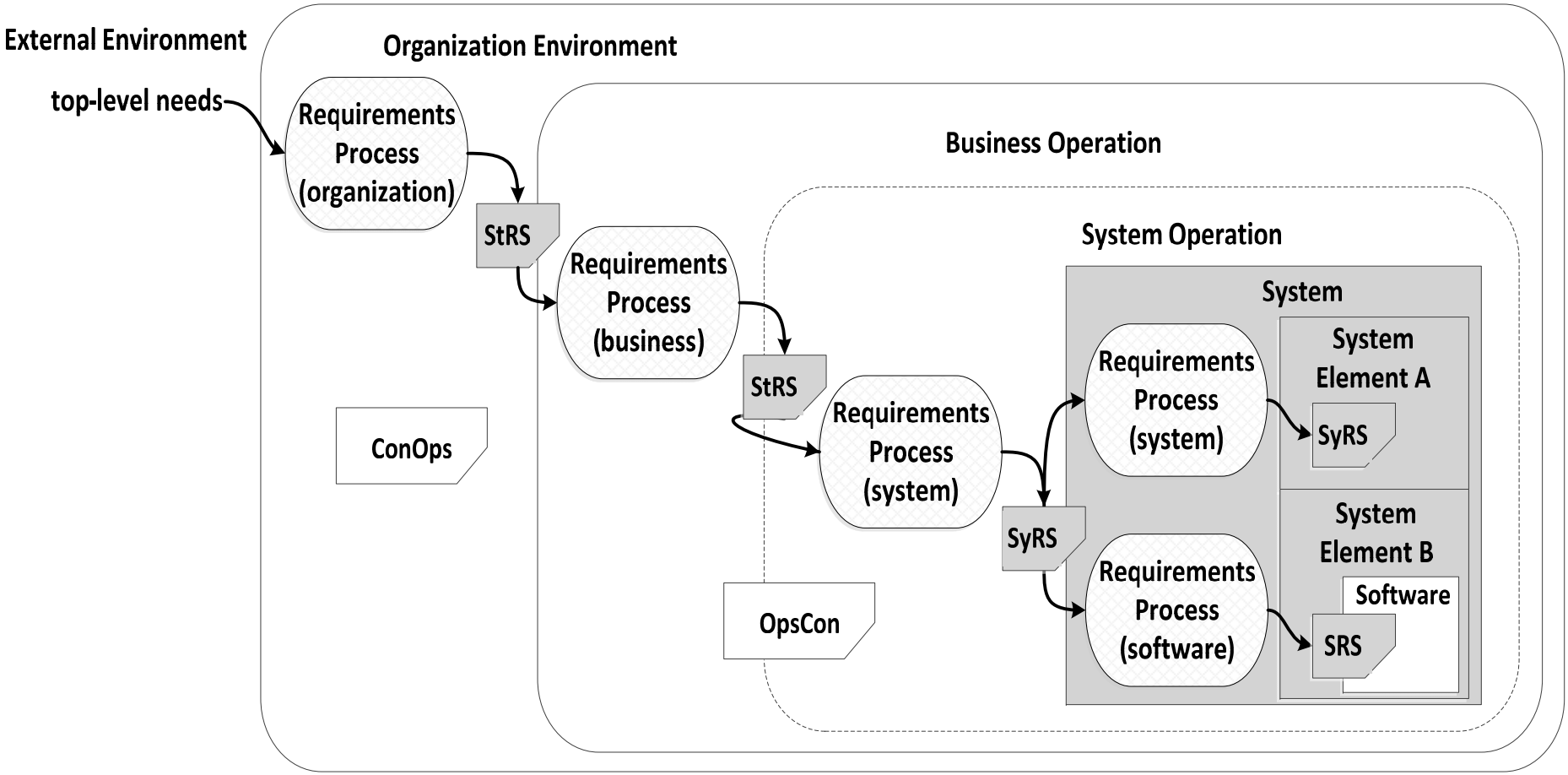

Figure 3.2 – Possible sequence of requirements processes and specifications

SyRS specifies requirements for *a system* or *a system element*. SyRS may also *specify software requirements*.

SRS set lower level software requirements for *software specific elements* of systems or system elements.

Concept of Operation (ConOps) and the **System Operational Concept** (OpsCon) are useful for eliciting requirements from various stakeholders and as a practical means of communicating and sharing organization's intentions. At the organization level, **ConOps** addresses the *leadership's intended way of operating the organization*. It may refer to the use of one or more systems as “black boxes”. **OpsCon** addresses *specific systems-of-interest from the user's view point*.

Requirements represented by StRS, SyRS, SRS, ConOps, and OpsCon are interdependent. Their development requires interaction and cooperation, particularly in relation to business processes, organizational practices, and technical solutions.

Different types of systems may have parallel documentation for the requirements they should meet. However, in general, they start with StRS and SyRS and include software specifications as well as specifications for hardware and interfaces.

4 CHARACTERISTICS OF SOFTWARE REQUIREMENTS SPECIFICATION

Desirable characteristics of SRSs are put together in Table 4.1.

Table 4.1 – Quality measures of individual SRS statements

SRS Quality Characteristic	Description
Valid, or "correct"	A valid SRS is the one understandable, analysable, acceptable for, or approved by all parties and project participants. This is one of the main reasons SRSs are written using natural languages. An SRS is correct if, and only if, every requirement stated therein is one that software shall meet. There is no tool or procedure that ensures correctness. The SRS should correspond to any applicable superior specifications such as system requirements specifications, other project documentation, and with other applicable standards. Customers and users should be able to determine whether the SRS reflects their needs correctly. Traceability makes this procedure easier and less prone to errors.
Complete	A SRS is complete if, and only if, it includes the following elements: 1. All significant requirements for functionality, performance, design constraints, attributes, or external interfaces. Any external requirements imposed by the system specification should be acknowledged and treated. 2. Software responses to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify responses to both valid and invalid input values. 3. Full labels and references to all figures, tables, and diagrams in the SRS as well as definitions of all terms and measure units.
Unambiguous	A SRS is unambiguous if, and only if, every requirement stated therein has only one possible interpretation. At the very least, this requires all characteristics of the final product be described using one unique term. In cases when terms used in particular contexts have several meanings, these terms should be included in a glossary where its meaning is specified. SRS should be unambiguous both to those who develop it and to those who use it. *However, developers and users often do not have the same background and therefore do not describe software requirements in the same way. Representations that improve requirements specification for the developer may be counterproductive because they prevent the user's understanding and vice versa.*
Verifiable	A SRS is verifiable if, and only if, every specified requirement is verifiable i.e. there exists a procedure to check that final software meets the requirement (a process exists that tests whether every requirement is met, and every requirement is specified behaviourally). If a method cannot be devised to determine whether the software meets a particular requirement, then this requirement should be removed or revised.

Table 4.1 (Continued)

SRS Quality Characteristic	Description
	If a method cannot be devised to determine whether the software meets a particular requirement, then this requirement should be removed or revised. *Non-verifiable requirements include statements such as "works well", "good human interface", and "shall usually happen". These requirements cannot be verified because it is impossible to define the terms "good", "well", or "usually". The statement "the program shall never enter an infinite loop" is non-verifiable because this quality is theoretically impossible to test.*
Consistent	Consistency refers here to *internal consistency*. If a SRS does not agree with some higher-level document such as a system requirements specification then it is not correct. A SRS is *internally consistent* if, and only if, no subset of individual requirements described in it conflict. Three likely SRS conflicts are: 1. Conflict between specified characteristics of real-world objects, e.g. requirements call for tabular/textual output report or for green/blue lights. 2. Logical or temporal conflict between two specified actions, e.g. requirements specify that the program will add two inputs / multiply them; 3. Requirements state that *A* must always follow *B* / that *A* and *B* occur simultaneously. Requirements may describe the same real-world object but use different terms for this, e.g. requirements call the program's request for a user input *prompt* / *cue*. Standard terminology and definitions promote consistency.
Feasible	A SRS is feasible if, and only if, all requirements are technically achievable, do not require major technology advances, and fit within system constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk.
Traceable	A SRS is traceable if the origin of each requirement is clear, and if the SRS facilitates referencing to each requirement in future development or enhancement documentation. There are two types of traceability: a) *Backward traceability (i.e., to previous stages of development):* each requirement refers explicitly to its source in earlier documents. b) *Forward* traceability *(i.e., to all documents spawned by the SRS):* each requirement in the SRS has a unique name or reference number. *SRS forward traceability is especially important when software products enter operation and maintenance phase. As code and design documents are modified, it is essential to ascertain the complete set of requirements that can be affected by those modifications*. □□
Modifiable	A SRS is modifiable if its structure and style are such that any necessary change may be made easily while preserving completeness

Table 4.1 (Continued)

SRS Quality Characteristic	Description
	and consistency (SRS should have good structure and secure cross-referencing).
Ranked for importance and/or stability	A SRS is ranked for importance and/or stability if importance and stability of the requirements is indicated for each requirement. Typically, all requirements relating to a software product are not equally important: Some requirements are essential, especially for critical applications (mission-critical, safety-critical, security-critical, business-critical, life-critical, etc.), while others are only desirable.

SRS are used by:

– *users, purchasers* – mostly interested in system requirements, not in detailed software requirements;

– *systems analysts* – write interrelated specifications;

– *developers, programmers* –implement the requirements;

– *testers* – determine whether requirements have been met;

– *project managers* – measure and control analysis and development processes.

SRS writer(s) **should avoid** the following mistakes:

– including design or project requirements into the SRS;

– describing any design or implementation details: These should be described in the design stage of the project;

– addressing software products not the process of their production.

5 SOFTWARE REQUIREMENTS SPECIFICATION DEVELOPMENT

Stakeholder's requirements, system requirements, and system element requirements should be formulated before starting SRS development. SRS uses precise and explicit language to state which functions and capabilities should be provided by the software system (software application, eCommerce Web-site, etc.) as well as to describe constraints which the system should follow.

It is important to agree on specific keywords and terms that signal about a requirement.

Goddard Space Flight Center (GSFC) studied dozens of NASA requirements specifications and found out **nine categories of SRS quality indicators.** Individual components of each category are words, phrases, and sentences that are related to quality attributes. These nine categories fall into two classes, those related to individual specification statements, and those related to the total SRS document. Table 5.1 summarizes classes, categories, and components of these quality indicators. This table was originally presented at the April 1998 Software Technology Conference presentation "Doing Requirements Right the First Time". **Common approach** is **to stipulate following quality measures** is described in Table 5.1.

Table 5.1 – Quality measures of individual SRS statements

Cat No.	Description of quality category	
1	***Imperatives*** Words and phrases that command the presence of some feature, function, or deliverable. Are listed below in decreasing order of strength.	
	shall	Used to dictate the provision of a functional capability.
	must, must not	Most often used to establish performance requirement or constraints.
	is required to	Used as an imperative in SRS statements when written in passive voice.
	are applicable	Used to include, by reference, standards, or other documentation as an addition to the requirement being specified.
	responsible for	Used as an imperative in SRSs written for systems with pre-defined architectures.
	will	Used to cite things that the operational or development environment is to provide to the capability being specified: *The vehicle's exhaust system will power the ABC widget.*
	should	Usually not used as an imperative in SRS statements; when used, the SRS statement always reads weak. Avoid using "should" in your SRSs.

Table 5.1 (Continued)

Cat No.	Description of quality category
2	***Continuances*** Phrases that follow an imperative and introduce the specification of requirements at a lower level. There is a correlation with the frequency at which continuances are used and SRS organization and structure. Excessive use of continuances often indicates a very complex, detailed SRS. Use continuances in your SRSs but balance their frequency with the appropriate level of detail called for in the SRS. Continuances (listed in decreasing order of use): *below, as follows, following, listed, in particular, support.*
3	***Directives*** Categories of words and phrases that indicate illustrative information within the SRS. A high ratio of total number of directives to total text line count appears to correlate with how precisely requirements are specified within the SRS. Incorporate directives in your SRSs. Directives (listed in decreasing order of use): *figure, table, for example, note.*
4	***Options*** A category of words that provide latitude in satisfying SRS statements containing them. This category of words loosens the SRS, reduces the client's control over the final product, and allows for possible cost and schedule risks. You should avoid using them in your SRS. Options (listed in increasing order of use): *can, may, optionally*.
5	***Weak phrases*** A category of clauses that can create uncertainty and multiple/subjective interpretation. Total number of weak phrases found in an SRS indicates relative ambiguity and incompleteness of SRSs. Weak phrases (listed alphabetically): *adequate, as a minimum, as applicable, as appropriate, at a minimum, be able to, be capable of, but not limited to, capability of, capability to, capability to, easy, effective, if possible, if practical, normal, provide for, timely, TBD*.
6	***Size*** Used to indicate the size of the SRS document, and is the total number of *lines of text, number of imperatives, subjects of SRS statements, Paragraphs*.
7	***Text Structure*** Related to the number of statement identifiers found at each hierarchical level of the SRS. Indicates document organization, consistency, and level of detail. Most detailed NASA SRSs are nine levels deep. High-level SRSs are rarely more than four levels deep.

Table 5.1 (Continued)

Cat No.	Description of quality category
	Hourglass-shaped *text structure* (many level-1 headings, few mid-level headings, and many lower-level headings) usually has a greater amount of introductory and administrative information. Diamond-shaped *text structure* (pyramid shape followed by decreasing statement counts at levels below the pyramid) indicates that subjects introduced at higher levels are addressed at various levels of detail.
8	***Specification Depth*** Number of imperatives found at each of the SRS levels of text structure. It includes the count of lower level list items that are introduced at a higher level by an imperative and followed by a continuance. Provides some insight into how much of the requirements document was included in the SRS, and can indicate how concise the SRS is in specifying the requirements.
9	***Readability Statistics*** Measurements of how easily an adult can read and understand the requirements document. While readability statistics provides a relative quantitative measure, don't sacrifice sufficient technical depth in your SRS for a number. Four readability statistics (calculated by Microsoft Word) usually used: *Flesch Reading Ease index, Flesch-Kincaid Grade Level index, Coleman-Liau Grade Level index, Bormuth Grade Level index*.

All terms specific to requirements engineering should be formally defined and applied consistently throughout all requirements of the system. Figure 5.1 shows several examples of syntax that is allowed in SRS.

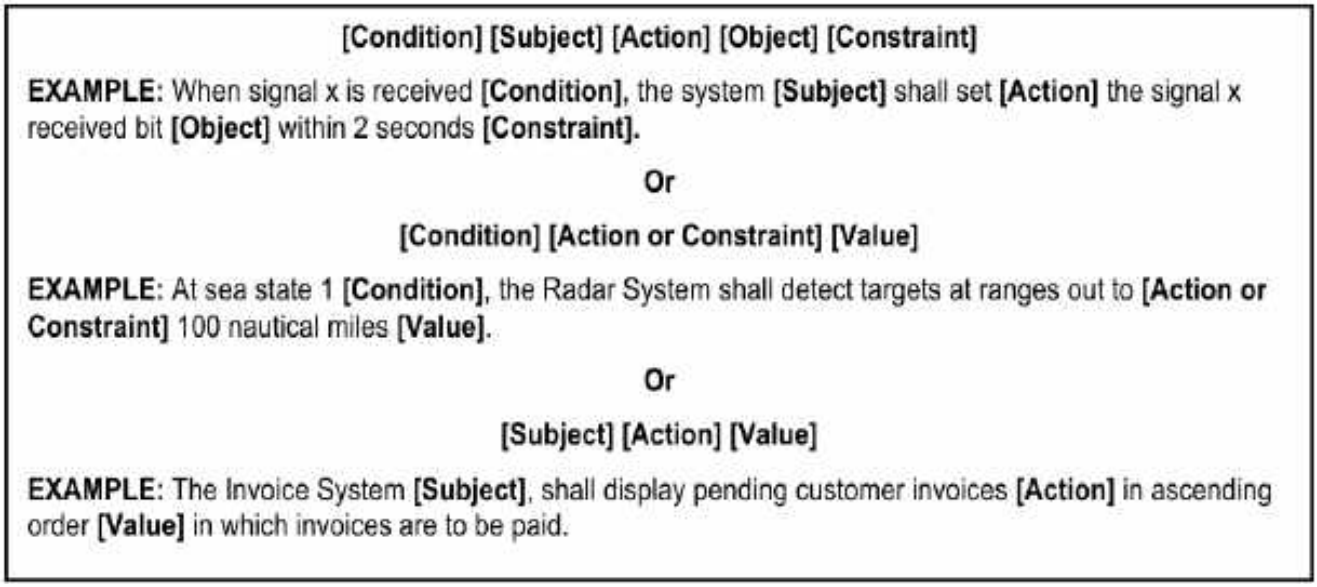

Figure 5.1 – Examples of Requirement Syntax

Conditions are measurable qualitative or quantitative attributes that are stipulated for a requirement. They qualify requirements at a greater detail and provide attributes which permit requirements to be formulated and stated in a manner that can be validated and verified. Conditions may limit the options open to developers.

It is important to transform stakeholder's needs into stakeholder's requirements without imposing unnecessary bounds on the solution space.

Constraints restrict the development solution or implementation of the system's engineering process. Constraints may apply across all requirements, may be specified in a relationship to a specific requirement or set of requirements, or may be identified as stand-alone requirements (i.e., not bounding any specific requirement).

Constraints include:

– interfaces to already existing systems (e.g., format, protocol, or content) where interfaces cannot be changed,

– physical size limitations (e.g., controllers should fit in a limited space of airplane wings),

– laws of a particular country,

– available duration or budget,

– pre-existing technology platform,

– user or operator capabilities and limitations.

Requirements may be ranked or weighted to indicate priority, timing, or relative importance. In a scenario form, requirements depict the system's operation from a user's perspective.

6 LANGUAGE CRITERIA FOR SOFTWARE REQUIREMENTS SPECIFICATIONS

Requirements are often written in natural languages (e.g., English). Natural languages are inherently ambiguous. A natural-language **SRS should be reviewed by an independent party** to identify ambiguous use of language so that it can be corrected.

When you write textual requirements follow these considerations in order to ensure that good requirements characteristics are employed.

Requirements should state *“what”* is needed, not *“how”*. Requirements should state only what is needed for the system-of-interest and not include any development decisions. However, as requirements are allocated and decomposed through the levels of the system, develop decisions / solution architectures should be recognised as defined at higher levels. This is the iterative and recursive application of requirement analysis and architectural design.

Requirements should avoid vague and general terms, otherwise they are difficult or even impossible to verify or may allow for multiple interpretations. You better avoid such unbounded or ambiguous terms as:

– superlatives (*"best"*, *"most"*);

– subjective language ("*user friendly"*, *"easy to use"*, *"cost effective"*);

– vague pronouns ("*it"*, *"this"*, *"that"*);

– ambiguous adverbs and adjectives ("*almost always"*, *"significant"*, *"minimal"*);

– open-ended, non-verifiable terms (*"provide support"*, *"but not limited to", "as a minimum"*);

– comparative phrases ("*better than"*, *"higher quality"*);

– loopholes ("*if possible"*, *"as appropriate"*, *"as applicable"*);

– incomplete references (not specifying references with their dates and version numbers; not specifying applicable parts of references to restrict verification);

– negative statements (*"system capability not to be provided"*).

All assumptions regarding requirements should be documented and validated either by attributes associated with certain requirements or in accompanying documents which include definitions as declarative statements, not requirements.

7 SOFTWARE REQUIREMENTS SPECIFICATION CONTENT

This section introduces normative content of software requirements specification (SRS). Organization of information in the document such as order and structure of sections is selected in accordance with projects' documentation policies.

7.1 Purpose

Specifies the purpose of the software.

7.2 Scope

Describes the scope of the software under consideration by:

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

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

c) describing how software will be applied, including relevant benefits, objectives, and goals;

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

7.3 Product perspective

Defines system's relationships with other products.

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

A block diagram can be helpful which shows major elements of the larger system, interconnections, and external interfaces.

You should provide necessary information on:

a) system interfaces;

b) user interfaces;

c) hardware interfaces;

d) software interfaces;

e) communications interfaces;

f) memory;

g) operations;

h) site adaptation requirements.

Below all the elements of this list are described starting with system requirements and finishing with site adaptation requirements.

7.3.1 System interfaces

Lists system interfaces; identifies software functionality to accomplish system requirements; and describes interfaces to match the system.

7.3.2 User interfaces

Specifies:

a) logical characteristics of each interface between the software product and its users. This includes configuration characteristics necessary to accomplish the software requirements (*e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys*);

b) all aspects of optimizing the system interface with the person who uses, maintains, or provides other support to the system. This may simply comprise a list of do's 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*. This may also be specified in "Software System Attributes" in a section titled "Ease of Use".

Style guides for user interfaces can provide consistent rules for their organization, coding, and user-system interaction.

7.3.3 Hardware interfaces

Specifies logical characteristics of each interface between the software product and the system hardware. This includes configuration characteristics (*number of ports, instruction sets, etc.*) and protocols and covers such matters as what devices are to be supported, and how they are to be supported. *For example, terminal support may specify full-screen support as opposed to line-by-line support.*

7.3.4 Software interfaces

Specifies use of other required software products (*e.g., data management systems, operating systems, mathematical packages, computer-aided design tools, or integrated development environments*) as well as interfaces with other application systems (*e.g., linkage between an accounts receivable system and a general ledger system*).

For each required software product, you should specify:

a) name;

b) specification number;

c) version number;

For each interface, you should:

a) discuss the purpose of interfacing software as related to developed or additional software product;

b) define the interface in terms of message content and format (not necessary for well-documented interface if you refer to the interface documentation).

7.3.5 Communications interfaces

Specifies interfaces to communications *such as local network protocols*.

7.3.6 Memory constraints

Specifies all applicable characteristics and limits on RAM and ROM.

7.3.7 Operations

Specifies normal and special operations required by users such as:

a) operation modes with respect to the user organization (*e.g., user-initiated operations*);

b) periods of interactive and unattended operations;

c) data processing support functions;

d) backup and recovery operations;

This is sometimes specified in the "User Interfaces" section.

7.3.8 Site adaptation requirements

This section includes:

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

7.4 Product functions

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

Sometimes function summary that is necessary as 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) product functions should be described in a way that is 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 different functions and their relationships. Such a diagram is not intended to demonstrate product design but simply shows logical relationships among variables.

7.5 User characteristics

Describes the intended groups of the product users including characteristics that may influence usability such as education, experience, disabilities, and technical expertise. The section should not include specific requirements but rather state reasons why certain specific requirements are later specified in specific requirements (7.9).

Where appropriate, user characteristics of SyRS and SRS should be consistent.

7.6 Limitations

Provides general description of any other items that limit supplier's options, including

a) regulatory policies (*e.g. the Software should be in compliance with standard XXX; The requirement functioning that is described in section XXX can be granted only if all conditions from XXX are met*);

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

c) interfaces to other applications (e*.g. the web-server should have Apache Version 2.0 application installed)*;

d) parallel operation;

e) audit functions (*e.g. after each 10000 operation hours, the Software should be audited for the purposes of Standard XXX compliance; The audit of XXX Software should be performed each 500 hours from the installation on-site in order to comply with safety/security requirements of XXX normative document. The audit should be performed with the use of YYY technique…*);

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) application criticality (*e.g. the Software belongs to safety-critical applications, so it must comply with additional safety requirements specifications listed below:….; the Web-application developed for remote internet banking is of business-critical type, so it should ensure real-time on-line transactions…*);

k) safety and security considerations (*e.g. MTBF should be calculated according to MIL-HDBK-217F and should be 30000 hours; Each software audit should include evaluation of security requirements; A configuration management (…description…) and corrective action process (…list of actions with their detailed description…) should be in place to provide security for the Software and to ensure that any proposed changes do not inadvertently create security violations or vulnerabilities (list of vulnerabilities with their criticality analysis)*);

l) physical/mental considerations.

7.7 Assumptions and dependencies

Lists factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to them can affect 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*.

7.8 Apportioning of requirements

Apportions software requirements to software elements, e.g. implementation over multiple software elements or apportionments to software elements initially undefined. Cross reference tables by function and software elements should be used to summarize the apportionments.

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

7.9 Specific requirements

Specifies all software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements.

Specifies all software requirements to a level of detail sufficient to enable testers to test that the software system satisfies those requirements.

At a minimum, describes 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.

Specific requirements should:

a) conform with all the characteristics described earlier;

b) cross-refer to earlier relative documents if any;

c) be uniquely identifiable.

7.10 External interfaces

Defines all inputs into and outputs from the software system. The section should complement interface descriptions (7.3.1 – 7.3.5) and should not repeat their content.

For each interface defined, you should include:

a) name;

b) purpose;

c) input source or output destination;

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

e) measure units;

f) timing;

g) relationships with other inputs/outputs;

h) screen formats/organization;

i) window formats/organization;

j) data formats;

k) command formats;

l) end messages.

7.11 Functions

Defines fundamental actions that have to take place then the software accepts and processes inputs; or processes and generates outputs, including

a) validity checks of the inputs;

b) exact sequence of operations;

c) responses to abnormal situations including overflow, communication facilities, and error handling and recovery;

d) parameters effects;

e) output-to-input relationships including input/output sequences, formulas for input to output conversion.

It may be appropriate to partition functional requirements into subfunctions or subprocesses. This does not imply that software design will also be partitioned.

7.12 Usability requirements

Defines usability (quality in use) requirements. Usability requirements and objectives for the software system include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use.

7.13 Performance requirements

Specifies both 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:

a) number of terminals to be supported;

b) number of simultaneous users to be supported;

c) type and amount of date to be handled.

Static numerical requirements are sometimes identified in a separate, "Capacity" section.

Dynamic numerical requirements may include:

a) numbers of transactions and tasks;

b) amount of data to be processed within certain time periods for both normal and peak workload conditions.

Performance requirements should be stated in measurable terms:

"95 % of the transactions shall be processed in less than 1 second"

rather than

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

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

7.14 Logical database requirements

Specifies logical requirements for any data to go to a database, including:

a) data types used by various functions;

b) use frequency;

c) accessing capabilities;

d) data entities and their relationships;

e) integrity constraints;

f) data retention requirements.

7.15 Design constraints

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

7.16 Standards compliance

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

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 necessary for some applications to meet minimum regulatory or financial standards. An audit trace requirement may state that all changes to a payroll database should be recorded in a trace file with before and after values.

7.17 Software system attributes

Specifies attributes required from the software product, e.g.:

a) *reliability* – factors required to establish the required reliability of the software system at the time of delivery;

b) *availability* – factors required to guarantee defined availability level for the entire system such as checkpoint, recovery, and restart;

c) *security* – 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) use 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 program;
5) check data integrity for critical variables;
6) assure data privacy.

d) *Maintainability* – software attributes that provide easy maintenance. 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* – software attributes that provide easy porting of 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;
5) use of a particular operating system.

7.18 Verification

Provides verification approaches and methods planned to qualify the software. You should better organise verification data parallel to 7.10 – 7.17.

7.19 Supporting information

Provides additional supporting information including:

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

b) supporting or background information that can help SRS readers;

c) 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.

SRS should explicitly state whether or not these data are considered a part of the requirements.

References

1. International Organization for Standardization, 2011. *ISO/IEC/IEEE 29148-2011. Systems and software engineering – Life cycle processes – Requirements engineering*, Switzerland-USA: International Organization for Standardization, International Electrotechnical Commission, The Institute of Electrical and Electronic Engineers, Inc.
2. Institute of Electrical and Electronic Engineers, 1998. *IEEE 830-1998. IEEE Recommendation practice for software requirements specifications*, USA: The Institute of Electrical and Electronic Engineers, Inc.
3. Institute of Electrical and Electronic Engineers, 1998. *IEEE 1233-1998. IEEE Guide for developing system requirements specifications*, USA: The Institute of Electrical and Electronic Engineers, Inc.
4. Institute of Electrical and Electronic Engineers, 1998. *IEEE 1362-1998. IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document*, USA: The Institute of Electrical and Electronic Engineers, Inc.
5. International Organization for Standardization, 2008. *ISO/IEC/IEEE 12207-2008, Systems and software engineering – Software life cycle processes*, Switzerland-USA: International Organization for Standardization, International Electrotechnical Commission, The Institute of Electrical and Electronic Engineers, Inc.
6. International Organization for Standardization, 2008. *ISO/IEC/IEEE 15288:2008, Systems and software engineering – System life cycle processes*. Switzerland-USA: International Organization for Standardization, International Electrotechnical Commission, The Institute of Electrical and Electronic Engineers, Inc.
7. International Organization for Standardization, 2011. *ISO/IEC/IEEE 15289:2011, Systems and software engineering – Content of life-cycle information products (documentation)*, Switzerland-USA: International Organization for Standardization, International Electrotechnical Commission, The Institute of Electrical and Electronic Engineers, Inc.
8. Institute of Electrical and Electronic Engineers, 2006. *IEEE 1074-2006. IEEE Standard for Developing a software project life cycle process*, USA: The Institute of Electrical and Electronic Engineers, Inc.
9. International Organization for Standardization, 2006. *ISO/IEC 25051-2006. Software engineering – Software product Quality Requirements and*

Evaluation (SQuaRE) – Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing, Switzerland: International Organization for Standardization, International Electrotechnical Commission.

10. International Organization for Standardization, 2005. *15. ISO/IEC TR 19759-2005. Software Engineering – Guide to the Software Engineering Body of Knowledge (SWEBOK)*, Switzerland: International Organization for Standardization, International Electrotechnical Commission.
11. U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, 2001. *NUREG/CR-6734 Vol. 1. Digital Systems Software Requirements Guide-lines*. [pdf] Washington, DC: U.S. Nuclear Regulatory Commission. Available at: <http://pbadupws.nrc.gov/docs/ML0123/ML012330160.pdf> [Accessed 18 February 2014].
12. U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, 2001. *NUREG/CR-6734 Vol. 2 Digital Systems Software Requirements Failure Guidelines*. [pdf] Washington, DC: U.S. Nuclear Regulatory Commission. Available at: <http://pbadupws.nrc.gov/docs/ML0123/ML012330184.pdf> [Accessed 18 February 2014].
13. Программное обеспечение для компьютерных систем и сетей. Разработка технического задания [Текст]: учеб. пособие / В. И. Дужий, А. В. Волковой, А. А. Волковая и др.; Мин-во образования и науки Украины, Нац. аэрокосм. ун-т им. Н. Е. Жуковского “ХАИ”. – Нац. аэрокосм.ун-т им. Н. Е. Жуковского “ХАИ”, 2007. – 99 с.
14. Hull, E., Jackson, K., and Dick, J., 2004. *Requirements Engineering*. 2nd ed. USA Springer: London Berlin Heidelberg, 198 p.
15. Terzakis, John, 2010. *ICCGI Tutorial writing higher quality software requirements Version 1.0*. [pdf] Spain: ICCGI Conference. Available at: <http://www.iaria.org/conferences2010/filesICCGI10/ICCGI_Software_Requirements_Tutorial.pdf> [Accessed 18 February 2014].
16. Tuffley, D. *Software Requirements Specification (SRS)*. Available at: <http://www.software-requirements.tuffley.com/sample.pdf> [Accessed 18 February 2014].
17. Doe, J., 2011. *Recommended Practice for Software Requirements Specifications (IEEE)*. [pdf] Available at:

<http://midori.hu/downloads/jpdf/jira-software-requirement-specification.pdf> [Accessed 18 February 2014].

18. *Guidelines: Software Requirements Specification*. [online] Available at: <http://www.upedu.org/process/gdlines/md_srs.htm> [Accessed 18 February 2014].
19. *How to write a software requirements specification (SRS) document?* [online] Available at: <http://www.jaysonjc.com/p rogramming/how-to-write-a-software-requirements-specification-srs-document.html> [Accessed 18 February 2014].
20. Japenga, Robert. *How to write a software requirements specification*. [online] Available at: <http://www.microtoolsinc.com/Howsrs.php> [Accessed 18 February 2014].
21. Kumar, R., and Mann, N. *Introduction to Software Engineering*. [pdf] Available at: <http://www.ddegjust.ac.in/studymaterial/msc-cs/ms-12.pdf> [Accessed 15 November 2013].
22. Le Vie Jr, D. *Writing Software Requirements Specifications (SRS)*. [online] Available at: <http://techwhirl.com/writing-software-requirements-specifications/> [Accessed 18 February 2014].
23. Armitage, S. *Software Requirements Specification*. [pdf] Available at: <http://www4.informatik.tu-muenchen.de/proj/va/SRS.pdf> [Accessed 18 February 2014].

Appendix A

Software requirements specification outline

1 Introduction
- 1.1 Software name
- 1.2 Brief description of the application

2 Grounds for development
- 2.1 Base for development
- 2.2 Name and type of the subject of development

3 Purpose of development
- 3.1 Functional purpose
- 3.2 Operational purpose

4 Requirements for a program or software product
- 4.1 Functional requirements specification
  - 4.1.1 Requirements for composition of performed functions
  - 4.1.2 Requirements for organization of input data
  - 4.1.3 Requirements for organization of output data
  - 4.1.4 Timing requirements
- 4.2 Reliability requirements
  - 4.2.1 Measures ensuring reliable (resilient) operation of a program or software product
  - 4.2.2 Requirements ensuring reliable operation of a program or software product
  - 4.2.3 Recovery-after-failure time
  - 4.2.4 Failures caused by operator's incorrect actions
- 4.3 Operation conditions
  - 4.3.1 Climatic operation conditions
  - 4.3.2 Requirements for maintenance types
  - 4.3.3 Requirements for staff number and qualifications
- 4.4 Requirements for structure and parameters of technical equipment
- 4.5 Requirements for data-program or data-software product compatibility
  - 4.5.1 Requirements for data structures and solution methods
  - 4.5.2 Requirements to source codes and programming languages
  - 4.5.3 Requirements for software facilities applied by a program or software product
  - 4.5.4 Requirements for protection of data and programs

4.6 Requirements for marking and packing

Appendix B

Examples of software requirements specifications

1. Kalaidjian, A. *Software Requirements Specification. Elevator System Controller*. [pdf]
   Available at: <https://www.student.cs.uwaterloo.ca/~se463/Examples/SRS-Alex-Kalaidjian.pdf> [Accessed 18 February 2014].
2. Blanco, R. *Requirements Specification for an Elevator*. [pdf] Available at: <https://www.student.cs.uwaterloo.ca/~se463/Examples/SRS3-Rolando-Blanco.pdf> [Accessed 18 February 2014].
3. Langlois, J. *Software Requirements Specification for Stories Matter*. [pdf] Available at: <http://storytelling.concordia.ca/storiesmatter/wp-content/uploads/2012/11/software-requirements-specifications-stories-matter.pdf> [Accessed 18 February 2014].
4. Aasen, R., Cerauskis, L., Matson, B., Carlisle, E., Jeffers, E., Ritchey, J., Carson, N., Green, T., and Vo, P. *Software Requirements Specification for SplitPay Version 1.0 approved ZILDOR, Inc*. [pdf] Available at: <http://www.cise.ufl.edu/class/cen3031fa13/SRS_Example_1_2011.pdf> [Accessed 18 February 2014].
5. Batzilis, C. *Software Requirements Specification for PNotes, Requirements for Version 5.5*. [doc] Available at: <http://ftp.jaist.ac.jp/pub/sourceforge/p/pn/pnotes/PNotes/5.5.0/Software_Requirement_Specification-PNotes%205.5.doc> [Accessed 18 February 2014].
6. Despoudis, T. *Software Requirements Specification for <Project iTest>, Requirements for Version 1.3*. [doc] Available at: <http://itest.sourceforge.net/documentation/developer/Software_Requirements_Specification-iTest.pdf> [Accessed 18 February 2014].
7. *Comprehensive watershed management water use tracking project Software Requirements Specification*. [pdf] Available at: <http://open.sjrwmd.com/ep/docs/Elaboration/Software%20Requirements%20Specification.doc> [Accessed 18 February 2014].
8. Marvel Electronics and Home Entertainment. *E-Store Project Software Requirements Specification Version <4.0>*. [doc] Available at: <http://www.utdallas.edu/~chung/RE/Presentations07S/Team_1_Doc/Documents/SRS4.0.doc> [Accessed 18 February 2014].

9. Teamleader, J., Adams, P., Baker, B., and Charlie, C. *Software Requirements Specification Web Publishing System*. [doc] Available at: <http://www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc> [Accessed 18 February 2014].
10. Geagea, S., Zhang, S., Sahlin, N., Hasibi, F., Hameed, F., Rafiyan, E., and Ekberg, M. *Software Requirements Specification Amazing Lunch Indicator*. [pdf] Available at: <http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf> [Accessed 18 February 2014].

Appendix C

Examples of software requirements specification templates and instructions

1. David McKinnon, A. *Software Requirements Specification Template*. [doc] Available at: <http://www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc> [Accessed 18 February 2014].
2. J2Lite Software. *Requirements Specification for <Project Name>*. [pdf] Available at: <http://www.jaysonjc.com/dl/srs/J2Lite%20SRS%20Template.pdf> [Accessed 18 February 2014].
3. Ackerman F. [Product] *MTM Program Product Software Project Plan*. [docx] Available at: <http://cs.mtech.edu/main/images/standards/programProjects/mtmprogprodspptmplt.docx> [Accessed 18 February 2014].
4. Texas Project Delivery Framework. *Software Requirements Specification Template*. [doc] Available at: <http://www.dir.texas.gov/SiteCollectionDocuments/IT%20Leadership/Framework/Framework%20Extensions/SDLC/SDLC_softwareRequirements_template.doc> [Accessed 7 November 2013].
5. Texas Project Delivery Framework. *Software Requirements Specification Instruction*. [pdf] Available at: <http://www.dir.texas.gov/SiteCollectionDocuments/IT%20Leadership/Framework/Framework%20Extensions/SDLC/SDLC_softwareRequirements_instructions.pdf> [Accessed 7 November 2013].
6. *Software Requirements Specification*. [doc] Available at: <http://www.env.gov.bc.ca/csd/imb/3star/sdlc/3analysis/Software_Requirements_Specification_Standards.doc> [Accessed 7 November 2013].
7. Wiegers, K. E. *Software Requirements Specification for <Project>*. [doc] Available at: <http://www.cise.ufl.edu/class/cen3031fa13/SRS_TEMPLATE.doc> [Accessed 18 February 2014].
8. *[YourProject] Requirements Specification*. [online] Available at: <https://wiki.cac.washington.edu/download/attachments/4273783/Requirements%20Specification%20Template.doc?version=2&modificationDate=1203990391473&api=v2> [Accessed 18 February 2014].

9. *TEMPLATE – Software Requirements Specification (SRS)*. [online] Available at: <https://wiki.base22.com/download/attachments/3242/Template%20-%20Software%20Requirements%20Specification.docx?version=1&modificationDate=1294886541000&api=v> [Accessed 18 February 2014].
10. *Software Requirements Specification (SRS) Template*. [doc] Available at: <http://www.computing.dcu.ie/~renaat/CA326/s rs_template2.doc> [Accessed 7 November 2013].

Навчальне видання

Ілляшенко Олег Олександрович
Харченко В'ячеслав Сергійович

РОЗРОБЛЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ ДЛЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

Частина 1

(Англійською мовою)

Редактор Є. В. Пизіна
Технічний редактор Л. О. Кузьменко

Зв. план, 2014
Підписано до друку 17.04.2014
Формат 60x84 1/16. Папір офс. №2. Офс. друк
Ум. друк. арк. 2,2. Обл.-вид. арк. 2,5. Наклад 50 пр. Замовлення 152. Ціна вільна

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http:// www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції сер. ДК № 391 від 30.03.2001