D. Software Requirements Specification (SRS)

inputs and outputs of a system but which shows no decomposition.

External System Behaviour - those properties of a system that can be observed by a human being, for example, user interface presentation, speed and accuracy.

Natural Language - a language spoken normally by people, as opposed to a formal language with explicit rules for use such as a computer language or formal specification language.

Software Requirements Specification - a description of the required external behaviour of a software component in terms of functions, performance, design constraints, external interfaces and quality attributes.

System Requirements Specification - a statement of what the system must do from the user's point of view providing functional behaviours, performance, design constraints, external interfaces and quality attributes.

Structure Chart - a diagram that identifies modules, activities or other entities in a system or computer program and shows how larger or more general entities break down into smaller, more specific entities.

Taxonomy - a scheme that partitions a body of knowledge and defines the relationships among the pieces. It is used for classifying and understanding the body of knowledge.

Top-down Problem Decomposition - a requirements engineering concept that means going form abstract view to physical entity; from general description to detailed description; from a broad view to a detailed view via progressive partitioning of large problems into smaller component problems.

SRS - Software Requirements Specification.

D.1.Responsibilities

A full description of the responsibilities should be described in the project plan, but in general the following responsibilities apply:-

The project team is responsible for the following:-

- Producing the SRS according to the content specified in this standard.
- Consulting with the Customer in producing the SRS.
- Conducting Walkthroughs and an Inspection of the SRS.
- Gaining agreement with the Customer.
- The customer is responsible for the following:-
- Commit time and resources to participate in the development and review of the SRS with the project team

D.2.Introduction

The Software Requirement Specification is the major deliverable arising from the requirements capture process.

This process consists of four stages.

- **Problem analysis** - where an analyst team interviews the users to establish what problems are to be solved without defining how they will be solved (Requirements List).
- **External product description** - where analysts and users describe the external behaviour of the product without defining how it will work internally. This occurs in either the business or the operations environment.
- **Conceptual design** - where the major functional and/or physical components of the system are identified.
- **Software Requirements Specification** - where external behaviour of the software components of a system are described without regard to how they work internally.

In all but the smallest of systems, requirements engineering is an iterative process involving requirements definition followed by conceptual design (component definition), followed by further requirements definition for each component. The end result being a hierarchy of components with associated requirements specifications. The specifications may be contained in a single document or a set of documents (one for each component). This process simplifies problem definition by progressively partitioning a problem into its component subproblems. Each smaller subproblem can then be analysed in isolation.

D.3.Characteristics of an SRS

The basic feature of an SRS is that it should specify the results to be achieved by the software, not the means by which those results are obtained. Specify the functionality, performance and design constraints without embedding any design specifications.

This section provides desirable properties of an SRS. The absence of any of the properties described below may be viewed as a defect in the context of an SRS document inspection.

D.3.1. Correct

The degree to which a documented statement of requirement adheres to stated facts. For example, if the customer states that the system is required to detect a 20 millisecond pulse and the SRS states 'the data acquisition system shall be capable of detecting a 30 millisecond pulse' - the requirement is incorrect because it embeds a design statement - “the data acquisition system”.

Note: The correctness of an SRS can only be improved by a thorough review with the customer.

D.3.2. Unambiguous

Each requirement statement has only *one* semantic interpretation. As a minimum, each characteristic of the final product shall be described in a single unique term.

In practise this means that if ten people were given the same sentence they would interpret it in the same way. If a natural language description introduces ambiguity, simple requirements modelling tools may be used.

D.3.3. Complete

The degree to which the SRS records *all* of the desired behaviours of the system. More specifically, a SRS is complete if it thoroughly addresses the following four issues:

- **Functions** - everything that the software is supposed to do is included in the SRS. Omissions may be detected by thorough review of the SRS document with the customer.

- **Stimulus/response -** definitions of all responses of the software to all realisable classes of input data under all realisable classes of situations should be included. **Note:** It is important to specify the responses to both valid and invalid inputs. Responses to inputs must also be viewed in the context of various system states.

- **Document housekeeping -** all pages are numbered and identified as per the standard, all figures and tables are numbered named and referenced, all terms are defined, all units of measure are provided and all referenced material and sections are present.

- **TBDs -** no sections are marked 'to be determined (TBD)'. If the insertion of a TBD is unavoidable it should be accompanied by a notation of who is responsible for its resolution and by when.

D.3.4. Verifiable

A requirement is verifiable if there exists some finite cost-effective process with which a person or a machine can check that the actual as-built software product meets the requirement. In practical terms this means that statements of requirement must be specific, unambiguous and quantitative. For example.

- **Nonverifiable** - the system shall respond to user queries in a timely manner.

- **Verifiable** - the system shall respond to a user request for an account balance in five seconds or less under operating conditions.

In general SRS authors should avoid the use of subjective adjectives such as **suitable**, **appropriate** and **timely**.

D.3.5. Consistent

An SRS is consistent if there is no conflict between subsets of requirements.

Examples of inconsistencies include the following:

- **Conflicting terms** - two terms are used in different contexts to mean the same thing. (i.e. speed and velocity).
- **Conflicting characteristics** - two parts of the SRS demand that the product exhibit contradictory traits.

D.3.6. Modifiable

An SRS is modifiable if its structure and style are such that any necessary changes can be made easily, completely and consistently.

Modifiability involves the following:

- **Organisation -** coherent, logical organisation, including a table of contents and cross references.
- **No redundancy** - the same requirement should not appear in more than one place.

D.3.7. Traceable

An SRS is traceable if the origin of each of its requirements is clear (backward traceability) and if it allows the referencing of each requirement in future documentation (forward traceability).

Both classes of traceability shall be provided as follows:

- **Forward** - this allows the designer to demonstrate that he/she has produced a design that satisfies all the customer's requirements. To this end all statements of requirement shall be named and identified with a paragraph number such that they may be referenced in future requirements and design documents.
- **Backward** - this demonstrates that the SRS author knows why every requirement in the SRS exists. Each statement of requirement shall explicitly reference its source in previous documents. If a document does not exist or is not available the individual who stated the requirement shall be identified.

When a requirement is an apportionment or a derivative of another requirement, both forward and backward traceability should be provided. Examples include the following:

- The allocation of a response time to a database function from the overall user response time requirement.
- The identification of a report format with certain functional and user interface requirements.
- A software product that supports legislative or administrative needs (i.e. passenger safety). In this case, the exact legal requirement should be specified.

Forward traceability is particularly important when the software product enters the operations and maintenance phase. As code and design documents are modified, it is essential to be able to determine the complete set of requirements that may be affected by those modifications.

D.3.8. Usable during the operations & maintenance phase

The SRS must address the needs of the operations and maintenance phase, including the eventual replacement of the software.

Maintenance is often done by someone other than the original developer. Local changes can be implemented by means of well-commented code. For changes with a broader scope, however, the design and requirements documentation is essential.

D.3.9. Standards compliant

The SRS must address each issue specified in this standard. In practical terms this means the following:

- All specifications shall have the paragraph headings specified below.
- If a paragraph heading is not applicable to the user's needs it is to be included and marked not applicable.
- All applicable paragraphs shall address the checklist of key points covered in each subsection of this standard.

D.4.Level of abstraction

D.4.1. Problem partitioning

A common problem faced by the requirements engineer is establishing the level of detail, and consequently the level of abstraction, to be provided in an SRS. Requirements engineers are often hampered by customers who wish to supply either too little or too much detail. Where too little detail is forthcoming, there are complaints that "we are designing the system for you" while the customer who wants to provide too much detail is in effect wanting to be involved in the design process.

The level of detail to be presented to the designer must be consciously set in the context of each requirements specification exercise. As the level of detail increases the SRS becomes less abstract (i.e. the level of abstraction decreases). This amounts to deciding where the external behaviour of the software will be constrained by the customer and where it will be left to the imagination of the designer.

The level of detail may be formally quantified by viewing the problem as a hierarchy of interrelated sub-problems that may be represented as a bill of material family tree. The most abstract statement of the problem is represented at Level 1 or root of the tree and progressively decomposed into subproblems.

D.4.2. Issues constraining level of abstraction

The point at which requirements end and design begins is still a hotly debated subject. This is because it will necessarily vary in the context of each requirements definition exercise.

What criteria can be used to determine an acceptable level of abstraction for a particular SRS?

The following issues provide guidelines:

- **Meeting business objectives -** if the omission of certain required behaviour will cause a system to fail in a business sense, the behavioural description must be included. For example, if a securities house bases its buy/sell decisions on the calculation of a share index, the algorithm for calculation of the index must be provided in detail.

- **Designability** - the designer should be able to take the statement of requirement and produce a design with minimal contact with the customer.

- **Preferred modes of operation** - the user may expect the system to behave in a certain way. In this case the description of user interface shall fall within the scope of requirements definition. For example, an air traffic controller is vitally interested in the modes used by the system to present aircraft coordinates, speed and direction. This will extend to the definition, in the SRS, of exact screen formats, modes of operation, refresh frequencies, data accuracy, data display precision and key stroke sequences. At the other extreme, in the definition of an accounts payable system, the accountant will probably not be concerned with the order in which the cursor moves from one screen data window to another.

- **External system constraints** - if the target system is to become a component of a larger existing system its external behaviour may be heavily constrained by that system. Details of interface with the parent system shall therefore be included in the scope of the SRS. For example, the requirements of a telephone hand set will be constrained by the user's expectation of telephone operation and the standard mode of interface with the telephone network. In this instance a user interface description and a telephone network interface protocol description would be a minimum requirement.

- **Business/legal constraints** - if system behaviour is constrained by externally imposed business or legal rules, procedures and work instructions they shall be referenced or described in detail in the SRS. For example, in the context of a telephone exchange operation, a telephone company policy may state that "no employee shall have the ability to manually adjust a customer’s telephone meter". With this in mind a Customer Exchange Requirements Specification would state.

- The system shall increment the calling party's meter when the called party hand set is taken off hook.

- The meter value shall not be modified by any system function other than that described in Requirement 1.

- **Equitable tender evaluation** - if the SRS is to form the basis of a request for tender, sufficient detail must be provided to allow for a fair and reasonable comparison of the cost/performance of each response. If the scope of the tender is design/implement/test/install, all major functional descriptions, data, data flows and capacity and performance requirements must be provided.

- **Touch it, see it, smell it** - in the context of requirements definition the computer literate requirements engineer typically progresses from real world abstract concepts to the description of physical objects to the definition of computer related logical concepts. Systems engineers tend to drift toward their field of expertise allowing preconceived design approaches to creep into the SRS. As a general guideline, requirements stop the behavioural description of physical objects that we can touch, see and smell. For example we can touch a telephone hand set, we can hear a dial tone and we can see a calling line identification number. We have no physical senses to detect an index, a message terminator or a data structure.

D.5.Content

This section outlines the minimum content of the SRS.

The SRS contains three broad sections. Sections 1 & 2 provide introductory and background information to the target system. Section 3 provides all the details that the software developer needs in order to create the design and is the largest and most important part of the SRS. Section 3 describes the system in terms of the behaviour of its component parts.

While all the generic types of requirements described in the following sections must be addressed, the structure of the document will be determined by whether a specific type of requirement applies to an individual component or to the system as a whole. This makes possible a variety of organisational schemes (see section 2.4.5 - for detail).

Issues relevant to particular systems that are not provided by this outline may be included as required by the Analyst and approved by the Project Manager.

If a section is not relevant to the target system it must be included and marked 'not applicable' in the body of the text.

In general terms, the behaviour of each component requirement must be described in the following terms:

- Introduction.
- Input.
- Processing.
- Output.

Either individual components or the complete system must also be described as follows:

- External interface requirements.

- Performance requirements.
- Design constraints.
- Security requirements.
- Maintainability requirements.
- Reliability requirements.
- Availability requirements.
- Database requirements.
- Documentation requirements.
- Safety requirements.
- Operational requirements.
- Site adaptation.

The sections which follow indicate the content of each component of the SRS.

D.5.1. Table of contents

The following table indicates the document outline of the SRS. The arrangement of section 3 can vary according to the nature of the software being specified. Examples of variations are given in section 2.4.5 Organising the SRS.

```
Table of Contents
1.  Introduction
    1.1  Purpose
    1.2  Scope
    1.3  Definitions, acronyms & abbreviations
    1.4  References
    1.5  Document overview

2.  General description
    2.1  Product perspective
    2.2  Product functions
    2.3  User characteristics
    2.4  General constraints
    2.5  Assumptions & dependencies

3.  Specific requirements
    3.1  Functional requirements
         3.1.1  Functional requirement 1
                3.1.1.1  Introduction
                3.1.1.2  Input
                3.1.1.3  Processing
                3.1.1.4  Output
         3.1.n  Functional requirement n
    3.2  External interface requirements
    3.3  Performance requirements
    3.4  Design constraints
    3.5  Security requirements
    3.6  Maintainability requirements
    3.7  Reliability requirements
    3.8  Availability requirements
    3.9  Database requirements
    3.10 Documentation requirements
    3.11 Safety requirements
    3.12 Operational requirements
    3.13 Site adaptation

Index
```

Figure 3 - SRS Table of Contents

D.5.2. Introduction (section 1)

The purpose of this section of the SRS is to provide an overview of the purpose and scope of the system.

D.5.2.1. Purpose (section 1.1)

Include the following:

- Identify the system name.
- Describe the purpose of the SRS.
- Identify the intended audience for the document.
- Identify the customer and the project team.

D.5.2.2. Scope (section 1.2)

Include the following:

- List the software products described by the SRS.
- Describe what the products will and if necessary won't do.
- Describe the application of the software being specified, including benefits, objectives and goals.

D.5.2.3. Definitions, acronyms and abbreviations (section 1.3)

Include the following:

- Describe terms used that cannot be expected to be common knowledge within the document's target audience.
- Exclude data entity and attribute names that are to be included in data dictionaries.

D.5.2.4. References (section 1.4)

Include the following:

- Identify the preceding documents from which this document was created.

- Identify all documents referenced in this document specifying reference code (to be used in the body of the text), document name, author, version number, release date and location.
- Identify the sources from which the references may be obtained (if applicable).

D.5.2.5. Document overview (section 1.5)

Provide an overview of the remainder of the document describing, in brief, the purpose and content of each level one paragraph.

For example.
Section 2 provides a general description of user requirements.
Section 3 provides specific details of user requirements.

D.5.3. General description (section 2)

This section of the SRS describes the general factors that affect the product and its requirements. It does not state specific requirements, it only makes those requirements easier to understand.

D.5.3.1. Product perspective (section 2.1)

Include the following points.

- Describe (in overview) the business process that the system will support.
- State if the product is self contained (stand alone) or a component of a larger system.
- If the product is a component of a larger system describe its principal functions.
- Identify principal external interfaces.
- Provide an overview of computer hardware to be used if already determined.
- Provide a block diagram describing major components and interfaces.

This subsection should not prescribe specific design solutions or design constraints on the solution. Rather it needs to explain why certain design constraints are specified in section 3.

D.5.3.2. Product functions (section 2.2)

This subsection provides a summary of the functions that the software will perform. It will discuss major functional areas without mentioning any of the large amount of detail that goes with those areas. The functions need to be organised so that the customer will understand them.

Include the following points:

- A summary of functions to be performed.
- A functional model (i.e. data flow diagram) if appropriate.
- Reasons why specific requirements are stated in Section 3 (i.e. relate the functions to the business processes they support).

Where applicable, the function summary for this subsection can be taken directly from the higher-level specification.

This subsection should not prescribe specific design solutions or design constraints on the solution. Rather it needs to explain why certain design constraints are specified in section 3.

D.5.3.3. User characteristics (section 2.3)

This subsection describes the general characteristics of the users that will influence the SRS.

Describe intended system users in terms of the following:

- Educational level.
- Experience.
- Technical expertise.

Describe the following:

- The operational environment (i.e. physical location, temperature, humidity, vibration, electrical noise).
- Frequency of use (i.e. occasional use, constant use). Provide reasons why specific user-related requirements or design constraints are stated in section 3.

D.5.3.4. General constraints (section 2.4)

This subsection gives a general description of any other items that will limit the developer’s design options.

Define limiting factors in the creation of a design.

- Regulatory policies.
- Hardware/system software resource limitations (i.e. memory, disk space).
- External interfaces.
- Audit requirements.
- Company financial control policy.
- Communications protocols.
- Application criticality.
- Safety, security.
- Operations (i.e. unattended operation).

Provide general descriptions only. Further detail specific to the requirements impacted shall be provided in section 3.

D.5.3.5. Assumptions & dependencies (section 2.5)

Define assumed factors that would cause the SRS to change should the assumption be incorrect.

The successful operation of the system is dependant upon the following assumptions and dependencies. Acceptance of this specification means acceptance of the risks associated with these issues.

For example.

- Hardware availability.
- Behaviour of interfacing external systems
- Availability of experienced operations staff.

These assumptions and dependencies are verified individually with the customer. This verification is important in that it helps to arrive at a clear understanding between the developer and the customer which will, in turn, avoid potential contract disputes later.

D.5.4. Specific requirements (section 3)

This section of the SRS provides comprehensive detail on all requirements. It should include all of the detail which the designer will need to create the design. The details shown in section 3 should be defined as individual specific requirements,