Software Requirements Specification (SRS) Tailoring Standards

The following description provides the basic document organization, and required information (see the *Document Style Guidelines and Standards* for information about format). The main thing to remember is that the SRS must contain a Requirements Traceability Matrix (RTM). Each *requirement* is uniquely identified (using a number) in the body of the document and must be called out in the RTM. Each requirement will use the verb "shall." Otherwise the statement shall not be considered a requirement. Each item below should be considered required as a major element or section in the SRS. The standard referred to here is IEEE Std. 830-1993.

Title Page

Abstract

Frontmatter

1. Introduction (see section 5.1 of the standard for details)

1.1. Purpose

1.2. Scope

1.3. Definitions, acronyms, and abbreviations

1.4. References

1.5. Overview

2. Overall description (see section 5.2 of the standard for details)

2.1. Product perspective

2.2. Product functions

2.3. User characteristics

2.4. Constraints

2.5. Assumptions and dependencies

3. Specific Requirements (see section 5.3 and/or appendix A of the standard for details)

Appendixes

Requirements Traceability Matrix

Index (optional)

Software Requirements Specification (IEEE Std 830-19933): Review ¶ 4.1 about the nature of an SRS and the following subparagraphs that define characteristics of a good SRS. In ¶ 5, the parts of the SRS are described. This is the important part that gives what should be included content-wise. See the appendix for templates on structure organization (A.7 is recommended but not required).

Table 1 Example Requirements Traceability Matrix (filled in with DFD Identifiers).

Req. ID System Level.	**Req. ID** Sub-system Level.	**DFD Identifier(s)** (fill during design)	**Module Name(s)** (fill during implementation)	**Verification Method**	**Tested**
A0001		1	ValidateAccess	..	
	A01.10	1.1	Read, Details	A, D	
	A01.20	1.2	CheckDate, Report	D	
	A01.30	1.3	ValidatePIN	I	
A0002		2	...	...	
	A02.10	2.1	...	...	
		2.2	...	...	
	A02.20	...	...	...	
	...	...	...	...	
			...	...	
A0003	A03.10	3	...	...	
			...	...	
A0004		4	...	...	
	A04.10	4.1	...	...	

KEY: A = by Analysis, I = by Inspection, D = by Demonstration and An = by Analogy

Table 1 shows an example of how to construct the RTM (Requirements Traceability Matrix). Notice the hierarchy of system level requirements and their corresponding component level requirements. The component, labeled subsystem level requirements, are said to be traceable to the system level which are their parent requirements. The subsystem level requirements are so-called *derived requirements*. As an example, Figures 1 and 2 show the overall hierarchy of the physical system design based on the SSA/SD design methodology.

The RTM provides one place where all requirements are listed, where they are located within the program (i.e., software product), and finally how each is to be verified. A column is provided to mark if each of the requirements has been tested as a matter of tracking your progress and for use in the *demonstration*.

Finally, lets consider how you will use the various verification methods to show that a particular requirement has been satisfied. A= Analysis - used to show that nonfunctional requirements have been met like response time. I = Inspection - used to show that something like using specific coding standards has been adhered to (I.e., lets look at all the code and verify that each module has the requisite preamble). D= Demonstration - used to show that by say running the program that it computes the correct output (in the required amount of time perhaps in the case of real-time systems). K= Analogy - used when all else fails. This method in some reasonably rigorous fashion (up to the customer how rigorous) shows or proves that a particular requirement(s) is met through some indirect means (e.g., if the program

displays the correct symbology then we must know that some other requirement that cannot be reasonably isolated, has also been satisfied).

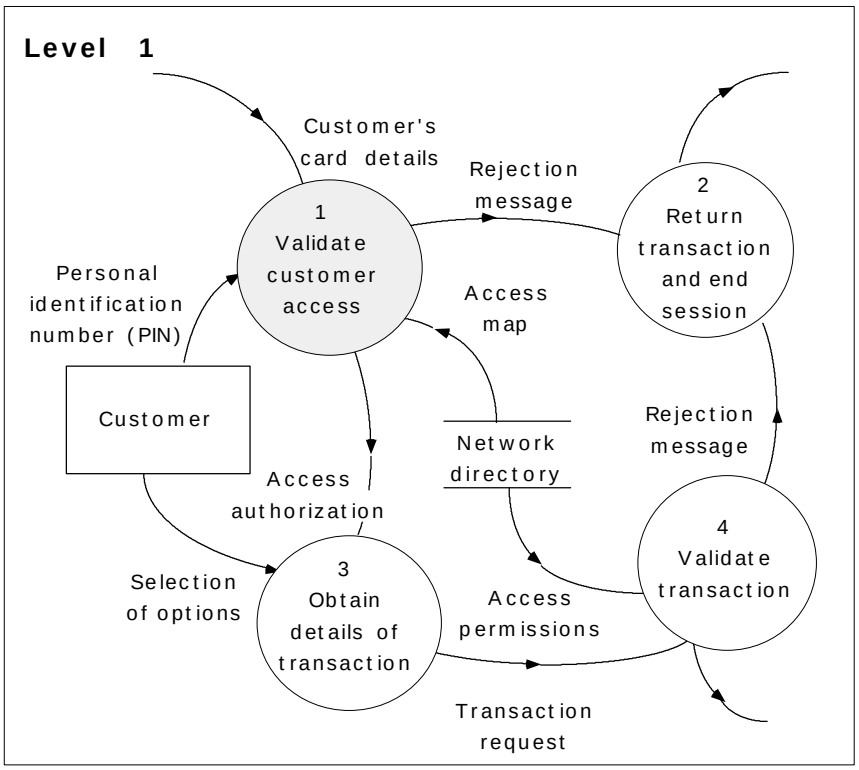

Figure 1. Top-level Data Flow Diagram (DFD).

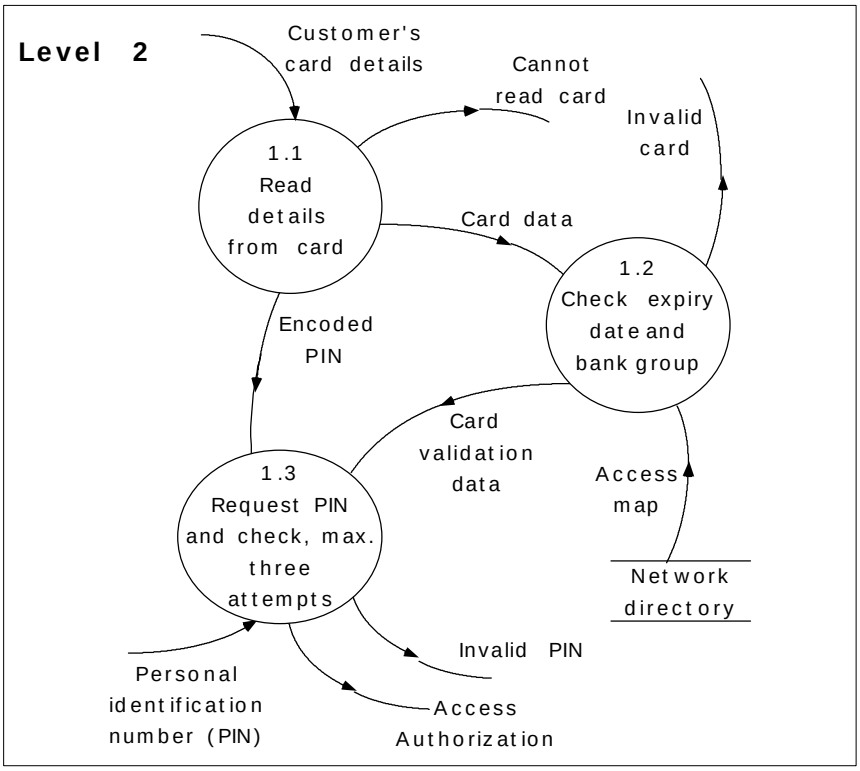

Figure 2. Second-level Data Flow Diagram (DFD).