



STANDARD IEEE-STD-830-1998

IMPLEMENTACIÓN Y EJEMPLO DE LAS ESPECIFICACIONES DE REQUISITOS DE SOFTWARE

Septiembre 2017

TABLA DE CONTENIDO

INTRODUCCIÓN

1. DEFINICIONES PRELIMINARES

- 1.1. Contrato
- 1.2. Cliente
- 1.3. Proveedor
- 1.4. Usuario

2. LAS CONSIDERACIONES PARA PRODUCIR UN BUEN SRS.

- 2.1. La Naturaleza del SRS
- 2.2. El Ambiente del SRS
- 2.3. Las Características de un buen SRS
 - 2.3.1. Correcto
 - 2.3.2. Inequívoco
 - 2.3.2.1. Trampas del idioma natural
 - 2.3.2.2. Idiomas de especificación de requisitos
 - 2.3.2.3. Representación hecha con herramientas
 - 2.3.3. Completo
 - 2.3.3.1. Representación hecha con herramientas
 - 2.3.4. Consistente
 - 2.3.4.1. Consistencia interior
 - 2.3.5. Delinear que tiene importancia y/o estabilidad
 - 2.3.5.1. Grado de estabilidad
 - 2.3.5.2. Grado de necesidad
 - 2.3.6. Comprobable
 - 2.3.7. Modificable
 - 2.3.8. Identificable
- 2.4. La preparación de los Joins del SRS
- 2.5. La evolución de SRS
- 2.6. Prototipos
- 2.7. Generando el diseño en el SRS
 - 2.7.1. Requisitos del plan necesarios
- 2.8. Generando los requisitos del proyecto en el SRS.

3. LAS PARTES DE UN SRS

- 3.1. Introducción
 - 3.1.1. Propósito
 - 3.1.2. Alcance
 - 3.1.3. Definiciones, siglas, y abreviaciones
 - 3.1.4. Referencias
 - 3.1.5. Apreciación global
- 3.2. Descripción global
 - 3.2.1. Perspectiva del producto
 - 3.2.1.1. Las interfases del Sistema
 - 3.2.1.2. Las interfases del Usuario
 - 3.2.1.3. Las interfases del Hardware
 - 3.2.1.4. Las interfases del Software

- 3.2.1.5. Las interfases de Comunicaciones
- 3.2.1.6. La Memoria
- 3.2.1.7. Los Funcionamientos
- 3.2.1.8. Los requisitos de adaptación del Site
- 3.2.2. Funciones del producto
- 3.2.3. Características del usuario
- 3.2.4. Restricciones
- 3.2.5. Atención y dependencias
- 3.3. Los requisitos específicos
- 3.3.1. Interfases externas
- 3.3.2. Funciones
- 3.3.3. Requisitos del desarrollo
- 3.3.4. Requisitos del banco de datos lógicos
- 3.3.5. Restricciones del diseño
- 3.3.5.1. Aceptación de las normas
- 3.3.6. Atributos del software del sistema
- 3.3.6.1. Fiabilidad
- 3.3.6.2. Disponibilidad
- 3.3.6.3. Seguridad
- 3.3.6.4. Mantenimiento
- 3.3.6.5. Portabilidad
- 3.3.7. Organizar los requisitos específicos
- 3.3.7.1. Modo del sistema
- 3.3.7.2. Clases de usuario
- 3.3.7.3. Objetos
- 3.3.7.4. Rasgo
- 3.3.7.5. Estímulo
- 3.3.7.6. Contestación
- 3.3.7.7. Jerarquía Funcional
- 3.3.8. Comentarios adicionales
- 3.4. Información de apoyo
- 3.4.1. Tabla de contenidos e índice
- 3.4.2. Apéndices

4. ANEXOS

- 4.1. Formato de especificación de requisitos para un proyecto de software
- 4.2. Referencias y normas recomendadas para usar simultáneamente con la IEEE-830:1998
- 4.3. Pautas para la conformidad de la norma IEEE 830:1998 con IEEE/EIA 12207.1-1997

BIBLIOGRAFÍA

INTRODUCCIÓN

Para llevar a cabo un proyecto de realización de software, se deben aplicar principios de la ingeniería de requerimientos. Esta área, comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requerimientos de los inversores, que pueden entrar en conflicto entre ellos. Puede ser conocida también como "Análisis de requerimientos", "especificación de requerimientos", etc.

El propósito de la ingeniería de requerimientos es hacer que los mismos, alcancen un estado óptimo antes de seguir adelante con el proyecto. Los buenos requerimientos deben ser medibles, comprobables, sin ambigüedades o contradicciones, etc.

Para su práctica se siguen los pasos nombrados a continuación:

1. Fases de implementación
2. Técnicas principales
 - 2.1. Entrevistas
 - 2.2. Talleres
 - 2.3. Forma de contrato
 - 2.4. Objetivos mensurables
 - 2.5. Prototipos
 - 2.6. Casos de uso
3. Especificación de requisitos del software
4. Identificación de las personas involucradas
5. Problemas
 - 5.1. Relacionados con las personas involucradas
 - 5.2. Relacionados con los analistas
 - 5.3. Relacionados con los desarrolladores
 - 5.4. Soluciones aplicadas
6. Fuentes

La fase de la especificación de requisitos del software es una descripción completa del comportamiento del sistema a desarrollar. Incluye un conjunto de casos de uso que describen todas las interacciones que se preveen que los usuarios tendrán con el software. También contiene requisitos no funcionales (o suplementarios). Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño).

Las estrategias recomendadas para la especificación de los requisitos del software están descritas por la IEEE 830-1998. Este estándar describe las estructuras posibles, contenido deseable y calidades de una especificación de requisitos del software.

A continuación se realiza un análisis de la norma IEEE 830-1998, entregando una base amplia para una futura implementación en un programa determinado de software.

1. DEFINICIONES PRELIMINARES

En general las definiciones de los términos usados en estas especificaciones están conforme a las definiciones proporcionadas en IEEE Std 610.12-1990.

1.1. Contrato:

Un documento es legalmente obligatorio y en el estarán de acuerdo las partes del cliente y proveedor. Esto incluye los requisitos técnicos y requerimientos de la organización, costo y tiempo para un producto. Un contrato también puede contener la información informal pero útil como los compromisos o expectativas de las partes involucradas.

1.2. Cliente:

La persona(s) que pagan por el producto y normalmente (pero no necesariamente) definen los requisitos. En la práctica el cliente y el proveedor pueden ser miembros de la misma organización.

1.3. Proveedor:

La persona (s) que producen un producto para un cliente.

1.4. Usuario:

La persona (s) que operan o actúan recíprocamente directamente con el producto. El usuario (s) y el cliente (s) no es (son) a menudo las mismas persona(s).

2. CONSIDERACIONES PARA PRODUCIR UN BUEN SRS

Estas cláusulas proporcionan información a fondo que deben ser consideradas al momento de producir un SRS (software requirements specification). Esto incluye lo siguiente:

2.1. Naturaleza del SRS

El SRS son especificaciones para un producto del software en particular, programa o juego de programas que realizan ciertas funciones en un ambiente específico. El SRS puede escribirse por uno o más representantes del proveedor, uno o más representantes del cliente o por ambos.

Los problemas básicos que se presentan al escribir un SRS van dirigidos a lo siguiente:

| Funcionalidad | ¿Qué se supone va hacer el software? |
|--|--|
| Las interfases Externas. | ¿Cómo el software actúa recíprocamente con las personas, el hardware de los sistemas, otro hardware, y otro software? |
| La Actuación. | ¿Cuál es la velocidad, la disponibilidad, tiempo de la contestación, tiempo de la recuperación de varias funciones del software, etc.? |
| Los Atributos. | ¿Qué portabilidad tiene, exactitud, el mantenimiento, la seguridad, las consideraciones, etc.? |
| Las restricciones del diseño que impusieron en una aplicación. | ¿Hay algún requerimiento Standard, idioma de aplicación, las políticas para la integridad del banco de datos, los límites de los recursos, operando en qué ambiente (s), etc.? |

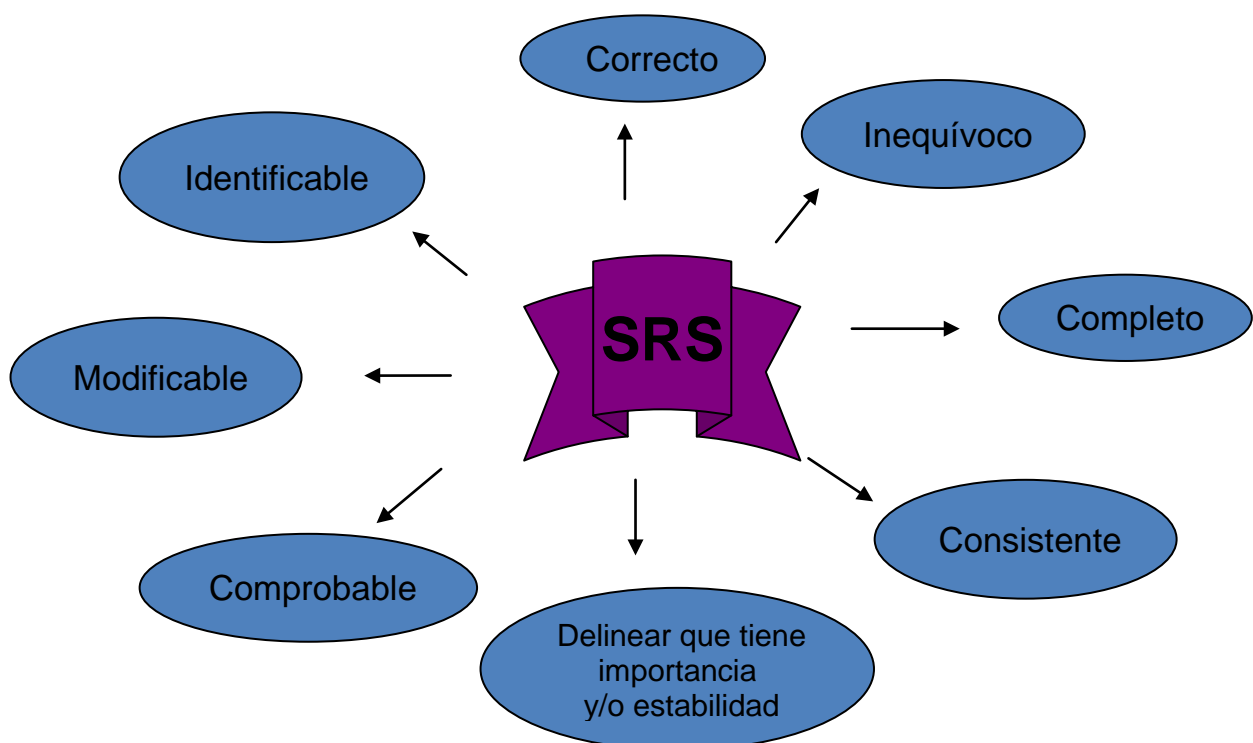
2.2. Ambiente del SRS

El software puede contener toda la funcionalidad del proyecto esencialmente o puede ser parte de un sistema más grande. En el último caso habrá un SRS que declarará las interfases entre el sistema y su software modular y pondrá qué función externa y requisitos de funcionalidad tiene con el software modular.

Desde que el SRS tiene un papel específico en el proceso de desarrollo de software, el que lo define, debe tener el cuidado para no ir más allá de los límites de ese papel. Esto significa que:

- Debe definir todos los requisitos del software correctamente. Un requisito del software puede existir debido a la naturaleza de la tarea a ser resuelta o debido a una característica especial del proyecto.
- No debe describir cualquier plan o detalles de aplicación. Éstos deben describirse en la fase del diseño del proyecto.
- No debe imponer las restricciones adicionales en el software. Éstos se especifican propiamente en otros documentos.

2.3 Características de un buen SRS



2.3.1. Correcto: Un SRS es correcto si y sólo si, cada requisito declarado se encuentra en el software. No hay ninguna herramienta o procedimiento que aseguran la exactitud.

Alternativamente el cliente o el usuario pueden determinar si el SRS refleja las necesidades reales correctamente, identificando los requerimientos; esto hace el procedimiento más fácil y con menor probabilidad de error.

2.3.2. Inequívoco: Un SRS es inequívoco si y sólo si, cada requisito declarado tiene sólo una interpretación. Como un mínimo, se requiere que cada característica de la última versión del producto se describa usando un único término. En casos donde un término en un contexto particular tenga significados múltiples, el término debe ser incluido en un glosario donde su significado sea más específico. El SRS debe ser inequívoco para aquéllos que lo crean y para aquéllos que lo usan. Sin embargo, estos grupos no tienen a menudo el mismo fondo y por consiguiente, no tienden a describir los requisitos del software de la misma manera.

2.3.2.1. Trampas del idioma natural: Los requisitos son a menudo escritos en el idioma natural (por ejemplo, inglés); este idioma es inherentemente ambiguo. Un SRS podría ser revisado por una parte independiente, para identificar el uso ambiguo del idioma para que pueda corregirse.

2.3.2.2. Idiomas de especificación de requisitos: Una manera de evitar la ambigüedad inherente en el idioma natural es escribir el SRS en un idioma de especificación de requisitos particular. Sus procesadores del idioma descubren muchos errores léxicos, sintácticos y semánticos automáticamente. Una desventaja en el uso de tales idiomas, es que la falta de tiempo exige aprenderlos. También, muchos usuarios no-técnicos los encuentran ininteligibles. Es más, estos idiomas tienden a ser buenos a expresar ciertos tipos de requisitos y dirigirse a ciertos tipos de sistemas. Así, ellos pueden influir en los requisitos de las maneras sutiles.

2.3.2.3. Representación hecha con herramientas: En general, los métodos de requisitos e idiomas y las herramientas que los apoyan, se clasifican en tres categorías generales:

| | |
|------------|--|
| Objeto | Organizan los requisitos en lo que se refiere a los objetos en el mundo real, sus atributos, y los servicios realizados por esos objetos. |
| Procesos | Organizan los requisitos en las jerarquías de funciones que comunican vía el flujo de datos |
| Conductual | Describen conducta externa del sistema por lo que se refiere a alguna noción de lo abstracto, las funciones matemáticas o el estado de las máquinas. |

El grado en que se usan estas herramientas y los métodos pueden ser útiles preparando un SRS pero depende del tamaño y complejidad del programa. Aún usando cualquiera de estos términos es mejor retener las descripciones del idioma natural.

2.3.3. Completo: Un SRS está completo si y sólo si, incluye:

- Los requisitos están relacionados a la funcionalidad, el desarrollo, las restricciones del diseño, los atributos y las interfases externas. En particular debe reconocerse cualquier requisito externo impuesto por una especificación del sistema y si lo requiere, debe tratarse.

- La definición de las respuestas del software a todos los posibles datos de la entrada del sistema y a toda clase de situaciones. Una nota que es importante especificar son las contestaciones a las entradas válidas e inválidas a ciertos valores.
- Tener todas las etiquetas llenas y referencias a todas las figuras, tablas, diagramas en el SRS y definición de todas las condiciones y unidades de medida.

2.3.3.1. Uso de TBDs: Cualquier SRS que usa la frase "para ser determinado" (to be determined -TBD) no es un SRS completo. El TBD es sin embargo, ocasionalmente necesario y debe acompañarse por:

- Una descripción de las condiciones que causan el TBD (por ejemplo, por qué una respuesta no es conocida) para que la situación pueda resolverse.
- Una descripción de lo que debe hacerse para eliminar el TBD, quien es el responsable de su eliminación y como debe hacerlo.

2.3.4. Consistente: La consistencia se refiere a la consistencia interior. Si un SRS no está de acuerdo con algún documento de nivel superior, como una especificación de requisitos del sistema, entonces no es correcto.

2.3.4.1. Consistencia interior: Un SRS es internamente consistente si y sólo si, ningún subconjunto de requisitos individuales generó conflicto en él.

Los tres tipos de conflictos probables en un SRS son:

| | |
|---|---|
| Las características especificadas en el mundo real de los objetos pueden chocar. | El de un informe del rendimiento puede describirse en un requisito como tabular pero en otro como textual |
| | Un requisito puede declarar que todas las luces serán verdes mientras otro puede declarar que todas las luces sean azules. |
| Puede haber conflicto lógico o temporal entre dos acciones especificadas. | Un requisito puede especificar que el programa sumará dos entradas y otro puede especificar que el programa los multiplicará. |
| | Un requisito puede declarar que "A" siempre debe seguir "B", mientras otro puede requerir que "A" y "B" ocurran simultáneamente. |
| Dos o más requisitos pueden describir el mismo mundo real del objeto pero uso las condiciones diferentes para ese objeto. | Una demanda del programa para una entrada del usuario puede llamarse una "sugerencia" en un requisito y una "señal" en otro. El uso de terminología normal y definiciones promueve la consistencia. |

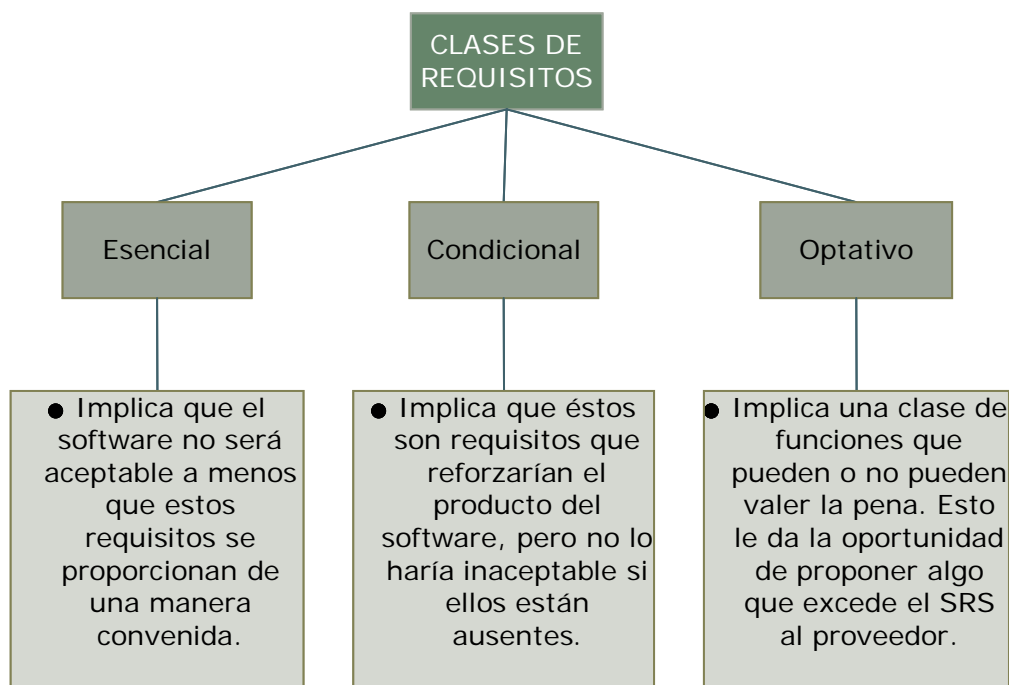
2.3.5. Delinear que tiene importancia y/o estabilidad: Un SRS debe delinear la importancia y/o estabilidad, si cada requisito en él tiene un identificador para indicar la importancia o estabilidad de ese requisito en particular. Típicamente, todos los requisitos que relacionan a un producto del software no son igualmente importantes. Algunos requisitos pueden ser esenciales, sobre todo para las aplicaciones de vida crítica, mientras otros pueden ser deseables. Cada requisito en el SRS debe identificarse para representar

estas diferencias, aclarar y ser explícito. Para esto, se deben identificar los requisitos de la manera siguiente:

- Los clientes deben dar las consideraciones muy cuidadosamente a cada requisito para que se clarifique cualquier omisión que ellos pueden tener.
- Tener diseñadores que hagan diseños correctos y pongan el mismo esfuerzo en todos los niveles del producto del software.

2.3.5.1. Grado de estabilidad: Puede expresarse la estabilidad, por lo que se refiere al número de cambios esperados a cualquier requisito basado en experiencia o conocimiento de eventos venideros que afectan la organización, funciones y a las personas que apoyan el sistema del software.

2.3.5.2. Grado de necesidad: Otra manera de alinear los requisitos es distinguir las clases de requisitos que hay: el esencial, el condicional y optativo.

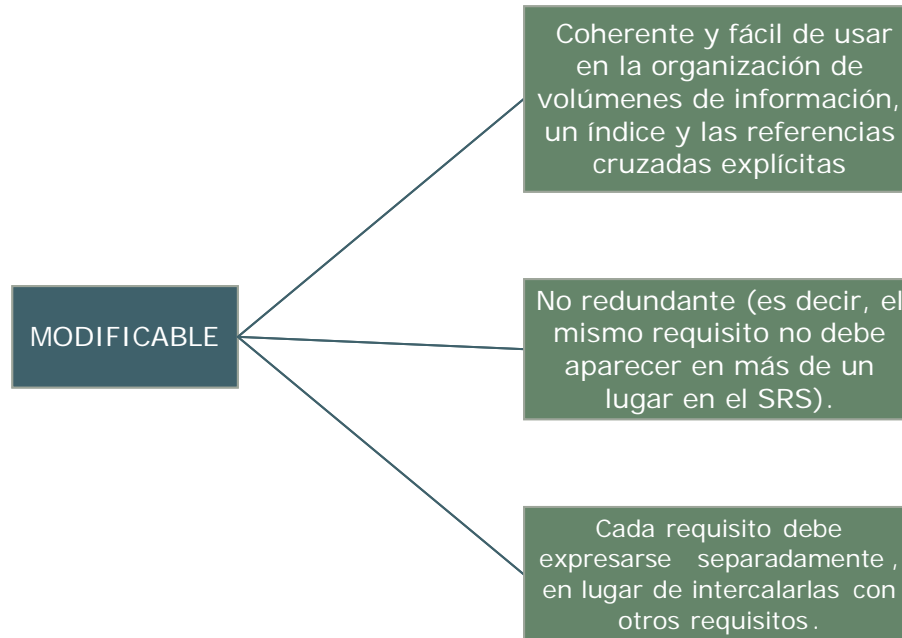


2.3.6. Comprobable: Un SRS es comprobable si y sólo si, cada requisito declarado es comprobable. Un requisito es comprobable si y sólo si, allí existe algún proceso rentable finito con que una persona o la máquina puede verificar que el producto del software reúne el requisito. En general cualquier requisito ambiguo no es comprobable. Los requisitos no verificables, incluyen las declaraciones como "trabaja bien", "interface humana buena" y "normalmente pasará". No pueden verificarse estos requisitos, porque es imposible definir las condiciones "bueno," "bien" o "normalmente". La declaración "el programa nunca entrará en una vuelta infinita" es no verificable porque la comprobación de esta calidad es teóricamente imposible. Un ejemplo de una declaración comprobable es:

El rendimiento del programa se producirá dentro de 20 seg. de evento 60% del tiempo; y se producirá dentro de 30 seg. de evento 100% del tiempo. Esta declaración puede verificarse porque usa condiciones concretas y las cantidades mensurables. Si un método no puede inventarse para determinar si el software reúne un requisito particular, entonces ese requisito debe quitarse o debe revisarse.

2.3.7. Modificable: Un SRS es modificable si y sólo si, su estructura y estilo son tales que puede hacerse cualquier cambio a los requisitos fácilmente, completamente y de forma consistente mientras conserva la estructura y estilo.

Las características necesarias para que un SRS sea modificable se muestran en el siguiente esquema:



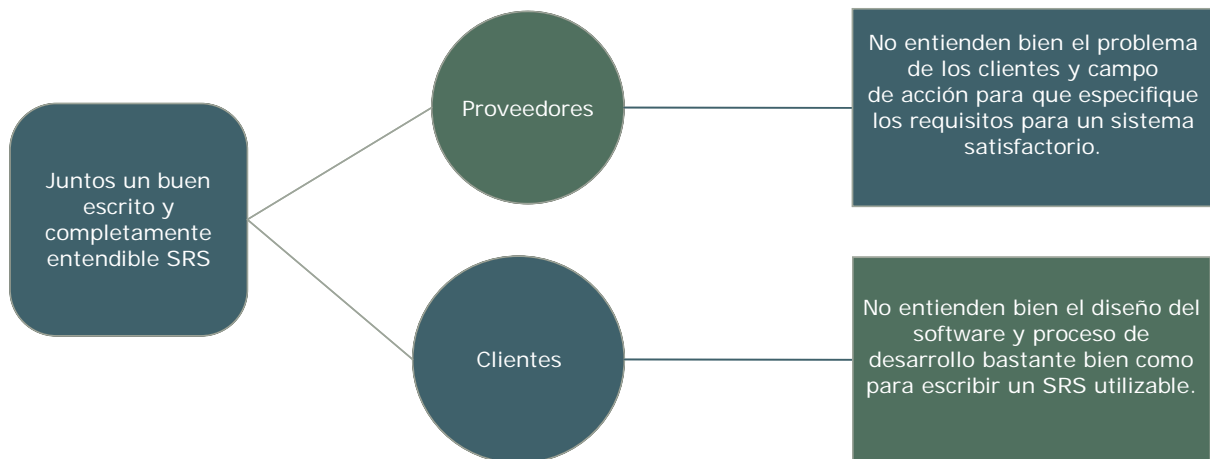
La redundancia no es un error, pero puede llevar fácilmente a los errores. La redundancia puede ayudar hacer un SRS más leíble de vez en cuando, pero un problema puede generarse cuando el documento redundante se actualiza.

2.3.8. Identificable: Un SRS es identificable si el origen de cada uno de sus requisitos está claro y si facilita las referencias de cada requisito en el desarrollo futuro o documentación del mismo. Hay dos tipos de identificabilidad:

- El identificable dirigido hacia atrás (es decir, a las fases anteriores de desarrollo). Esto depende explícitamente en cada requisito, de las referencias de su fuente en los documentos más antiguos.
- El identificable delantero (es decir, a todos los documentos desovados por el SRS). Esto depende en cada requisito en el SRS que tiene un único nombre o número de la referencia. El identificable delantero del SRS es especialmente importante cuando el producto del software entra en el funcionamiento y fase de mantenimiento. Como el código y documentos del plan se modifican, es esencial poder determinar el juego completo de requisitos que pueden afectarse por esas modificaciones.

2.4. Preparación de los JOIN del SRS

El proceso de desarrollo de software debe empezar con el proveedor y con el acuerdo del cliente en lo que el software completo debe hacer.



Existe una situación especial, cuando el sistema y su software se están definiendo concurrentemente. Entonces la funcionalidad, interfases, desarrollo y otros atributos, no son predefinidos, sino que se definen conjuntamente y están sujetos a la negociación y al cambio.

2.5. Evolución de SRS

El SRS puede necesitar evolucionar, así como el desarrollo de las actualizaciones del producto de software. Puede ser imposible especificar a detalle en el momento que el proyecto se inicia (por ejemplo, puede ser imposible definir toda la estructura de la pantalla para un programa interactivo durante la fase de requisitos). Los cambios adicionales pueden suceder según como las deficiencias, las limitaciones e inexactitudes en el SRS, se vayan descubriendo.

Es importante seguir dos consideraciones en este proceso:

- Deben especificarse los requisitos completamente como se es conocido en el momento, aun cuando las revisiones evolutivas pueden preverse como inevitables. Si están incompletos, este hecho debe ser anotado.
- Un proceso de cambio formal debe comenzarse para identificar el control, dejar huella e informe de lo que proyectaron los cambios. Los cambios aprobados en los requisitos deben incorporarse en el SRS de manera que:
 - a. Proporcione un lineamiento de la auditoria exacta y completa de cambios.
 - b. El permiso de la revisión actual y reemplazo de los cambios en el SRS.

2.6. Prototipos

Los prototipos frecuentemente se usan durante una fase de los requisitos de un proyecto. Muchas herramientas existen para generar un prototipo que exhiba algunas características de un sistema, como ser creado muy rápida y fácilmente.

Los prototipos son útiles por las siguientes razones:

- El cliente puede ver el prototipo y reaccionar a este, al leer el SRS. Así, el prototipo proporciona la regeneración rápida.

- El prototipo despliega aspectos que se anticipan a la conducta de los sistemas. Así, no sólo produce las respuestas sino también las nuevas preguntas. Esto ayuda a ver el alcance en el SRS.
- Un SRS basado en un prototipo tiende a sufrir menos cambios durante el desarrollo; de esta manera, se acorta el tiempo de desarrollo.

Un prototipo debe usarse como una manera de sacar los requisitos del software. Pueden extraerse algunas características como pantalla o formatos del reporte directamente del prototipo. Otros requisitos pueden ser inferidos ejecutando los experimentos con el prototipo.

2.7. Generando el diseño en el SRS

Un requisito especifica una función externa visible o atributo de un sistema. Un diseño describe un subcomponente particular de un sistema y/o sus interfases con otros subcomponentes. El diseñador del SRS debe distinguir claramente entre, identificar las restricciones del diseño requeridos y proyectar un plan específico. La nota es que cada requisito en el SRS limita las alternativas del plan. Esto no significa, sin embargo, que cada requisito es el plan. El SRS debe especificar qué funciones serán realizadas, con qué datos, para producir qué resultados, en qué situación y para quien. El SRS se debe enfocar en los servicios a ser realizados. Este normalmente, no debe especificar los puntos del plan, como:

- Partir el software en módulos
- Asignar las funciones a los módulos
- Describir el flujo de información o controles entre los módulos
- Escoger las estructuras de los datos.

2.7.1. Requisitos del plan necesarios: En casos especiales, algunos requisitos pueden restringir el plan severamente. Por ejemplo, seguridad o requisitos de seguridad pueden reflejarse directamente en el plan, como la necesidad de:

- Guardar ciertas funciones en los módulos separadamente
- Las limitaciones que produce el permiso en la comunicación entre algunas áreas del programa
- La integridad de datos mediante chequeos para las variables críticas

Los ejemplos de restricciones del diseño válidos son requisitos físicos, requisitos del desarrollo, normas de desarrollo de software y software de calidad según los estándares. Por consiguiente, los requisitos deben declararse desde un punto de vista completamente externo. Al usar a modelos para ilustrar los requisitos, se debe tener en cuenta, que el modelo sólo indica la conducta externa y no especifica un plan.

2.8 Requisitos del proyecto generados en el SRS

El SRS debe dirigir el producto del software, no el proceso de producir el producto del software. Los requisitos del proyecto representan una comprensión entre el cliente y el proveedor sobre materias contractuales que pertenecen a la producción de software y así no deben ser incluidos en el SRS. Los puntos que normalmente se incluyen se presentan a continuación:

- El Costo
- Los Tiempos de la entrega
- Información de los procedimientos
- Los Métodos de desarrollo de Software
- La convicción de Calidad
- La Aprobación y Criterio de la comprobación
- Los Procedimientos de aceptación.

Se especifican los requisitos del proyecto en otros documentos, generalmente en un plan de desarrollo de software, un software de calidad o una declaración de trabajo.

3. PARTES DE UN SRS

A continuación se presenta una guía del contenido del SRS.

| Tabla de Contenido |
|--|
| 1. Introducción |
| 1.1 Propósito |
| 1.2 Alcance |
| 1.3 Definiciones, siglas, y abreviaciones |
| 1.4 Referencias |
| 1.5 Apreciación global |
| 2. Descripción global |
| 2.1 Perspectiva del producto |
| 2.2 Funciones del producto |
| 2.3 Características del usuario |
| 2.4 Restricciones |
| 2.5 Atención y dependencias |
| 2.6. Repartir proporcionalmente los requisitos |
| 3. Los requisitos específicos |
| Apéndices |
| Índice |

3.1 Introducción

La introducción del SRS debe proporcionar una apreciación global del SRS completo. Debe contener las siguientes subdivisiones:

- Propósito
- Alcance
- Definiciones, siglas y abreviaciones
- Referencias
- Apreciación global

3.1.1. Propósito: Esta subdivisión debe:

- Delinear el propósito del SRS
- Especificar a qué público intencional va dirigido el SRS.

3.1.2. Alcance: Esta subdivisión debe:

- Identificar el producto(s) del software para ser diseñado por el nombre (por ejemplo, Anfitrión DBMS, el Generador del Reporte, etc.)
- Explicar que hará y que no hará el producto(s) del software.
- Describir la aplicación del software especificando los beneficios pertinentes, objetivos y metas
- Ser consistente con las declaraciones similares en las especificaciones de niveles superiores (por ejemplo, las especificaciones de los requisitos del sistema), si existen.

3.1.3. Definiciones, siglas, y abreviaciones: Esta subdivisión debe proporcionar las definiciones de todas las condiciones, siglas y abreviaciones que exigen interpretar el SRS propiamente. Esta información puede proporcionarse por la referencia de uno o más apéndices en el SRS o por la referencia a otros documentos.

3.1.4. Referencias: Esta subdivisión debe:

- Proporcionar una lista completa de todas las referencias de los documentos en otra parte del SRS
- Identificar cada documento por el título, número del reporte (si es aplicable), fecha y publicación de la organización
- Especificar de donde se obtuvieron las referencias (fuentes).

Esta información puede proporcionarse por la referencia a un apéndice o a otro documento.

3.1.5. apreciación global: Esta subdivisión debe:

- Describir lo que el resto del SRS contiene
- Explicar cómo el SRS es organizado.

3.2. Descripción global

Esta sección del SRS debe describir los factores generales que afectan el producto y sus requisitos. Esta sección no declara los requisitos específicos. En cambio, mantiene un fondo de esos requisitos que se definen en detalle en la Sección 3 del SRS y lo hace más fácil entender.

Esta sección normalmente consiste en seis subdivisiones, como sigue:

3.2.1. Perspectiva del producto: Esta subdivisión del SRS debe poner el producto en la perspectiva con otros productos relacionados. Si el producto es independiente y totalmente autónomo, debe declararse como tal. Si el SRS define un producto que es un componente

de un sistema más grande, como frecuentemente ocurre, entonces esta subdivisión debe relacionar los requisitos de ese sistema más grande, relacionados con la funcionalidad del software y debe identificar las interfases entre ese sistema y el software. Un diagrama del bloque que muestre los componentes mayores del sistema más grande, las interconexiones, y las interfases externas puede ser útil en este caso.

Esta subdivisión también debe describir cómo el software opera dentro de las varias restricciones. Por ejemplo, estas restricciones podrían incluir:

3.2.1.1. Interfases del sistema: Esto debe listar cada interfaz del sistema y debe identificar la funcionalidad del software para lograr el requisito del sistema y la descripción de la interfaz para empatar el sistema.

3.2.1.2. Interfases con el usuario: Esto debe especificar:

- Las características lógicas de cada interfaz entre el producto del software y sus usuarios. Esto incluye las características de la configuración (por ejemplo, formatos de la pantalla requeridos, página o esquemas de la ventana, los reportes o menús o disponibilidad de llaves de la función programables) necesarios para lograr los requisitos del software.
- Todos los aspectos para perfeccionar la interfaz con la persona que debe usar el sistema. Esto puede comprender una lista de lo que hace y no hace simplemente basado en cómo el sistema aparecerá ante el usuario. Un ejemplo puede ser un requisito para la opción de mensajes de error largos o cortos. Como todos, estos requisitos deben ser comprobables, deben especificarse en los Atributos de Sistema de Software bajo una sección tituló Facilidad de Uso.

3.2.1.3. Interfases con el hardware: Debe especificar las características lógicas de cada interfaz entre el producto del software y los componentes del hardware del sistema. Esto incluye las características de la configuración (el número de puertos, la instrucción set, etc.), también los dispositivos con los que serán apoyados, cómo ellos serán apoyados y protocolos. Por ejemplo, el apoyo de las terminales puede especificarse cuando tienen full-screen.

3.2.1.4. Interfases con el software: Esto debe especificar el uso de otros productos del software requeridos (por ejemplo, un sistema de dirección de datos, un sistema operativo o un paquete matemático) e interfases con otros sistemas de la aplicación (por ejemplo, la unión entre el Sistema de Cuentas, el Sistema de cobros y un Sistema Mayor General). Para cada uno, el producto del software debe proporcionar:

- El nombre
- El código mnemotécnico
- El número de la especificación
- El número de la versión
- La fuente

Para cada interfaz, se debe proporcionar:

- La discusión del propósito de la interfaz del software en relación con el producto del software
- La definición de la interfaz por lo que se refiere a los mensajes contenidos y los formatos. No es necesario detallar la documentación de la interfaz, pero si se requiere una referencia al documento que define la interfaz.

3.2.1.5. Interfases de comunicaciones: Debe especificar las interfases a las comunicaciones como los protocolos de las redes locales, etc.

3.2.1.6. Restricciones de memoria: Esto debe especificar cualquier característica aplicable y límites en la memoria primaria y la memoria secundaria.

3.2.1.7. Funcionamientos: Esto debe especificar los funcionamientos normales y especiales requeridos por el usuario como:

- Los diferentes modos de funcionamiento en la organización del usuario (por ejemplo, los funcionamientos de inicio del usuario)
- los Periodos de funcionamiento interactivos y periodos de funcionamiento desatendidos
- Datos que procesan las funciones de apoyo
- el Apoyo y funcionamientos de la recuperación.

3.2.1.8. Requisitos de adaptación del Site: Esto debe:

- Definir los requisitos para cualquier dato o secuencia de inicialización que sean específicos de un sitio dado; la misión o el modo operacional (por ejemplo, los límites de seguridad, etc.)
- Especificar el sitio o los rasgos que se deben modificar para adaptar el software a una instalación particular.

3.2.2. Funciones del Producto: Esta subdivisión del SRS debe proporcionar un resumen de las funciones mayores que el software realizará. Por ejemplo, un SRS para un programa de contabilidad puede acostumbrar esta parte a dirigirse al mantenimiento de Cuentas de Cliente, declaración del cliente y preparación de la factura sin mencionar la inmensa cantidad de detalles que cada una de esas funciones requiere. A veces el resumen de la función que es necesario para esta parte puede tomarse directamente de la sección de la especificación en el nivel superior (si existe). Eso asigna las funciones particulares al producto del software. Note que eso es por causa de la claridad.

- Las funciones deben organizarse de modo que la lista de funciones sea entendible para el cliente o cualquiera que lea el documento por primera vez.
- Pueden usarse los métodos Textuales o gráficos para mostrar las funciones diferentes y sus relaciones.

No se piensa que el diagrama muestra un diseño de un producto, sino simplemente muestra la relación lógica entre las variables.

3.2.3. Características del usuario: Esta subdivisión del SRS debe describir las características generales de los usuarios intencionales del producto que incluye nivel educativo, experiencia, y la especialización técnica.

3.2.4. Restricciones: Esta subdivisión del SRS debe proporcionar una descripción general de cualquier otro punto que pueda limitar las opciones de los diseñadores. Éstos incluyen:

- las políticas reguladoras
- las limitaciones del Hardware
- las Interfases a otras aplicaciones
- el funcionamiento Paralelo
- las funciones de la Auditoría
- las funciones de Control
- los requisitos de lenguaje
- los protocolos Señalados (por ejemplo, XON-XOFF, ACK-NACK)
- los requisitos de Fiabilidad
- Credibilidad de la aplicación
- la Seguridad y consideraciones de seguridad

3.2.5. Atenciones y dependencias: Esta subdivisión del SRS debe listar cada uno de los factores que afectan los requisitos declarados en el SRS. Estos factores no son las restricciones del diseño en el software, más bien, son cualquier cambio a ellos; eso puede afectar los requisitos en el SRS. Por ejemplo, una suposición puede ser, que un sistema operativo específico estará disponible en el hardware designado para el producto del software. Si, de hecho, el sistema operativo no está disponible, los SRS tendrían que cambiar de acuerdo a esto.

3.2.6. Prorratear los requisitos: Esta subdivisión del SRS debe identificar los requisitos que pueden tardarse hasta las versiones futuras del sistema.

3.3 Requisitos específicos

Esta sección del SRS debe contener todos los requisitos del software a un nivel de detalle suficiente para permitirles a los diseñadores diseñar un sistema para satisfacer esos requisitos y a los auditores, probar que el sistema satisface esos requisitos. A lo largo de esta sección, cada requisito declarado debe ser externamente perceptible por los usuarios, operadores u otros sistemas externos. Estos requisitos deben incluir por lo menos una descripción de cada entrada (el estímulo) en el sistema, cada salida (la contestación) del sistema y todas las funciones realizadas por el sistema en la salida a una entrada o en el apoyo de la salida. Esta es la parte más grande y más importante del SRS. Para esto, se aplican los siguientes principios:

- deben declararse los requisitos específicos de conformidad con todas las características descritas en la sección de “características del usuario”.
- los requisitos específicos deben tener referencias cruzadas a documentos más actuales que los relacionen.
- Todos los requisitos deben ser singularmente identificables.

- debe prestarse la atención necesaria para organizar los requisitos, de manera que se aumente al máximo la legibilidad.

3.3.1. Interfases externas: Ésta debe ser una descripción detallada de todas las entradas y salidas del sistema del software.



3.3.2. Funciones: Los requisitos funcionales deben definir las acciones fundamentales que deben tener lugar en el software, aceptando y procesando las entradas, procesando y generando las salidas. Éstos generalmente se listan como declaraciones que empiezan con "El sistema debe...."

Éstos incluyen:

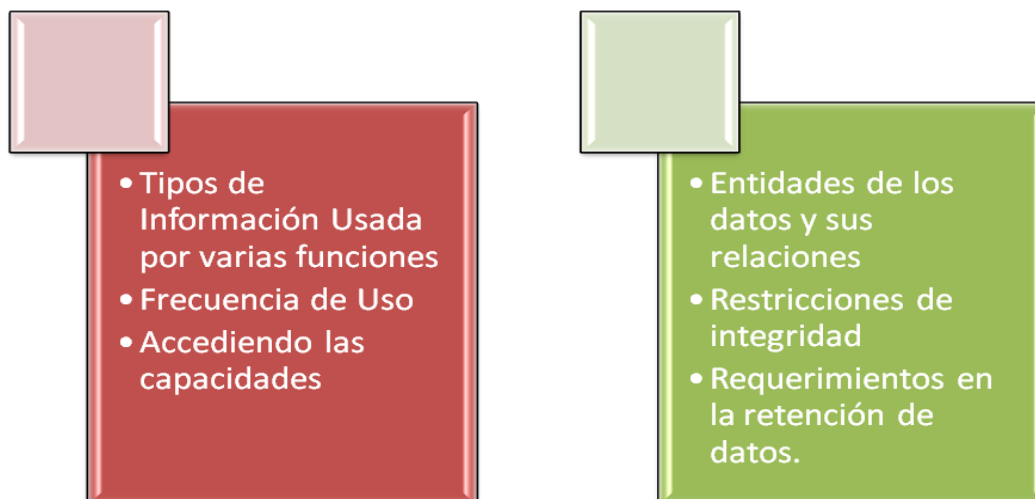


Puede ser apropiado dividir los requisitos funcionales en subfunciones o subprocesos. Esto no implica que el plan del software también se dividirá así.

3.3.3. Requisitos del desarrollo: Esta subdivisión debe especificar los requerimientos estáticos y dinámicos que se pusieron en el software o en la interacción humana con el software en conjunto. Los requisitos estáticos pueden incluir lo siguiente:



3.3.4. Requisitos del banco de datos lógicos: Esto debe especificar los requisitos lógicos para cualquier información que será puesta en un banco de datos. Esto puede incluir:



3.3.5. Restricciones del diseño: Debe especificar las restricciones del diseño que pueden imponerse por otros estándares, limitaciones del hardware, etc.

3.3.5.1. Aceptación de las normas: Esta subdivisión debe especificar los requisitos derivados de estándares existentes o regulaciones. Esto incluye:

- el formato de reporte
- los nombres de los datos
- los procedimientos de contabilidad
- los lineamientos de la Auditoría

Por ejemplo, esto podría especificar los requisitos para el software y rastrear la actividad del proceso. Se necesita rastrear algunas aplicaciones para encontrar al menos las normas reguladoras o financieras. Por ejemplo, un requisito de rastro de auditoría puede declarar que deben grabarse todos los cambios en un banco de datos de la nómina en un archivo del rastro con los valores antes y después del proceso.

3.3.6. Atributos del software del sistema: Hay varios atributos del software que pueden servir como requisitos. Es importante que estos atributos se especifiquen para que su logro pueda verificarse objetivamente. Ejemplos:

3.3.6.1. Fiabilidad: Debe especificar que exigieron los factores al establecer la fiabilidad requerida del sistema del software al momento de la entrega.

3.3.6.2. Disponibilidad: Este punto debe especificar que los factores exigieron garantizar un nivel de disponibilidad definido para el sistema como un punto de control.

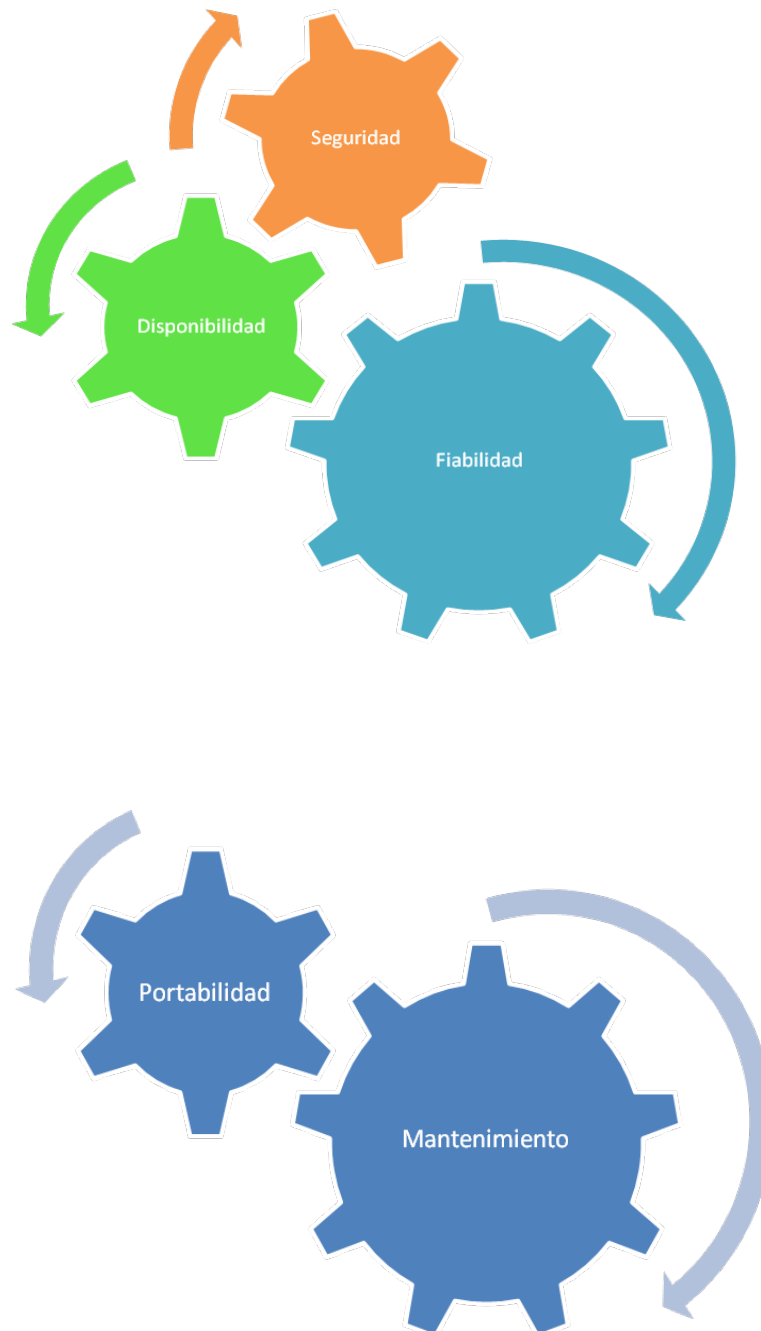
3.3.6.3. Seguridad: Esto debe especificar los factores que protegen el software del acceso accidental o malévolo, uso, modificación, destrucción o descubrimiento. Los requisitos específicos en esta área podrían incluir la necesidad de:

- Utilizar ciertas técnicas de encriptamiento
- Tener Log de entrada o históricos de datos
- Asignar ciertas funciones a módulos diferentes
- Restringir las comunicaciones entre algunas áreas del programa
- verificar la integridad de datos para variables críticas.

3.3.6.4. Mantenimiento: Debe especificar atributos de software que relacionen a la facilidad de mantenimiento del propio software. Puede haber algún requisito con toda seguridad de modularidad, interfases, la complejidad, etc. no deben ponerse los requisitos aquí.

3.3.6.5. Portabilidad: Esto debe especificar atributos de software que relaciona a la facilidad de poner el software a otro servidor y/o sistemas operativos. Esto incluye:

- el Porcentaje de componentes con código cliente-servidor
- el Porcentaje de código del cliente-servidor
- el Uso de un idioma portátil probado
- el Uso de un compilador particular o subconjunto de lenguajes
- el Uso de un sistema operativo particular.



3.3.7. Organizar los requisitos específicos: Los requisitos detallados de los sistemas triviales, generalmente tienden a ser extensos. Por esta razón, se recomienda que se sean cuidadosos al organizar éstos de una manera óptima para que sean entendibles.

3.3.7.1. Modo del sistema: Algunos sistemas se comportan de diferente manera, dependiendo del modo de operación. Por ejemplo, un sistema de control puede tener diferentes juegos de funciones que dependen de su control: entrenando, normal o emergencia. La opción depende de las interfases y del desarrollo que son dependientes del modo de acceso.

3.3.7.2. Clases de usuario: Algunos sistemas proporcionan juegos diferentes de funciones a las diferentes clases de usuarios. Por ejemplo, un sistema de mando de ascensor presenta las capacidades diferentes a los pasajeros, obreros de mantenimiento y bomberos.

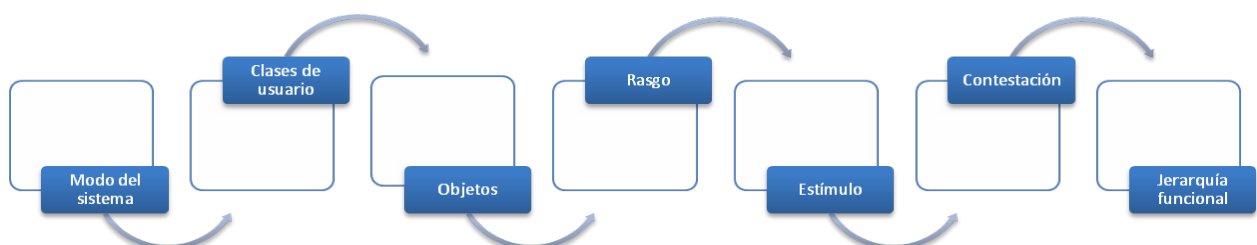
3.3.7.3. Objetos: Los objetos son entidades del mundo real que tienen una contraparte dentro del sistema. Por ejemplo, en un sistema que supervisa pacientes, los objetos incluyen a los pacientes, los sensores, enfermeras, los cuartos, médicos, las medicinas, etc. Por lo tanto, se asocia con cada objeto un juego de atributos (de ese objeto) y funciones (realizadas por ese objeto). Estas funciones también se llaman servicios, métodos o procesos. Nótese que al poner los objetos, estos pueden compartir atributos y servicios. Por lo tanto, estos se agrupan como las clases.

3.3.7.4. Rasgo: Un rasgo es un servicio externamente deseado por el sistema, que puede exigir a una secuencia de entradas efectuar el resultado deseado. Por ejemplo, en un sistema del teléfono, los rasgos incluyen la llamada local, llamada remitida y llamada en conferencia. Cada rasgo generalmente se describe en una secuencia de estímulo-contestación.

3.3.7.5. Estímulo: Algunos sistemas pueden organizarse mejor, describiendo sus funciones basándose en los estímulos. Por ejemplo, pueden organizarse las funciones de un avión automático que aterriza, el sistema en las secciones para la pérdida del control, esquivación del viento, el cambio súbito en el destino, la velocidad vertical excesiva, etc.

3.3.7.6. Contestación: Algunos sistemas pueden organizarse mejor describiendo todas las funciones en el apoyo de la generación de una contestación. Por ejemplo, pueden organizarse las funciones de un sistema del personal en secciones que corresponden a todas las funciones asociadas con los sueldos generados, todas las funciones asociadas con generar una lista actual de empleados, etc.

3.3.7.7. Jerarquía Funcional: Cuando ninguno de los esquemas orgánicos anteriores demuestra ser útil, la funcionalidad global puede organizarse en una jerarquía de funciones organizada por cualesquiera entradas comunes, rendimientos comunes o el acceso de los datos interiores comunes. Pueden usarse diagramas y diccionarios de datos para mostrar las relaciones entre las funciones y datos.



3.3.8. Comentarios adicionales: Siempre que un nuevo SRS se contemple, más de una de las técnicas organizacionales dadas en 3.3.7.7 pueden ser apropiadas. En tal caso, se debe organizar los requisitos específicos para jerarquías múltiples detalladas según las necesidades específicas del sistema. Hay muchas anotaciones, métodos y herramientas de apoyo automatizadas disponibles para ayudar en la documentación de requisitos. La mayor parte del tiempo, su utilidad es una función de organización. Por ejemplo, al organizar por el modo, máquinas de estado finitas o los mapas estatales, estos pueden demostrar utilidad; al organizar por el objeto, el análisis objeto-orientado puede demostrar utilidad; al organizar por el rasgo, las secuencias de estímulo-contestación pueden demostrar utilidad y al organizar por la jerarquía funcional, los datos fluyen según los diagramas y los diccionarios de datos pueden demostrar también utilidad.

3.4. Información de apoyo

La información de apoyo hace más fácil usar el SRS. Esta información Incluye:

3.4.1. Tabla de contenidos e índice: La tabla de contenidos e índice es bastante importante y debe seguir las prácticas de las composiciones generales.


3.4.2. Apéndices: Los apéndices no siempre son considerados parte del SRS real y no siempre son necesarios. Estos pueden incluir:

- Ejemplos de formatos de las entradas/salidas, las descripciones del análisis del costo que se realizó o resultados de estudios del usuario
- Información a fondo que puede ayudar a los lectores del SRS
- Una descripción de los problemas a ser resueltos por el software
- las instrucciones del empaquetamiento especiales para el código y los medios de comunicación para reunir la seguridad, exportar la carga inicial u otros requisitos.

Cuando los apéndices son incluidos, el SRS debe declarar explícitamente si los apéndices serán considerados parte de los requisitos.

4. ANEXOS

4.1. Formato de especificación de requisitos para un proyecto de software

| | | |
|---|--|--------------|
|  | | [Mes de año] |
|---|--|--------------|

| |
|--|
| |
|--|

Especificación de requisitos de software

Proyecto: [Nombre del proyecto]

Revisión [99.99]

Instrucciones para el uso de este formato

Este formato es una plantilla tipo para documentos de requisitos del software.

Está basado y es conforme con el estándar IEEE Std 830-1998.

Las secciones que no se consideren aplicables al sistema descrito podrán de forma justificada indicarse como no aplicables (NA).

Notas:

Los textos en color azul son indicaciones que deben eliminarse y, en su caso, sustituirse por los contenidos descritos en cada apartado.

Los textos entre corchetes del tipo “[Inserte aquí el texto]” permiten la inclusión directa de texto con el color y estilo adecuado a la sección, al pulsar sobre ellos con el puntero del ratón.

Ficha del documento

| Fecha | Revisión | Autor | Verificado dep. calidad. |
|---------|----------|---------------|--------------------------|
| [Fecha] | [Rev] | [Descripcion] | [Firma o sello] |

Documento validado por las partes en fecha: [Fecha]

| Por el cliente | Por la empresa proveedora |
|--|--|
| | |
| Firma [Nombre] | Firma [Nombre] |

CONTENIDO

1 INTRODUCCIÓN

- 1.1 Propósito
- 1.2 Alcance
- 1.3 Personal involucrado
- 1.4 Definiciones, acrónimos y abreviaturas
- 1.5 Referencias
- 1.6 Resumen

2 DESCRIPCIÓN GENERAL

- 2.1 Perspectiva del producto
- 2.2 Funcionalidad del producto
- 2.3 Características de los usuarios
- 2.4 Restricciones
- 2.5 Suposiciones y dependencias
- 2.6 Evolución previsible del sistema

3 REQUISITOS ESPECÍFICOS

- 3.1 Requisitos comunes de los interfaces
 - 3.1.1 Interfaces de usuario
 - 3.1.2 Interfaces de hardware
 - 3.1.3 Interfaces de software
 - 3.1.4 Interfaces de comunicación
- 3.2 Requisitos funcionales
 - 3.2.1 Requisito funcional1
 - 3.2.2 Requisito funcional2
 - 3.2.3 Requisito funcional3
 - 3.2.4 Requisito funcional n
- 3.3 Requisitos no funcionales
 - 3.3.1 Requisitos de rendimiento
 - 3.3.2 Seguridad
 - 3.3.3 Fiabilidad
 - 3.3.4 Disponibilidad
 - 3.3.5 Mantenibilidad
 - 3.3.6 Portabilidad
- 3.4 Otros requisitos

4 APÉNDICES

Introducción

[Inserte aquí el texto]

La introducción de la Especificación de requisitos de software (SRS) debe proporcionar una vista general de la SRS. Debe incluir el objetivo, el alcance, las definiciones y acrónimos, las referencias, y la vista general del SRS.

Propósito

[Inserte aquí el texto]

- *Propósito del documento*
- *Audiencia a la que va dirigido*

Alcance

[Inserte aquí el texto]

- *Identificación del producto(s) a desarrollar mediante un nombre*
- *Consistencia con definiciones similares de documentos de mayor nivel (ej. Descripción del sistema) que puedan existir*

Personal involucrado

| | |
|-------------------------|-------------------------|
| Nombre | [Inserte aquí el texto] |
| Rol | [Inserte aquí el texto] |
| Categoría profesional | [Inserte aquí el texto] |
| Responsabilidades | [Inserte aquí el texto] |
| Información de contacto | [Inserte aquí el texto] |
| Aprobación | [Inserte aquí el texto] |

Relación de personas involucradas en el desarrollo del sistema, con información de contacto.

Esta información es útil para que el gestor del proyecto pueda localizar a todos los participantes y recabar la información necesaria para la obtención de requisitos, validaciones de seguimiento, etc.

Definiciones, acrónimos y abreviaturas

[Inserte aquí el texto]

Definición de todos los términos, abreviaturas y acrónimos necesarios para interpretar apropiadamente este documento. En ella se pueden indicar referencias a uno o más apéndices, o a otros documentos.

Referencias

| Referencia | Título | Ruta | Fecha | Autor |
|------------|----------|--------|---------|---------|
| [Ref.] | [Título] | [Ruta] | [Fecha] | [Autor] |
| | | | | |

Relación completa de todos los documentos relacionados en la especificación de requisitos de software, identificando de cada documento el título, referencia (si procede), fecha y organización que lo proporciona.

Resumen

[Inserte aquí el texto]

- *Descripción del contenido del resto del documento*
- *Explicación de la organización del documento*

Descripción general

Perspectiva del producto

[Inserte aquí el texto]

Indicar si es un producto independiente o parte de un sistema mayor. En el caso de tratarse de un producto que forma parte de un sistema mayor, un diagrama que sitúe el producto dentro del sistema e identifique sus conexiones facilita la comprensión.

Funcionalidad del producto

[Inserte aquí el texto]

Resumen de las funcionalidades principales que el producto debe realizar, sin entrar en información de detalle.

En ocasiones la información de esta sección puede tomarse de un documento de especificación del sistema de mayor nivel (ej. Requisitos del sistema).

Las funcionalidades deben estar organizadas de manera que el cliente o cualquier interlocutor pueda entenderlo perfectamente. Para ello se pueden utilizar métodos textuales o gráficos.

Características de los usuarios

| | |
|-----------------|-------------------------|
| Tipo de usuario | [Inserte aquí el texto] |
| Formación | [Inserte aquí el texto] |
| Habilidades | [Inserte aquí el texto] |
| Actividades | [Inserte aquí el texto] |

Descripción de los usuarios del producto, incluyendo nivel educacional, experiencia y experiencia técnica.

Restricciones

[Inserte aquí el texto]

Descripción de aquellas limitaciones a tener en cuenta a la hora de diseñar y desarrollar el sistema, tales como el empleo de determinadas metodologías de desarrollo, lenguajes de programación, normas particulares, restricciones de hardware, de sistema operativo etc.

Suposiciones y dependencias

[Inserte aquí el texto]

Descripción de aquellos factores que, si cambian, pueden afectar a los requisitos. Por ejemplo una asunción puede ser que determinado sistema operativo está disponible para el hardware requerido. De hecho, si el sistema operativo no estuviera disponible, la SRS debería modificarse.

Evolución previsible del sistema

[Inserte aquí el texto]

Identificación de futuras mejoras al sistema, que podrán analizarse e implementarse en un futuro.

Requisitos específicos

Esta es la sección más extensa y más importante del documento.

Debe contener una lista detallada y completa de los requisitos que debe cumplir el sistema a desarrollar. El nivel de detalle de los requisitos debe ser el suficiente para que el equipo de desarrollo pueda diseñar un sistema que satisfaga los requisitos y los encargados de las pruebas puedan determinar si éstos se satisfacen.

Los requisitos se dispondrán en forma de listas numeradas para su identificación, seguimiento, trazabilidad y validación (ej. RF 10, RF 10.1, RF 10.2,...).

Para cada requisito debe completarse la siguiente tabla:

| | | | |
|-------------------------|---|--|--|
| Número de requisito | [Inserte aquí el texto] | | |
| Nombre de requisito | [Inserte aquí el texto] | | |
| Tipo | <input type="checkbox"/> Requisito <input type="checkbox"/> Restricción | | |
| Fuente del requisito | [Inserte aquí el texto] | | |
| Prioridad del requisito | <input type="checkbox"/> Alta/Esencial | <input type="checkbox"/> Media/Deseado | <input type="checkbox"/> Baja/Opcional |

y realizar la descripción del requisito

La distribución de los párrafos que forman este punto puede diferir del propuesto en esta plantilla, si las características del sistema aconsejan otra distribución para ofrecer mayor claridad en la exposición.

Requisitos comunes de los interfaces

[Inserte aquí el texto]

Descripción detallada de todas las entradas y salidas del sistema de software.

Interfaces de usuario

[Inserte aquí el texto]

Describir los requisitos del interfaz de usuario para el producto. Esto puede estar en la forma de descripciones del texto o pantallas del interfaz. Por ejemplo posiblemente el cliente ha especificado el estilo y los colores del producto. Describa exacto cómo el producto aparecerá a su usuario previsto.

Interfaces de hardware

[Inserte aquí el texto]

Especificar las características lógicas para cada interfaz entre el producto y los componentes de hardware del sistema. Se incluirán características de configuración.

Interfaces de software

[Inserte aquí el texto]

Indicar si hay que integrar el producto con otros productos de software.

Para cada producto de software debe especificarse lo siguiente:

- Descripción del producto software utilizado
- Propósito del interfaz
- Definición del interfaz: conteniendo y formato

Interfaces de comunicación

[Inserte aquí el texto]

Describir los requisitos del interfaces de comunicación si hay comunicaciones con otros sistemas y cuáles son los protocolos de comunicación.

Requisitos funcionales

[Inserte aquí el texto]

Definición de acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.

En ellas se incluye:

- *Comprobación de validez de las entradas*
- *Secuencia exacta de operaciones*
- *Respuesta a situaciones anormales (desbordamientos, comunicaciones, recuperación de errores)*
- *Parámetros*
- *Generación de salidas*
- *Relaciones entre entradas y salidas (secuencias de entradas y salidas, formulas para la conversión de información)*
- *Especificación de los requisitos lógicos para la información que será almacenada en base de datos (tipo de información, requerido)*

Los requisitos funcionales pueden ser divididos en sub-secciones.

Requisito funcional 1

Requisito funcional 2

Requisito funcional 3

Requisito funcional n

Requisitos no funcionales

Requisitos de rendimiento

[Inserte aquí el texto]

Especificación de los requisitos relacionados con la carga que se espera tenga que soportar el sistema. Por ejemplo, el número de terminales, el número esperado de usuarios simultáneamente conectados, número de transacciones por segundo que deberá soportar el sistema, etc.

Todos estos requisitos deben ser medibles. Por ejemplo, indicando “el 95% de las transacciones deben realizarse en menos de 1 segundo”, en lugar de “los operadores no deben esperar a que se complete la transacción”.

Seguridad

[Inserte aquí el texto]

Especificación de elementos que protegerán al software de accesos, usos y sabotajes maliciosos, así como de modificaciones o destrucciones maliciosas o accidentales. Los requisitos pueden especificar:

- *Empleo de técnicas criptográficas.*
- *Registro de ficheros con “logs” de actividad.*
- *Asignación de determinadas funcionalidades a determinados módulos.*
- *Restricciones de comunicación entre determinados módulos.*
- *Comprobaciones de integridad de información crítica.*

Fiabilidad

[Inserte aquí el texto]

Especificación de los factores de fiabilidad necesaria del sistema. Esto se expresa generalmente como el tiempo entre los incidentes permisibles, o el total de incidentes permisible.

Disponibilidad

[Inserte aquí el texto]

Especificación de los factores de disponibilidad final exigidos al sistema. Normalmente expresados en % de tiempo en los que el software tiene que mostrar disponibilidad.

Mantenibilidad

[Inserte aquí el texto]

Identificación del tipo de mantenimiento necesario del sistema.

Especificación de quien debe realizar las tareas de mantenimiento, por ejemplo usuarios, o un desarrollador.

Especificación de cuando debe realizarse las tareas de mantenimiento. Por ejemplo, generación de estadísticas de acceso semanal y mensual.

Portabilidad

[Inserte aquí el texto]

Especificación de atributos que debe presentar el software para facilitar su traslado a otras plataformas u entornos. Pueden incluirse:

- *Porcentaje de componentes dependientes del servidor.*
- *Porcentaje de código dependiente del servidor.*
- *Uso de un determinado lenguaje por su portabilidad.*
- *Uso de un determinado compilador o plataforma de desarrollo.*
- *Uso de un determinado sistema operativo.*

Otros requisitos

[Inserte aquí el texto]

Cualquier otro requisito que no encaje en ninguna de las secciones anteriores.

Por ejemplo:

Requisitos culturales y políticos

Requisitos Legales

Apéndices

[Inserte aquí el texto]

Pueden contener todo tipo de información relevante para la SRS pero que, propiamente, no forme parte de la SRS.

4.2. Referencias y normas recomendadas para usar simultáneamente con la IEEE-830:1998

References

This recommended practice shall be used in conjunction with the following publications.

ASTM E1340-96, Standard Guide for Rapid Prototyping of Computerized Systems.¹

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.²

IEEE Std 730-1998, IEEE Standard for Software Quality Assurance Plans.

IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning.

IEEE Std 828-1998, IEEE Standard for Software Configuration Management Plans.

IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.

IEEE Std 982.2-1988, IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software.

IEEE Std 1002-1987 (Reaff 1992), IEEE Standard Taxonomy for Software Engineering Standards.

IEEE Std 1012-1998, IEEE Standard for Software Verification and Validation.

IEEE Std 1012a-1998, IEEE Standard for Software Verification and Validation: Content Map to IEEE/EIA 12207.1-1997.

IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions.

IEEE Std 1028-1997, IEEE Standard for Software Reviews.

IEEE Std 1042-1987 (Reaff 1993), IEEE Guide to Software Configuration Management.

IEEE Std 1058-1998, IEEE Standard for Software Project Management Plans.

IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes.

IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications.

4.3. Pautas para la conformidad de la norma IEEE 830:1998 con IEEE/EIA 12207.1-1997

Guidelines for compliance with IEEE/EIA 12207.1-1997

B.1 Overview

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

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

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

- IEEE/EIA 12207.1-1997, IEEE/EIA Guide for Information Technology—Software life cycle processes—Life cycle data;
- IEEE/EIA 12207.2-1997, IEEE/EIA Guide for Information Technology—Software life cycle processes—Implementation considerations; and
- Additions to 11 SESC standards (i.e., IEEE Std 730, 828, 829, 830, 1012, 1016, 1058, 1062, 1219, 1233, 1362) to define the correlation between the data produced by existing SESC standards and the data produced by the application of IEEE/EIA 12207.1-1997.

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

B.1.1 Scope and purpose

Both IEEE Std 830-1998 and IEEE/EIA 12207.1-1997 place requirements on a Software Requirements Description Document. The purpose of this annex is to explain the relationship between the two sets of requirements so that users producing documents intended to comply with both standards may do so.

B.2 Correlation

This clause explains the relationship between IEEE Std 830-1998 and IEEE/EIA 12207.0-1996 and IEEE/EIA 12207.1-1997 in the following areas: terminology, process, and life cycle data.

B.2.1 Terminology correlation

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

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

B.2.2 Process correlation

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

B.2.3 Life cycle data correlation

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

B.3 Content mapping

This clause provides details bearing on a claim that an SRS complying with this recommended practice would also achieve "document compliance" with the SRD described in IEEE/EIA 12207.1-1997. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA 12207.1-1997. That row is reproduced in Table B.1 of this recommended practice.

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

| Information item | IEEE/EIA 12207.0-1996 Clause | Kind of documentation | IEEE/EIA 12207.1-1997 Clause | References |
|-----------------------------------|------------------------------|---|------------------------------|---|
| Software Requirements Description | 5.1.1.4, 5.3.4.1, 5.3.4.2 | Description (See note for 6.22.1 of IEEE/EIA 12207.1-1997.) | 6.22 | IEEE Std 830-1998; EIA/IEEE J-STD-016, F.2.3, F.2.4; MIL-STD 961D. Also see ISO/IEC 5806, 5807, 6593, 8631, 8790, and 11411 for guidance on use of notations. |

The requirements for document compliance are discussed in the following subclauses:

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

- B.3.2 discusses compliance with the generic content guideline (the “kind” of document) noted in column 3 of Table B.1 as a “description”. The generic content guidelines for a “description” appear in 5.1 of IEEE/EIA 12207.1-1997.
- B.3.3 discusses compliance with the specific requirements for a Software Requirements Description noted in column 4 of Table B.1 as prescribed by 6.22 of IEEE/EIA 12207.1-1997.
- B.3.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA 12207.0-1996 as described in 4.2 of IEEE/EIA 12207.1-1997.

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

The information requirements for an SRD are those prescribed by 5.1.1.4, 5.3.4.1, and 5.3.4.2 of IEEE/EIA 12207.0-1996. The requirements are substantively identical to those considered in B.3.3 of this recommended practice.

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

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

The purpose of a description is:

IEEE/EIA 12207.1-1997, subclause 5.1.1: Purpose: Describe a planned or actual function, design, performance, or process.

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

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

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

| IEEE/EIA 12207.1-1997 generic content | Corresponding clauses of IEEE Std 830-1998 | Additions to requirements of IEEE Std 830-1998 |
|--|---|---|
| a) Date of issue and status | — | Date of issue and status shall be provided. |
| b) Scope | 5.1.1 Scope | — |
| c) Issuing organization | — | Issuing organization shall be identified. |
| d) References | 5.1.4 References | — |
| e) Context | 5.1.2 Scope | — |
| f) Notation for description | 4.3 Characteristics of a good SRS | — |
| g) Body | 5. The parts of an SRS | — |
| h) Summary | 5.1.1. Overview | — |
| i) Glossary | 5.1.3 Definitions | — |
| j) Change history | — | Change history for the SRD shall be provided or referenced. |

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

The specific content requirements for an SRD in IEEE/EIA 12207.1-1997 are prescribed by 6.22 of IEEE/EIA 12207.1-1997. A compliant SRD shall achieve the purpose stated in 6.22.1 of IEEE/EIA 12207.1-1997.

The purpose of the SRD is:

IEEE/EIA 12207.1-1997, subclause 6.22.1: Purpose: Specify the requirements for a software item and the methods to be used to ensure that each requirement has been met. Used as the basis for design and qualification testing of a software item.

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

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

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

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

| IEEE/EIA 12207.1-1997 specific content | Corresponding clauses of IEEE Std 830-1998 | Additions to requirements of IEEE Std 830-1998 |
|--|---|---|
| a) Generic description information | See Table B.2 | — |
| b) System identification and overview | 5.1.1 Scope | — |
| c) Functionality of the software item including: – Performance requirements – Physical characteristics – Environmental conditions | 5.3.2 Functions 5.3.3 Performance requirements | Physical characteristics and environmental conditions should be provided. |
| d) Requirements for interfaces external to software item | 5.3.1 External interfaces | — |
| e) Qualification requirements | — | The requirements to be used for qualification testing should be provided (or referenced). |
| f) Safety specifications | 5.2.4 Constraints | — |
| g) Security and privacy specifications | 5.3.6.3 Security | — |
| h) Human-factors engineering requirements | 5.2.3 User characteristics 5.2.1.2 User interfaces | — |
| i) Data definition and database requirements | 5.3.4 Logical data base requirements | — |
| j) Installation and acceptance requirements at operation site | 5.2.1.8 Site adaptation requirements | Installation and acceptance requirements at operation site |
| k) Installation and acceptance requirements at maintenance site | — | Installation and acceptance requirements at maintenance site |
| l) User documentation requirements | — | User documentation requirements |
| m) User operation and execution requirements | 5.2.1.7 Operations | User execution requirements |

BIBLIOGRAFÍA

1. "SWEBOK, Software Engineering Body of Knowledge". 2004.
2. Sommerville, I., "Software Engineering. 7th edition". Addison Wesley. 2005
3. Pressman R.S. "Ingeniería del software. Un enfoque práctico". McGraw-Hill/Interamericana, 5º edición, 2001. Traducido del inglés "Software Engineering: A Practitioner's Approach", McGrawHill, European adaption.
4. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications: IEEE, 1998.
5. Jacobson, I., Booch, G. y Rumbaugh, J., "El proceso unificado de modelado", Addison Wesley, 2000.
6. Rolland C. Praskash, N. "From conceptual modeling to requirement engineering", Annals of software engineering 10 (2000) 151-176
7. "IEEE-STD-830-1998: Práctica recomendada para las especificaciones de requisitos de software", Universidad Nacional de Colombia 2008