**ESPECIFICACION DE REQUERIMIENTOS DE SOFTWARE (SRS)**

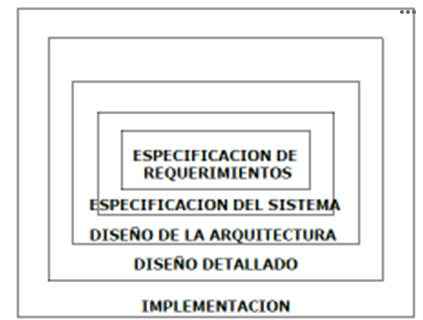
Hoy verá la importancia de la especificación de requerimientos y las características que tienen que cumplir.

**¿Que es SRS?**

Es la documentación de los requerimientos. También podemos decir que es una herramienta de comunicación de:

* la comprensión del dominio
* el negocio
* el sistema
* Es parte de un acuerdo contractual
* Papel clave en el testeo

La especificación de requerimientos es parte del acuerdo contractual y tiene un papel clave en el testeo.



En este esquema vemos como a partir de la especificación de requerimientos van apareciendo distintas capas que tienen que ver con el desarrollo de software.

Los beneficios de tener un documento de especificación de requerimientos son: (IEEE 830)

* Establece las bases del acuerdo consumidores-proveedores sobre qué debe hacer el producto
* Reduce el esfuerzo de desarrollo
* Provee una base para estimaciones
* Provee una “baseline” para validación y verificación
* Facilita la transferencia del producto
* Sirve como base para mejoras

Las especificaciones clásicas u ortodoxas, incluyen características como lo que tiene que proveer el sistema:

Núcleo: Conjunto de definiciones de funciones que establecen las funciones a ser ejecutadas por el sistema y están soportadas por las descripciones de las estructuras de datos a ser procesados

Típicamente incluyen:

* Funcionamiento normal: entradas, salidas, operaciones y control
* Funcionamiento anormal, excepciones
* Propiedades de las que depende: ej. Seguridad
* Performance
* Calidad: mantenibilidad, modificabilidad
* Cambios esperados
* Especificaciones de datos
* Definiciones de estructuras de datos
* Volúmenes (tasa de flujo de datos)
* Fuentes y destinos de datos

Pero estamos subestimando algunos conceptos importantes:

* objetivos
* aspectos deseables
* restricciones en su desarrollo

Estos aspectos son lo que potencia las metodologías ágiles. Cuando escribimos una historia de usuario no sólo se describe la funcionalidad sino también el objetivo de esa funcionalidad. Entonces se puede decir que en una especificación clásica que describe lo que se necesita también necesita el para que. Ese Para que no da un contexto para ayudarnos a tomar decisiones.

**UNA VISION AMPLIA DE LA SR**

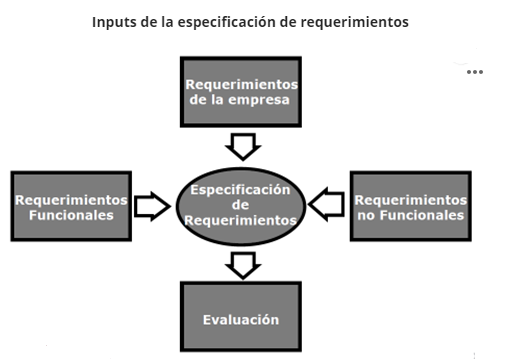
Complementar la especificación funcional con:

* La comprensión del dominio
* La comprensión de las restricciones

Incluir: La descripción de los requerimientos en la empresa expresados en términos de los fenómenos comunes a la empresa y al dominio del sistema.

No incluir en la SRS:

* Requerimientos del proyecto
* Diseños
* Planes de aseguramiento de calidad



En este cuadro puede observar las distintas entradas que tiene la especificación de requerimientos.

Los requerimientos funcionales, son las funcionalidades, operaciones, módulos que tiene que implementar la aplicación.

Los atributos de performance, disponibilidad, modificabilidad, entre otros, son requerimientos no funcionales.

Los requerimientos de la empresa se refieren al contexto de la empresa, el dominio y por otro lado las restricciones a nivel de negocio.

**¿Quiénes son los usuarios de las SRS?**

* Ejecutivos
* Gerentes
* Expertos en el dominio
* Arquitectos
* Diseñadores e implementadores
* Testeadores

**CARACTERISTICAS DE UNA BUENA SRS (IEEE)**

Una buena especificación de requerimientos debe ser:

* Correcta
* No ambigua
* Completa
* Consistente
* Rankeada
* Verificable
* Modificable
* Traceable

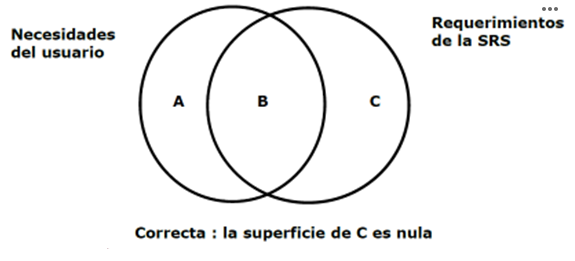
- Entenderlas pero no memorizar todo lo siguiente que explica cada una.

**Correcta**

Una especificación es correcta si no tiene errores en términos de correctitud.

“Sí y sólo si todo requerimiento formulado en ella debe ser satisfecho por el software”

Pero puede ser correcta pero no abarcar todas las necesidades de usuario



**No ambigua**

Según IEEE, Una SRS es inequívoca si, y solo si, cada requisito establecido en él tiene una sola interpretación. Como mínimo, esto requiere que cada característica del producto final se describa utilizando un solo término único.

En los casos en que un término utilizado en un contexto particular pueda tener múltiples significados, el término debe incluirse en un glosario en el que su significado se haga más específico.

Según la RAE, encontramos en su 22da edición , la siguiente definición:

Ambigüedad: cualidad de ambiguo

Ambiguo, gua:

(Del lat. ambigŭus).

1. adj. Dicho especialmente del lenguaje: Que puede entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión.

2. adj. Dicho de una persona: Que, con sus palabras o comportamiento, vela o no define claramente sus actitudes u opiniones.

3. adj. Incierto, dudoso

Entonces podemos decir que **una SRS es no ambigua:**

* “Sí y sólo sí todo requerimiento formulado en ella tiene una sola interpretación”, equivale a que no hay incertidumbre ni más de un significado
* Ello requiere que al menos cada característica del producto final se describa con un único término
* Debe evitarse toda afirmación ambigua.
* “Todos los clientes tienen el mismo campo de control”
* “Todos los archivos son controlados por un bloque de control de archivo”
* El lenguaje natural es fuente de ambigüedades
* Los lenguajes de Especificación Formal ayudan a evitarlas
* Tener presenta la SRS como documento de interfase

**SRS Completa.**

“Una SRS es completa si incluye los siguientes elementos:

a)Todos los requerimientos significativos que impone al software

b)Todas las respuestas del software a todas las clases posibles de inputs

c)Todas las etiquetas y referencias a las figuras, tablas y diagramas de la SRS y las definiciones de los términos y unidades de medida”

* Incluye todo lo que se supone que hace el software
* La “completitud” es el atributo más difícil de definir y de detectar violaciones. Se trata de detectar ausencias a partir de analizar lo presente en la SRS.
* Ventajas del uso de prototipos.
* **Un principio de redacción: el software hace sólo lo que dice en la SRS.**
* También define todas las respuestas del software, a todas las clases de datos de input en todas las clases de situaciones realizables.
* Incluye los inputs válidos y los no válidos.
* El output también puede ser función del estado actual del sistema.

No podemos dejar de lado el seguimiento de los estándares que la organización haya resuelto aceptar.

Si no se utiliza un standard, señalarlo y explicar por que se omite.

**SRS Consistente.**

Aquí hablamos de consistencia interna:

“Una SRS es internamente consistente sí y sólo sí ningún subconjunto de requerimientos está en conflicto”

Existen tres tipos posibles de conflictos entre:

a) Características de objetos del mundo real

b) Acciones

c) Descripciones de objetos del mundo real

Otros tipo de inconsistencias:

* Comportamiento conflictivo: dos partes especifican diferentes y conflictivos:
* estímulos para inducir una respuesta:
* respuestas a un mismo estímulo y condición
* Términos conflictivos: dos términos diferentes se utilizan para definir lo mismo
* Características conflictivas: dos partes requieren rasgos contradictorios
* Inconsistencia temporal: dos partes requieren comportamientos temporales contradictorios

**SRS rankeada (por importancia y/o estabilidad) 🡪 (hay algunas mas impoirtanbte quje otras pero son todas imporantes)**

“Si cada requerimiento tiene una identificación que indica la importancia o estabilidad de cada requerimiento particular” esta identificación ayuda a:

* que los clientes consideren mejor cada requerimiento
* que los desarrolladores toman mejores consideraciones de diseño y asignan esfuerzos más adecuados
* Estabilidad: se puede expresar en términos del número de cambios que se esperan
* Necesidad: Los requerimientos se pueden clasificar en

a )Esenciales: el software no es aceptable sin estos

b) Condicionales: pueden mejorar el software, pero no es inaceptable si está ausente

c) Opcional: esa función puede o no estar presente

**SRS verificable**

Todo requerimiento incluido en ella es “verificable”.

Existe un proceso finito y efectivo en costo con el que chequear que el sistema a construir satisface el requerimiento. En general los requerimientos ambiguos no son verificables

Ejemplos de requerimientos verificables y no verificables:

**Requerimientos no verificables:**

* “El producto debe trabajar bien” o “El producto debería tener una buena interfase”.
* “El programa no debe entrar nunca en un loop infinito”.
* “El output del programa debe estar habitualmente dentro de los 10 seg.”.

**Requerimiento verificable.**

“ A partir de producido el evento X, se obtendrá el output del proceso:

Debe estar dentro de los 20 seg. el 60% de las transacciones; y dentro de los 30 seg. el 99% de las transacciones”.

Las Acciones para corregir los requerimientos no verificables son:

* Eliminar o revisar requerimientos no verificables.
* Transformar en verificables a los no verificables
* Fuentes de no verificabilidad
* Ambigüedades
* Problemas no resolubles
* Cantidades no medibles

**Pregunta de examen: ¿Este requerimiento es verificable?**

**SRS Modificable. 🡪 Tiene que ver con la escalabilidad**

**Con respecto a esta característica se puede indicar que:**

Cualquier cambio puede hacerse fácilmente manteniendo la “completitud” y consistentemente.

También requiere tener una organización coherente y fácil de usar y no ser redundante.

**SRS Rastreable.**

Podemos decir que es rastreable si:

Si es claro el origen de cada requerimientos y facilita la referencia de cada requerimiento en futuros desarrollos o mejoras de las documentación.

**Tipos de rastreabilidad:**

* hacia atrás
* hacia adelante

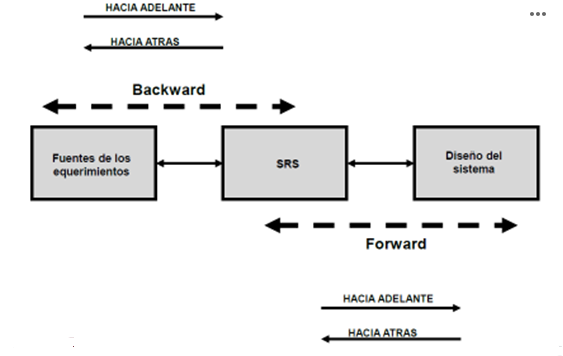
**Beneficios:**

* Importancia en operaciones y mantenimiento
* Claridad del origen de cada requerimiento
* Facilita negociar los requerimientos
* Facilita referenciar cada requerimiento individual establecido en la SRS

**Los objetivos de la “rastreabilidad”**

La rastreabilidad es importante en el diseño de los componentes y en el testing.

A continuación puede observar la trazabilidad en el ciclo de desarrollo.



**Tipos de rastreabilidad:**

**Podemos utilizar estos métodos:**

* numeración de párrafos
* un requerimiento por párrafo
* identificar cada requerimiento
* convenciones sintácticas

Otros criterios:

Estos criterios son complementarios de los criterios abordados en la IEEE, aunque en algunos casos tienen el mismo sentido:

**Otros criterios**

* Comprensible para los consumidores
* Independiente del diseño
* Concisa
* Organizada
* Anotada
* Utilizable en operación y mantenimiento

**Cuando hablamos de que debe ser compresible para los consumidores**, nos referimos a la utilización del lenguaje natural ya que permiten alcanzar las metas anteriores, pero son imposibles de comprender por los que están fuera del área.

De qué sirve tener una especificación que en realidad no se entiende. Mientras lo especificamos en lenguaje natural podemos identificar inconsistencias y ambigüedades.

•Primeros lectores de una SRS

•Clientes y usuarios

•No ambigüedad

•Es una condición clara de la comprensibilidad

**Independiente del diseño**.

* No implica una arquitectura específica de software o algoritmo.
* No limita a una alternativa de diseño
* Factibilidad de diseños potenciales
* Hay excepciones aceptables

En cuanto a excepciones aceptables, a veces sucede que necesitamos indicar ciertas restricciones porque tenemos que utilizar cierta tecnología r porque es un requerimiento, porque nuestro sistema se tiene que adaptar o integrar con otros sistemas entonces más allá de tecnologías arquitecturas de servicios micro servicios o lo que fuera, tenemos que respetar cierta tecnología para que esta integración suceda ,incluso puede ser que se necesite más integración: no tenemos que usar tal base de datos tal plataforma tales servidores y eso de alguna forma condiciona de a poco.

**SRS Concisa, organizada y anotada**

CONCISA. A igualdad de cualidades, la mejor es la más corta. Tiene que tener lo mínimo indispensable, no debemos agregar cosas que en realidad son agregados .

ORGANIZADA. Una SRS es organizada si los requerimientos que contiene son fácilmente localizables.

ANOTADA. Las anotaciones a una SRS proveen guía al desarrollo de la organización del documento. Hay dos casos de anotaciones requeridas necesidad y estabilidad

**SRS Utilizable en operación y mantenimiento**

Debe estar disponible para alguno de estos casos.

•Uso en el eventual reemplazo del software.

•Para el personal de mantenimiento

•Importante: Modificar requiere conocer el origen de la función

Vea el siguiente video para ampliar sus conocimientos de lo visto

**Estándares disponibles.**

Existen una serie de estándares para poder documentar las especificaciones, por ejemplo:

* Departamento de Defensa (DoD): DI-MCCR-80025A
* NASA: SMAP-DID-P200-SW
* IEEE/ANSI 830-1993
* Naval Research Laboratory: SRS del programa de vuelo del A-7E

Pero el estándar de la IEEE está organizado de una forma clara y simple, por lo que debemos tenerlo en cuenta. **Tenerlo en cuenta nomás**

**Indice del Standard de IEEE**

1. Introducción

1.1. Propósito

1.2. Alcance

1.3. Definiciones, acrónimos y abreviaturas

1.4. Referencias

1.5. Overview

2. Descripción general

2.1. Perspectiva del producto

2.2. Funciones del producto

2.3. Características del usuario

2.4. Restricciones generales

2.5. Supuestos y dependencias

3. Requerimientos específicos

Apéndices

Este índice abarca toda los aspectos de la documentación de una especificación de requerimientos. A continuación se explica cada uno de los puntos.

**1. Introducción**

En este punto debemos proveer una visión global de la SRS.

**1.1. Propósito**

Aquí se delinea el propósito de la SRS y se especifica a quien se dirige.

**1.2. Alcance**

En este capítulo se explica los productos de SW, que hará y que no hará cada uno. Describimos la aplicación.

**1.3. Definiciones, acrónimos y abreviaturas**

Aquí incluimos las definiciones de los términos, acrónimos y abreviaturas requeridas para interpretar las especificaciones de requerimientos de software.

**1.4. Referencias**

Aquí detallamos una lista completa de todos los documentos referenciados.

**1.5. Overview**

EN este capítulo describimosr qué contiene el resto de la SRS y explicar cómo está organizada la SRS.

**2. Descripción General**

Describe los factores generales que afectan al producto y a los requerimientos, facilita su comprensión

**2.1. Perspectiva del producto**

* Detallamos la relación con otros productos o proyectos
* Productos independientes
* Componentes de un sistema o de un proyecto:
* Hardware y equipamiento periférico
* Diagrama de bloques
* Restricciones de diseño

**2.2. Funciones del producto (características del producto)**

* Resumen de las funciones que ejecutará el software.
* Comprensibilidad
* Diagrama de bloques
* No establece requerimientos específicos,

**2.3. Características del usuario**

* Características generales del usuario
* Restricciones impuestas por los interactuantes

Aquí detallamos cuales son los perfiles de usuario en términos de usabilidad o accesibilidad o de nivel de entendimiento necesario o uso del producto.

**2.5. Supuestos y dependencias**

Son los factores que afectan los requerimientos, las restricciones de diseño y los cambios que pueden afectar los requerimientos en la SRS.

**2.4. Restricciones generales**

Son los límites a las opciones para diseñar el sistema:

* Políticas regulatorias
* Limitaciones de hardware
* Interfases con otras aplicaciones
* Operaciones paralelas
* Funciones de auditoría
* Funciones de control
* Requerimientos de lenguajes de alto nivel
* Protocolos de “signal handshake” (ej: XON/XOFF)
* Criticalidad de la aplicación
* Consideraciones de seguridad (Safety and Security)

**3. Requerimientos específicos**

* El sector mayor y más importante de la SRS
* Presentación y conceptualización del desarrollo de los requerimientos
* El contexto de la ingeniería de requerimientos
* Se organiza en torno a un conjunto de conceptos

**Conceptos involucrados**

**Interfaces externas**

En este punto se reflejar una descripción detallada de todos los inputs y outputs del sistema:

Refleja contenidos y formatos:

* Nombre del ítem
* Propósito
* Fuente/destino del input/output
* Rango de validez, precisión, tolerancia
* Unidad de medida
* Timing
* Relaciones con otros inputs/outputs
* Formatos y organización de pantallas y ventanas
* Formatos de datos y comandos
* Mensajes de finalización

**Requerimientos funcionales**

Definen las acciones del sistema al aceptar y procesar los inputs y en generar los outputs. En general tienen la forma “el sistema debe”

Incluye (entre otras):

* Chequeos de validez de los inputs
* Secuencias de operaciones
* Respuestas a situaciones anormales (overflow, manejo de errores)
* Relaciones entre inputs y outputs
* Puede convenir el particionamiento en subfunciones, sin que ello signifique que el diseño se particione así.

**Performance y restricciones de diseño**

**Requerimientos de performance:**

Son requerimientos estáticos y dinámicos del software o de la interacción humana del software como un todo

Estos requerimientos estáticos incluyen:

* Cantidad de terminales
* Usuarios simultáneos
* Cantidad y tipo de información a manejar
* Requerimientos dinámicos incluyen:
* Número de tareas y transacciones a procesar por unidad de tiempo, carga normal y picos
* Restricciones de diseño: se imponen por estándares, hardware, software disponible, etc.

**Atributos del sistema**

Hay atributos del software que pueden funcionar como requerimientos. Son atributos tales como:

* Confiabilidad
* Disponibilidad
* Seguridad
* Mantenibilidad
* Portabilidad

Vea el siguiente video para ampliar los conocimientos aquí propuestos: