Capítulo

3

Especificación de Requisitos

Centro Asociado Palma de Mallorca

Tutor: Antonio Rivero Cuesta

3 Especificación de Requisitos

3.1 Introducción.

Veremos la labor de análisis y definición de los requisitos que ha de cumplir un proyecto de software.

Esta labor da lugar a la especificación del software.

3.2 Objetivos

Conocer el concepto de especificación de un sistema software, obtención, análisis y validación de requisitos.

Comprender la importancia que tiene la obtención de requisitos.

Conocer y distinguir los diferentes tipos de requisitos que se presentan en la elaboración de un sistema software.

Conocer las técnicas para la captura de requisitos y ser capaz de elaborar un SRD.

Reconocer diferentes notaciones para elaborar los diagramas que se emplean en la elaboración del SRD.

3.3 Modelado de sistemas.

El Modelado de los sistemas tiene como objetivo entender mejor el funcionamiento requerido y facilitar la comprensión de los problemas que se plantean en el nuevo sistema a desarrollar. Se establecen Modelos Conceptuales que reflejen por un lado la organización de la información y por otro las transformaciones que se deben realizar con dicha información.

3.3.1 Concepto de modelo

Un modelo conceptual es todo aquello que nos permite conseguir una abstracción lógico-matemática del mundo real, y que facilita la comprensión del problema a resolver.

Se trata de ofrecer una visión de alto nivel, sin explicar detalles concretos del mismo.

Debe explicar qué debe hacer el sistema y no como. Objetivos del modelo:

- Facilitar la comprensión del problema a resolver.
- Establecer un marco para la discusión.
- Fijar las bases para realizar el diseño.
- Facilitar la verificación del cumplimiento de los objetivos del sistema.

3.3.2 Técnicas de modelado

Técnicas útiles para realizar el modelado de un sistema software.

3.3.2.1 Descomposición, modelo jerarquizado

Cuando un problema es complejo, se descompone en otros problemas más sencillos. De esta manera se establece un Modelo Jerarquizado en el que el problema queda subdividido en un cierto número de subproblemas.

Por ejemplo: si queremos modelar un sistema para la gestión de una empresa, este sistema se puede descomponer en los subsistemas siguientes: sistema de nominas, sistema de contabilidad, etc. Este tipo de descomposición se denomina **Descomposición Horizontal** y trata de descomponer **funcionalmente** el problema.

Ahora, cada uno de los subsistemas, se pueden descomponer a su vez en otros más simples. Por ejemplo, el sistema de nominas se puede dividir en: realización de nominas, pagos a la seguridad social, etc.

Cuando se descompone un modelo tratando de detallar su **estructura,** lo denominamos **Descomposición Vertical.** Por ejemplo, la realización de nominas se descompone según la secuencia: entrada de datos del trabajador, calculo de ingresos brutos, etc.

3.3.2.2 Aproximaciones sucesivas

Es corriente que el sistema software que se quiere modelar sustituya a un proceso de trabajo ya existente, que se realiza de forma totalmente manual. En este caso se puede crear un modelo de partida basado en la forma de trabajo anterior que tendrá que ser depurado mediante aproximaciones sucesivas hasta alcanzar un modelo final.

3.3.2.3 Empleo de diversas notaciones

Lo óptimo es emplear varias notaciones juntas cuando sea necesario. También se utilizan las herramientas CASE, que son herramientas de ayuda al análisis y al diseño, que combinan texto, tablas, gráficos, diagramas etc.

3.3.2.4 Considerar varios puntos de vista

Se debe elegir el más conveniente. En ocasiones será más adecuado adoptar el punto de vista del futuro usuario del sistema, en otras será más adecuado adoptar el punto de vista del mantenedor del sistema, en algunos modelos será suficiente con perfilarlo desde el punto de vista funcional, etc.

Para la elección será conveniente esbozar distintas alternativas y elegir aquellas que resulten la más idónea.

3.3.2.5 Realizar un análisis del dominio

Entendemos por dominio el campo de aplicación en el que se encuadra el sistema a desarrollar. Por ejemplo, si tenemos que desarrollar un sistema para el seguimiento de la evolución de los pacientes de un hospital, este sistema quedara encuadrado dentro de las aplicaciones para la gestión de hospitales.

En este campo como en cualquier otro, existe desde siempre una cierta manera de realizar las cosas y una terminología ya cuñada que debe ser respetada y tenida en cuenta. A esto es lo que llamamos **Realizar un Análisis del Dominio de Aplicación**.

Para realizar este análisis es conveniente estudiar aspectos como:

- Normativa que afecte al sistema.
- Otros sistemas semejantes.
- Estudios recientes en el campo de la aplicaron, etc.

Estudiar el dominio de la aplicación nos reportará las siguientes ventajas:

Facilitar la comunicación entre analista y usuario del sistema: la creación de un nuevo modelo requiere una colaboración entre el usuario y el analista. Por ejemplo, en una aplicación que gestione un hospital, para guardar la información de cada paciente, desde siempre se emplea el término de "Historia Clínica" y resultaría confuso cambiar esta denominación por la de "Ficha del Paciente" que podría sugerir el analista.

Creación de elementos realmente significativos del sistema: por ejemplo, si tenemos que realizar una aplicación de contabilidad para una empresa determinada, es bastante normal que se adapte fielmente a las exigencias del contable de dicha empresa, lo que puede dar lugar a soluciones no validas para otras empresas. Sin embargo, si se tiene en cuenta el Plan Contable Nacional, la aplicación será válida en cualquier empresa.

Reutilización posterior del software desarrollado: por ejemplo, supongamos que queremos realizar un sistema para la gestión de una base de datos en tiempo real que necesita un tiempo de acceso que no se puede lograr con ninguna de las aplicaciones disponibles en el mercado. Para que el esfuerzo que hay que realizar no sirva solo para la aplicación concreta, lo que se debe hacer es especificar un conjunto de subrutinas de consulta y modificación que se adapten al estándar SQL. Con estas subrutinas

cualquier programa que utilice una base de datos mediante SQL podrá utilizar nuestra base de datos con un mayor tiempo de respuesta.

3.4 Análisis de requisitos software.

La etapa de análisis se encuadra dentro de la primera fase del ciclo de vida. Distinguiremos al cliente que será el encargado o encargados de elaborar junto con el analista las especificaciones del proyecto de software y de verificar el cumplimiento de las mismas.

Por ejemplo, cuando se trata de automatizar la gestión de una empresa, existe un responsable de la contabilidad, otro de los pedidos, etc. Estos son los clientes. Habrá casos donde no existirá nadie capaz de asumir el papel de cliente, bien debido a que nadie conoce exactamente lo que se requiere del sistema o bien simplemente por lo novedoso de la aplicación. En estos casos el analista debe buscar su cliente mediante una documentación exhaustiva sobre el tema.

Como se puede observar, no se asocia al cliente con la persona o entidad que financia el proyecto, aunque en ocasiones puede coincidir.

3.4.1 Objetivos del análisis

El objetivo global es obtener las especificaciones que debe cumplir el sistema a desarrollar, obteniendo un modelo valido que recoja las necesidades del cliente y las restricciones que el analista considere. En el proceso de análisis se descartan las exigencias del cliente imposibles de alcanzar o que resulten inalcanzables con los recursos puestos a disposición del proyecto.

Para que una especificación sea correcta debe tener las siguientes propiedades:

Completo y sin omisiones: tenemos que tener en cuenta que a priori no es fácil conocer todos los detalles del sistema que se pretende especificar. Por otro lado, por ejemplo , si omitimos por considerarlo sobreentendido los Sistemas Operativos o la configuración mínima que se precisara para la ejecución del sistema a desarrollar , las consecuencias pueden dar lugar a que se anule o reduzca la utilidad del sistema desarrollado finalmente.

Conciso y sin trivialidades: una documentación voluminosa suele ser una prueba de no haber sido revisada y que probablemente tenga trivialidades y repeticiones innecesarias. Por ejemplo, esto suele ocurrir cuando se elabora un nuevo modelo partiendo de otro anterior semejante. Inicialmente se mantiene todo o que no entra en contradicción con el nuevo modelo. Esto da lugar a que se mantengan párrafos del anterior que no aportan nada al nuevo modelo y que puedan dar lugar a inexactitudes.

Además, un modelo muy voluminoso no es posible estudiarlo en detalle y resulta difícil distinguir los aspectos fundamentales de aquellos que no lo son.

Sin ambigüedades: si pasamos rápidamente el análisis de requisitos para entrar de lleno en el diseño y la implementación del sistema podemos dejar en el análisis ciertos aspectos completamente ambiguos. Esto produce consecuencias negativas como, dificultades en el diseño, retrasos y errores en la implementación, etc.

Sin detalles de diseño e implementación: el objetivo del análisis es definir el QUE debe de hacer el sistema y no el COMO lo debe de hacer. Por tanto, no se debe entrar en ningún detalle del diseño o implementación del sistema en la etapa de análisis.

Fácilmente entendible por el cliente: para que el cliente este de acuerdo con el modelo fruto del análisis debe de entender el modelo para que sea capaz de discutir todos sus aspectos. Por tanto, el lenguaje que se utilice debe ser asequible para facilitar la colaboración entre el analista y el cliente.

Separando requisitos funcionales y no funcionales:

- Funcionales: serán los destinados a establecer el funcionamiento del sistema y serán el fruto de las discusiones entre analista y cliente.

- No funcionales: serán los destinados a encuadrar el sistema dentro de un entorno de trabajo, como son: capacidades mínima máxima, interfases con otros sistemas, recursos que se necesitan, seguridad, fiabilidad, mantenimiento, calidad etc.

Dividiendo y jerarquizando el modelo: para simplificar un modelo complejo se procede a dividirlo y jerarquizarlo. Esta división dará lugar a submodelos.

En la definición del modelo global del sistema, se deberá ir de lo general a lo particular.

Fijando los criterios de validación del sistema: Se realizara con carácter preliminar el Manual de Usuario del Sistema. Aunque en el Manual no se recogen todos los aspectos a validar, siempre suele ser un buen punto de partida.

3.4.2 Tarea del análisis

Para la realización correcta de un análisis de requisitos conviene dar una serie de pasos o tareas. Las tareas por su orden cronológico serán:

Estudio del sistema en su contexto: los sistemas realizados mediante software normalmente son subsistemas de otros sistemas más complejos. Por tanto, la primera tarea del análisis del sistema software será conocer el medio en el que se va a desenvolver.

Otro aspecto importante es el análisis del dominio de la aplicación que incide en la creación de sistemas con una visión más globalizadora y que facilita la reutilización posterior de alguna de sus partes.

Identificación de necesidades: el cliente tiende a solicitar del sistema todas aquellas funciones de las que en algún momento sintió la necesidad, sin pararse a pensar el grado de utilidad que tendrá en el futuro. El analista debe concretar las necesidades que se pueden cubrir con los medios disponibles y dentro del presupuesto y plazos de entrega asignados a la realización del sistema.

Se deben descartar aquellas funciones muy costosas de desarrollar y que no aportan gran cosa al sistema. Por otro lado, para las necesidades que tengan un cierto interés y que requieran más recursos de los disponibles, el analista tendrá que aportar alguna solución aunque sea incompleta, que encaje dentro de los presupuestos del sistema.

El analista debe convencer al cliente de que la solución adoptada es la mejor con los medios disponibles y nunca tratar de imponer su criterio a toda costa.

Análisis de alternativas y Estudio de viabilidad: existen infinitas soluciones para un mismo problema. El analista debe buscar la alternativa que cubra las necesidades reales detectadas y además tenga en cuenta su viabilidad tanto técnica como económica.

Si no es posible determinar un modelo que cubra todas las necesidades se deben buscar soluciones alternativas.

Establecimiento del modelo del sistema: según se van obteniendo conclusiones de las tareas anteriores, estas se deben ir plasmando en el modelo del sistema. El resultado final será un modelo del sistema global jerarquizado en el que aparecerán subsistemas que su vez tendrán que ser desarrollados hasta concretar todos los detalles del sistema.

Para la elaboración del modelo simplificar y facilitar la comunicación entre el analista, cliente y diseñador. Para ello se utilizaran todos los medios disponibles (procesadores de texto, herramientas CASE, etc.)

Elaboración del documento de especificación de requisitos: El resultado final del análisis será el documento de especificación de requisitos que servirá como punto de partida para el diseñador. Asimismo este Documento también es el encargado de fijar las condiciones de validación del sistema una vez concluido su desarrollo e implementación.

Hay que resaltar la necesidad de elaborar bien este documento para no arrastrar errores a etapas posteriores.

Revisión continuada del análisis: A lo largo del desarrollo y según aparecen problemas en el diseño y codificación, se tendrán que modificar alguno de los requisitos del sistema. Esto implica que se debe proceder a una revisión continuada del análisis y de su documento de especificación de requisitos según se producen los cambios. Si se prescinde de esta última tarea se corre el peligro de

realizar un sistema del que no se tenga ninguna especificación concreta y en consecuencia tampoco ningún medio de validar si es o no correcto el sistema finalmente desarrollado.

3.5 Notaciones para la especificación.

La especificación será fundamentalmente una descripción del modelo a desarrollar, las notaciones más frecuentes son:

3.5.1 Lenguaje natural:

Es suficiente para especificar sistemas de una complejidad pequeña, pero insuficiente para sistemas más complejos.

Sus principales inconvenientes son las imprecisiones, repeticiones e incorrecciones.

Se empleará siempre que sea necesario aclarar cualquier aspecto concreto del sistema, que no sean capaces de reflejar el resto de notaciones.

Cuando se utilice el lenguaje natural se organizaran y estructuraran los requisitos recogidos en la especificación como una cláusula de un contrato entre el analista y el cliente. Por ejemplo, se pueden agrupar los requisitos según su carácter (Funcionales, Calidad, Seguridad,…) y dentro de cada grupo se pueden numerar correlativamente las cláusulas de distintos niveles y subniveles, por ejemplo:

1. Funcionales:
   R.1.1 Modos de funcionamiento.
   R.1.2 Formatos de entrada.
   R.1.3 Formatos de salida.

El ***lenguaje natural estructurado*** es una notación más formal que el lenguaje natural, en el que se establecen ciertas reglas para la construcción de las frases que especifican acciones tipo secuencia, iteración y selección. Lo que se intenta con este lenguaje es que dentro de una misma especificación, todas las frases se construyan de igual manera.

Por ejemplo, en distintos apartados de la especificación podríamos escribir:

Cuando se teclee la clave 3 veces mal, se debe invalidar la tarjeta…

Cuando el saldo sea menor que cero pesetas se aplicara un interés…

Sin embargo sería mejor que todas las frases se construyan de igual manera, como:

SI se teclea la clave 3 veces mal ENTONCES invalidar tarjeta…

SI el saldo es menor de cero pesetas ENTONCES el interés será…

3.5.2 Diagrama de flujo de datos

Un sistema software se puede modelar mediante el flujo de datos que entran, su transformación y el flujo de datos de salida. Los símbolos que se utilizan son: (Ver Figura 3.1. Pag.72).

***Flecha*:** indica el sentido del flujo de datos. Con cada flecha se tienen que detallar sus datos mediante un nombre o una descripción.

***Círculo o burbuja*:** es un proceso o transformación de datos. Dentro del círculo se describe el proceso que realiza.

***Línea* doble:** indica un almacén de datos. Estos datos pueden ser guardados, consultados o consumidos por uno o varios procesos. Entre las dos líneas se detalla el nombre de los datos que guarda el almacén.

***Rectángulo*:** es una entidad externa al sistema software que produce o consume sus datos, por ejemplo, el usuario, otro programa, un dispositivo hardware, etc.

A esta notación la denominaremos **DFD** (Diagrama de Flujo de Datos). Al **DFD del sistema Global** también se le denomina **DFD de Contexto**.

El DFD de Contexto nunca es suficiente para describir el modelo del sistema que se trata de especificar. Para evitar esto los DFD se usan de forma jerarquizada por niveles.

DFD de contexto: Es el del sistema global, se llama de nivel 0.

DFD 0 o nivel 1: Es el resultado de explotar el DFD de contexto.

DFD 1 hasta DFD n o nivel 2: Es el resultado de la explotación de los procesos anteriores, dando lugar a otros subprocesos.

Para facilitar la identificación de los sucesivos DFD se enumeran de forma correlativa los procesos de la forma siguiente.

Nivel 0 Diagrama de Contexto.

Nivel 1 DFD 0.

Nivel 2 DFD 1 hasta DFD n,
de los procesos 1 hasta n del nivel 1.

Nivel 3 DFD 1.1 hasta DFD 1.i,
de los procesos 1.1 hasta 1.i del nivel 2.

DFD 2.1 hasta DFD 2.j,
de los procesos 2.1 hasta 2.j del nivel 2.

……

DFD n.1 hasta DFD n.k ,
de los procesos n.1 hasta n.k del nivel 2.

Nivel 4 DFD 1.1.1 hasta DFD 1.1.x

……

DFD 1.1.1 hasta DFD 1.i.z

……

…etc…

En todos ellos los flujos de entrada y salida antes de la explosión del proceso debe coincidir con los flujos de entrada y salida del DFD resultado de la explosión.

La ventaja de esta notación es su simplicidad lo que permite que sea fácilmente entendida por todos: cliente, usuario, analista, etc.

Además de esto, también es necesario describir las estructuras de cada uno de los flujos de datos y las estructuras de los datos guardados en cada uno de los almacenes de datos.

En líneas generales, se trata de describir mediante una tabla, todos los elementos básicos de información que constituyen los diversos datos que se manejan en los distintos DFD del modelo del sistema.

Los DFD presentan algunos aspectos importantes:

Sirven para establecer un modelo conceptual del sistema que facilita la estructuración de su especificación.

El modelo desarrollado es fundamentalmente estático ya que refleja los procesos necesarios para su desarrollo y la interrelación entre ellos.

Mediante un DFD nunca se puede establecer la dinámica o secuencia en que se ejecutaran los procesos.

A los DFD se le puede asociar un carácter dinámico ya que se utiliza un modelo abstracto de cómputo del tipo flujo de datos. Así, en cada ejecución se utiliza y consume un elemento de cada uno de los flujos de datos de entrada y se generan los elementos de datos para cada uno de los flujos de salida.

En el caso de los almacenes de datos, los elementos se utilizan pero no se consumen y pueden ser utilizados posteriormente.

3.5.3 Diagramas de transición de estados:

El número de estados posibles que se pueden dar en un sistema software crece de una forma exponencial con su complejidad. Todos estos estados y las sucesivas transiciones entre ellos configuran la dinámica del sistema que se produce mientras se está ejecutando.

Para especificar un sistema únicamente es necesario resaltar aquellos estados que tengan cierta trascendencia desde un punto de vista funcional.

Podemos definir un Diagrama de Transición de Estados como la notación específica para describir el comportamiento dinámico del sistema a partir de los estados elegidos como más importantes. Esta notación se complementa con los diagramas de flujo de datos y ofrece un punto de vista del sistema imprescindible en la mayoría de los casos.

Los símbolos que se utilizan son:

Estado: se suele representar mediante un rectángulo (o círculo). Se indica mediante el nombre que encierra en su interior el estado concreto que representa.

Eventos: provocan el cambio de estado. Se indican mediante una flecha dirigida desde el estado viejo al nuevo. Se emplean flechas en forma de arco cuando se utilizan círculos y formadas por tramos cuando se utiliza un rectángulo.

Estados Inicial y Final: si quieren resaltar los Estados Inicial y Final del sistema, se representan con dos círculos concéntricos.

Cuando la complejidad del sistema lo requiera se utilizaran otros diagramas de estado, además del diagrama de estados global del sistema, para especificar la dinámica de los procesos más significativos.

3.5.4 Descripciones funcionales. Pseudocódigo

Los requisitos se deberán expresar como mínimo en un lenguaje natural estructurado y a ser posible en pseudocódigo que es una notación basada en un lenguaje de programación estructurado, del que se excluyen todos los aspectos de declaración de constantes, tipos, variables y subprogramas. El Pseudocódigo maneja datos en general, no tiene una sintaxis estricta y se pueden incluir descripciones en lenguaje natural siempre que se considere necesario, como parte del pseudocódigo.

Debemos tener especial cuidado al utilizar Pseudocódigo porque si se detalla demasiado una descripción funcional se puede estar realizando el diseño del sistema e incluso la codificación. De hecho, existen notaciones similares al pseudocódigo que están pensadas para realizar el diseño como son los lenguajes de descripción de programas (PDL).

Los objetivos que persiguen Pseudocódigo y PDL son muy diferentes.

Cuando se especifica se trata de indicar cuál es la naturaleza del problema a resolver sin detallar como se debe resolver. Por ello, en una especificación no se debería establecer ninguna forma concreta de organización de la información ni tampoco ninguna propuesta concreta de los procedimientos de resolución de los problemas.

3.5.5 Descripción de datos:

Se trata de detallar la estructura interna de los datos que maneja el sistema. Solo se deben describir aquellos datos que resulten relevantes para entender QUE debe hacer el sistema.

La notación adoptada es lo que se conoce como diccionario de datos en el que se describirán:

Nombre o nombres: Los que toma el dato en la especificación.

Utilidad: Se indican los procesos, descripciones funcionales y almacenes donde se use el dato.

Estructura: Se indican los elementos de los que está constituido el dato, con arreglo a la siguiente notación:

A + B: Secuencia o concatenación de los elementos A y B.

[A | B]: Selección entre los distintos elementos A o bien B.

{A} ^{N}: Repetición de N veces del elemento A.

(A): Opcionalmente se podrá incluir el elemento A.

/descripción/: Descripción en lenguaje natural como comentarios.

=: Separador entre el nombre de un elemento y su descripción. Ver Páginas 83 y 84.

3.5.6 Diagrama de modelo de datos

El modelo de E-R (Entidad Relación) permite definir todos los datos que manejará el sistema junto con las relaciones que se desea que existan entre ellos. Como cualquier elemento de la especificación también estará sujeto a revisiones durante el diseño y codificación. La notación que se utiliza es la siguiente:

- **Entidades o datos:** se encierran dentro de un rectángulo.
- **Relaciones:** se indican mediante un rombo que encierra el tipo de relación.
- **Flechas:** las Entidades y la Relación que se establece entre ellas se indican uniéndolas todas mediante líneas. Opcionalmente, se puede indicar el sentido de la relación con una flecha.

Cardinalidad de la Relación: es entre que valores mínimos y máximos se mueve la relación entre entidades. Tiene la siguiente nomenclatura:

- **Circulo:** indica 0.
- **Raya perpendicular a la línea de relación:** indica 1.
- **Tres Rayas en Bifurcación:** indica muchos o N.

La cardinalidad se dibuja siempre unida a la segunda entidad de la relación con un símbolo para el valor mínimo y otro para el máximo.

3.6 Documento de especificación de requisitos.

El SRD (Software Requirements Document) o SRS (Software Requirements Specification) es un documento fundamental que recogerá todo lo realizado durante la etapa de análisis y servirá como punto de partida para cualquier sistema software.

La mejor manera de redactarlo es en forma de contrato con sus cláusulas organizadas según el carácter de los requisitos para facilitar la revisión y la verificación. Sus partes son:

3.6.1.1 Introducción

Esta sección debe dar una visión general de todo el documento SRD.

- **Objetivo:** expone:
  - El objetivo del proyecto (brevemente).
  - A quien va dirigido.
  - Los participantes.
  - El calendario previsto.
- **Ámbito:**
  - Se identifica y nombra el producto resultante del Proyecto.
  - Se explica qué hace el producto obtenido.

- Se explica que NO será capaz de hacer el producto (si se considera necesario).
- Se detallan las posibles aplicaciones y beneficios del Proyecto.

- **Definiciones, siglas y abreviaturas:** se incluirá un glosario con las definiciones de términos, siglas y abreviaturas particulares utilizados en el documento y que convenga reseñar para facilitar su lectura o evitar ambigüedades. Esta información también podrá figurar como apéndice al final del documento.

- **Referencias:** si el documento contiene referencias a otros documentos, se detallará la descripción bibliográfica de los documentos referenciados y la manera de acceder a los mismos. Esta información también podrá figurar como apéndice al final del documento.

- **Panorámica del documento:** se describe la organización y el contenido del resto del documento.

3.6.1.2 Descripción general

Se da una visión general del sistema, ampliando el contenido de la sección de introducción.

Relación con otros proyectos: se describen las analogías y diferencias de este proyecto con otros similares o complementarios, o con otros sistemas ya existentes o en desarrollo. Si no hay proyectos o sistemas relacionados, se indicará "No aplicable".

Relación con otros proyectos anteriores y posteriores: se indica si este proyecto es continuación de otro o si se continuará el desarrollo en proyectos posteriores. Si no hay proyectos o sistemas relacionados, se indicará "No aplicable".

Objetivo y funciones: se describe el sistema con los objetivos y funciones principales.

Consideraciones de entorno: se describen las características especiales que debe tener el entorno en que se utilice el sistema a desarrollar. Si no hay características especiales, se indicará "No existen".

Relaciones con otros sistemas: se describen las conexiones del sistema con otros sistemas, si debe funcionar integrado con ellos o utilizando entradas o salidas indirectas de información. Si el sistema no necesita intercambiar información con ningún otro, se indicará "No existen".

Restricciones generales: se describen las restricciones generales a tener en cuenta en el diseño y desarrollo del sistema, tales como el empleo de determinadas metodologías de desarrollo, lenguajes de programación, normas particulares, restricciones hardware, de sistema operativo, etc.

Descripción del modelo: se describe el Modelo Conceptual propuesto para desarrollar el sistema en su conjunto y para cada una de sus partes más relevantes. Este modelo se puede realizar utilizando todas las notaciones y herramientas disponibles.

3.6.1.3 Requisitos específicos

Es la sección fundamental del documento. Debe contener completa y detalladamente los requisitos que debe cumplir el sistema a desarrollar.

Es importante no incluir aspectos de diseño o desarrollo. Cada requisito debe ir acompañado del grado de cumplimiento necesario, es decir, si es obligatorio, recomendable u opcional.

Si en algún apartado no hay requisitos, se indicará "No existe".

Requisitos funcionales: describen las funciones o el QUÉ debe hacer el sistema y están muy ligados al modelo conceptual propuesto. Se concretarán las operaciones de tratamiento de información, generación de informe, cálculos estadísticos, operaciones, etc.

Requisitos de capacidad: son los referentes a los volúmenes de información a procesar, tiempo de respuesta, tamaño de ficheros o discos, etc. Se deben expresar mediante valores numéricos e incluso cuando sea necesario, se darán valores para el peor, el mejor y el caso más habitual.

Requisitos de interfase: son los referentes a cualquier conexión a otros sistemas (hardware o software) con los que se debe interactuar o comunicar. Se incluirán por tanto, bases de datos,

protocolos, formatos de ficheros, sistemas operativos, datos, etc. a intercambiar con otros sistemas o aplicaciones.

Requisitos de operación: son los referentes al uso del sistema en general e incluyen los requisitos de la interfase de usuario (menús de pantalla, manejo de ratón, teclado, etc), el arranque y parada, copias de seguridad, requisitos de instalación y configuración.

Requisitos de recursos: son los referentes a elementos hardware, software, instalaciones, etc. necesarios para el funcionamiento del sistema. Se deben estimar los recursos con cierto coeficiente de seguridad en previsión de necesidades de última hora no previstas inicialmente.

Requisitos de verificación: son los que debe cumplir el sistema para que sea posible verificar y certificar que funciona correctamente (funciones de autotest, emulación, simulación, etc.).

Requisitos de pruebas de aceptación: son los que debe cumplir las pruebas de aceptación a que se someterá el sistema.

Requisitos de documentación: son los referentes a la documentación que debe formar parte del producto a entregar.

Requisitos de seguridad: son los referentes a la protección del sistema contra la manipulación indebida (confidencialidad, integridad, virus, etc).

Requisitos de transportabilidad: son los referentes a la posible utilización del sistema en diversos entornos o sistemas operativos de una forma sencilla y barata.

Requisitos de calidad: son los referentes a aspectos de calidad que no se incluyan en otros apartados.

Requisitos de fiabilidad: son los referentes al límite aceptable de fallos o caídas durante la operación del sistema.

Requisitos de mantenibilidad: son los que debe cumplir el sistema para que se pueda realizar adecuadamente su mantenimiento durante la fase de explotación.

Requisitos de salvaguarda: son los que debe cumplir el sistema para evitar que los errores en el funcionamiento o la operación del sistema tengan consecuencias graves en los equipos o las personas.

3.6.1.4 Apéndices

Se incluirán todos aquellos elementos que completen el contenido del documento, y que no estén recogidos en otros documentos accesibles a los que pueda hacerse referencia.

Ver ejemplos páginas 95 y 100.