

TEMA 2 ESPECIFICACIÓN DE SOFTWARE

2.1 MODELADO DE SISTEMAS

El modelado de los sistemas realizados mediante software también tiene como objetivo entender mejor el funcionamiento requerido y facilitar la comprensión de los problemas planteados.

El modelo, es completo y preciso del comportamiento u organización del sistema software.

Existen diversas metodologías para realizar el análisis de los requisitos que debe cumplir un proyecto software. Todas tienen un aspecto común: facilitar la obtención de uno o varios modelos que detallen el comportamiento deseado del sistema.

2.1.1 Concepto de modelo

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.

El modelo de un sistema software establece las propiedades y restricciones del sistema. Visión de alto nivel. El modelo debe explicar QUÉ debe hacer el sistema y no CÓMO lo debe de hacer.

Objetivos que se deben cubrir con los modelos:

- Facilitar la comprensión del problema a resolver.
- Establecer un marco para la discusión, que simplifique y sistematice tanto la labor del análisis inicial como de las futuras revisiones del mismo.
- Fijar las bases para realizar el diseño.
- Facilitar la verificación del cumplimiento de los objetivos del sistema.

2.1.2 Técnicas de modelado

- **DESCOMPOSICIÓN. MODELO JERARQUIZADO.**

La primera técnica que se debe usar con un problema complejo es descomponerlo en otros más sencillos. Divide y vencerás.

Se establece un modelo jerarquizado, en el que el problema global queda subdividido en un cierto número de subproblemas.

1. Descomposición horizontal. Descomponer funcionalmente el problema.

Ejemplo: Un sistema integral para una empresa, se puede descomponer.

Sistema de nóminas, sistema de contabilidad, sistema de facturación, etc.

2. Descomposición vertical. Descomponer tratando de detallar su estructura.

Entrada datos del trabajador, calculo de ingresos brutos, etc.

Es posible descomponer cada uno de estos subsistemas en otros más simples.

Para completar el modelo, se deben establecer las interfases entre las partes o subsistemas.

Se debe aplicar el método de refinamientos sucesivos al modelado del sistema.

- **APROXIMACIONES SUCESIVAS**

Es corriente que el sistema software que se quiera modelar sustituya a un proceso de trabajo ya existente, que se realiza de forma totalmente manual o con un grado de automatización menor del que se pretende lograr con el sistema nuevo.

Crear un modelo de partida basado en la forma de trabajo anterior, será un modelo sólo preliminar y tendrá que ser depurado mediante aproximaciones sucesivas hasta alcanzar un modelo final.

Es importante contar con la colaboración de alguien que conozca muy bien el sistema anterior y que sea capaz de incorporar mejoras dentro del nuevo sistema y discutir las ventajas e inconvenientes de cada uno de los modelos intermedios elaborados.

- **EMPLEO DE DIVERSAS NOTACIONES**

Una posible notación para describir el modelo de un sistema es mediante el lenguaje natural, este método puede dar lugar a un modelo difícil de apreciar en su conjunto por lo farragoso de las explicaciones. Se producirán imprecisiones, repeticiones e incluso incorrecciones en el modelo así realizado.

Lo ideal es emplear la notación con la que mejor se cubran los objetivos de un modelo. Es aconsejable emplear varias notaciones juntas cuando sea necesario.

Existen herramientas de modelado para la ayuda al análisis y diseño de los sistemas, herramientas CASE (Computer Aided Software Engineering)

- **CONSIDERAR DISTINTOS PUNTOS DE VISTA**

El modelo estará necesariamente influenciado por el punto de vista adoptado. Elegir el punto de vista que permita obtener el modelo más adecuado para representar el comportamiento deseado del sistema.

1. Punto de vista del usuario del sistema.
2. Punto de vista del mantenedor del sistema.
3. Punto de vista funcional.
4. Etc....

En ocasiones el modelo no quedará completamente definido con un solo punto de vista. Lo adecuado será realizar el modelo desde distintos puntos de vista.

- **REALIZAR UN ANÁLISIS DEL DOMINIO**

Dominio es el campo de aplicación donde se encuadra el sistema a desarrollar.

Es importante analizar el dominio de la aplicación para situarla dentro de un entorno mucho más global. Es aconsejable estudiar los siguientes aspectos:

1. Normativa que afecte al sistema.
2. Otros sistemas semejantes.
3. Estudios recientes en el campo de la aplicación.
4. Bibliografía clásica y actualizada.
5. Etc...

Este estudio facilitará la creación de un modelo más universal. Ventajas de este enfoque:

1. Facilitar la comunicación entre analista y usuario del sistema.
2. Creación de elementos realmente significativos del sistema. Cuando se ignora el dominio de una aplicación, la solución queda particularizada en exceso.
3. Reutilización posterior del software desarrollado.

2.2 ANÁLISIS DE REQUISITOS DE SOFTWARE

La etapa de análisis se encuadra dentro de la primera fase (fase de definición) del ciclo de vida. El análisis de requisitos trata de caracterizar el problema a resolver. Se puede concretar en la obtención del modelo global que define por completo el comportamiento requerido del sistema.

Un problema que se le plantea al analista es conseguir un interlocutor válido para poder llevar a cabo el análisis. Este interlocutor, denominado de forma genérica cliente, puede estar constituido por una o varias personas expertas en todo o sólo en una parte del trabajo que se pretende automatizar.

Hay casos en los que no existirá nadie capaz de asumir el papel de cliente, debido a que nadie conoce exactamente lo que se requiere del sistema o bien simplemente por lo novedoso de la aplicación.

No se debe de asociar cliente con la persona o entidad que financia el proyecto. El cliente será el encargado de elaborar junto con el analista las especificaciones del proyecto de software y que posteriormente se encargarán de verificar el cumplimiento de las mismas, como si de un contrato se tratara.

En ocasiones el cliente será el usuario final de la aplicación, en otras coincidirá con el cliente que realmente financia el proyecto, en otros puede ser simplemente parte de una especificación de otro proyecto de mayor envergadura en el que está encuadrado el nuestro.

2.2.1 OBJETIVOS DEL ANÁLISIS

El objetivo global del análisis es obtener las especificaciones que debe cumplir el sistema a desarrollar.

Para lograr este objetivo, se debe de obtener un modelo válido, necesario y suficiente donde recoger todas las necesidades y exigencias que el cliente precisa del sistema y además todas aquellas restricciones que el analista considere que debe de verificar el sistema. Las especificaciones se obtendrán basándose en el modelo obtenido.

Quedarán descartadas aquellas exigencias del cliente que resulten imposibles de lograr con los recursos puestos a disposición del proyecto.

Quedarán convenientemente matizadas cada una de las exigencias y necesidades del sistema para adaptarlas a los medios disponibles para el proyecto.

El modelo global del sistema deberá tener las siguientes propiedades:

1. Completo y sin omisiones.
2. Conciso y sin trivialidades.
3. Sin ambigüedades.

El modelo resultado del análisis es un contrato con el cliente. Nada debe de quedar ambiguo o de lo contrario se producirán fricciones y problemas de consecuencias difíciles de prever.

4. Sin detalles de diseño o implementación.

El objetivo del análisis es definir QUÉ debe hacer el sistema y no el CÓMO lo debe de hacer. No se debe entrar en ningún detalle del diseño o implementación del sistema en la etapa de análisis.

5. Fácilmente entendible por el cliente.

Importante usar un lenguaje que sea asequible y que facilite la colaboración entre analista y cliente. Emplear notaciones gráficas.

6. Separar requisitos funcionales y no funcionales.

- Requisitos funcionales. Destinados a establecer el modelo de funcionamiento del sistema y serán el fruto fundamental de las discusiones entre analista y cliente.
- Requisitos no funcionales. Destinados a encuadrar el sistema dentro de un entorno de trabajo. Tienen un origen más técnico y no tienen tanto interés para el cliente. Deben permanecer separados en el modelo del sistema.
 - Capacidades mínima y máxima.
 - Interfases con otros sistemas.
 - Recursos que se necesitan.
 - Aspectos de seguridad, fiabilidad, mantenimiento, calidad.
 - Etc.

7. Dividiendo y jerarquizando el modelo.

Simplificar un modelo complejo del sistema es dividirlo y jerarquizarlo. Esta división dará lugar a submodelos. En el modelo global del sistema se debe de ir de lo general a lo particular.

8. Fijando los criterios de validación del sistema.

Es importante que en el modelo del sistema queden indicados cuáles serán los criterios de validación del sistema. Un método para fijar los criterios de validación es realizar con carácter preliminar el Manual de Usuario del sistema, aunque en dicho manual no se recogen todos los aspectos a validar, es un buen punto de partida.

2.2.2 TAREAS DEL ANÁLISIS

Pasos para la realización correcta de un análisis de requisitos:

1. Estudio del sistema en su contexto.

Conocer el medio en el que se va a desenvolver el sistema.

Importante, el análisis del dominio de la aplicación. Incide directamente no sólo en la terminología a emplear en la especificación, sino también en la creación de sistemas con una visión más globalizadora y que facilita la reutilización posterior de alguna de sus partes.

2. Identificación de necesidades.

La labor del analista es 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.

Fundamental mantener un dialogo constante y fluido con el cliente. El analista deberá recoger y estudiar sus sugerencias para elaborar una propuesta que satisfaga a todos. Descartando funciones muy costosas de desarrollar y que no aportan gran cosa al sistema. Para aquellas funciones que tengan un cierto interés y que requieran más recursos de los disponibles, el analista deberá aportar una solución que aunque sea incompleta, encaje dentro de los presupuestos del sistema.

Las necesidades que finalmente se identifiquen deberán estar consensuadas por todos aquellos que junto al analista hayan participado en el análisis. El analista tratara de convencer a todos de que la solución adoptada es la mejor con los medios disponibles, nunca tratara de imponer su criterio.

3. Análisis de alternativas. Estudio de viabilidad.

Existen infinitas soluciones a un mismo problema. Es labor del analista buscar aquella alternativa que cubra las necesidades reales detectadas y que tenga en cuenta su viabilidad tanto técnica como económica.

Con cada alternativa presentada es necesario realizar su estudio de viabilidad correspondiente.

4. Establecimiento del modelo del sistema.

El modelo se irá perfilando a medida que se avanza en el estudio de las necesidades y las soluciones adoptadas.

Para la elaboración del modelo se utilizarán todos los medios disponibles, procesadores de texto, herramientas CASE, herramientas gráficas, y todo aquello que contribuya a facilitar la comunicación entre analista, cliente y diseñador.

5. Elaboración del documento de especificación de requisitos.

El resultado final del análisis es el **documento de especificación de requisitos**. Debe de recoger todas las conclusiones del análisis.

Una parte fundamental del documento será el modelo del sistema y es en este documento donde se debe ir perfilando su elaboración a lo largo de todo el análisis.

Este documento será el que utilice el diseñador como punto de partida de su trabajo.

Este documento es el encargado de fijar las condiciones de validación del sistema una vez concluido su desarrollo e implementación.

6. Revisión continuada del análisis.

La labor de análisis no acaba al dar por finalizada la redacción del documento de especificación de requisitos. A lo largo del desarrollo del sistema y según aparezcan problemas en las etapas de diseño e implementación se tendrán que modificar alguno de los requisitos del sistema. También se suelen modificar requisitos debido a cambios de criterio del cliente.

2.3 NOTACIONES PARA LA ESPECIFICACIÓN

La especificación es una descripción del modelo del sistema que se pretende desarrollar. Para un mismo modelo conceptual, dependiendo de la notación o notaciones que se empleen para su descripción, se obtendrán distintas especificaciones.

El empleo de la notación más adecuada en cada caso redundará en una mejor especificación del sistema. Será preferible usar una notación gráfica que el texto que la pueda describir.

Notaciones más usadas para especificar sistemas software:

- **Lenguaje natural**

Para sistemas de una complejidad pequeña, es suficiente el lenguaje natural.

Cuando la complejidad del sistema es mediana o grande, resulta prácticamente imposible utilizar sólo el lenguaje natural.

El lenguaje natural se usará siempre que sea necesario aclarar cualquier aspecto concreto del sistema, que no sean capaces de reflejar el resto de notaciones.

Siempre que se use el lenguaje natural es muy importante organizar y estructurar los requisitos recogidos en la especificación.

La mejor forma, es concebir cada uno de los requisitos como una cláusula de un contrato entre el analista y el cliente.

Se pueden agrupar los requisitos según su carácter:

1. Funcionales
2. Calidad
3. Seguridad
4. Fiabilidad
5. ...

Y dentro de cada grupo, establecer y numerar correlativamente las cláusulas de distintos niveles y subniveles que estructuren los requisitos.

1. Funcionales
 - 1.1 Modos de funcionamiento.
 - 1.2 Formatos de entrada.
 - 1.3 Formatos de salida.
 - 1.4 ...

El lenguaje natural estructurado. Notación más formal que el lenguaje natural. Establece ciertas reglas para la construcción de las frases en las que se especifica una acción de tipo secuencia, condición o iteración.

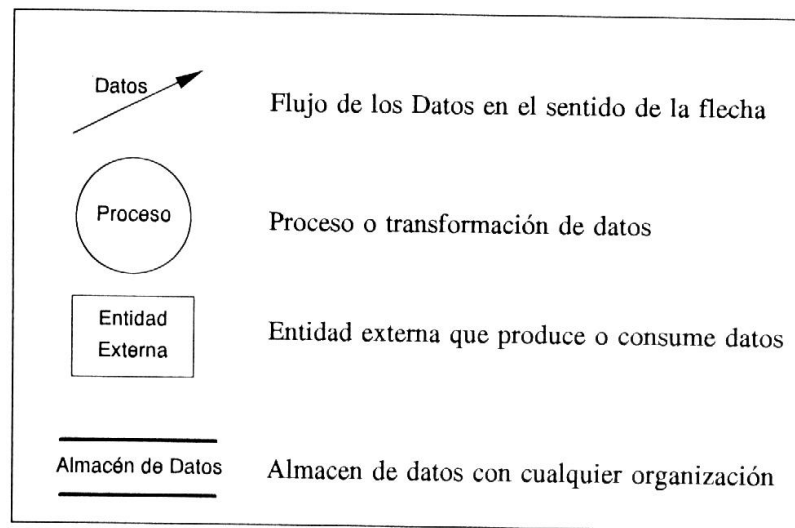
No se trata de usar siempre una misma palabra clave (SI, REPETIR, etc.) sino emplear la misma construcción en todas las frases, al menos, de una misma especificación.

- **Diagramas de flujo de datos (DFD).**

Notación asociada a la metodología de análisis estructurado, cuyo enfoque es considerar que un sistema software se puede modelar mediante el flujo de datos que entra al sistema, las transformaciones que se realizan con los datos de entrada y el flujo de datos que se produce a la salida como resultado de la transformación.

En los diagramas de flujo de datos (DFD) se pueden modelar las transformaciones y los flujos de datos.

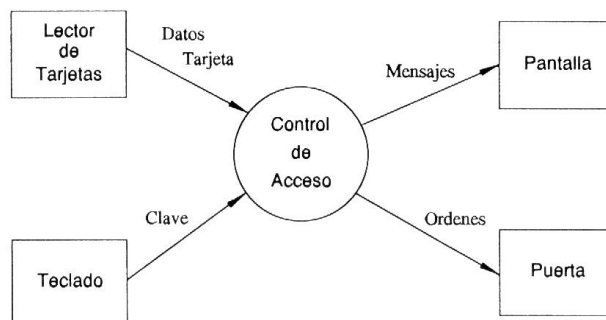
Notación de los DFD



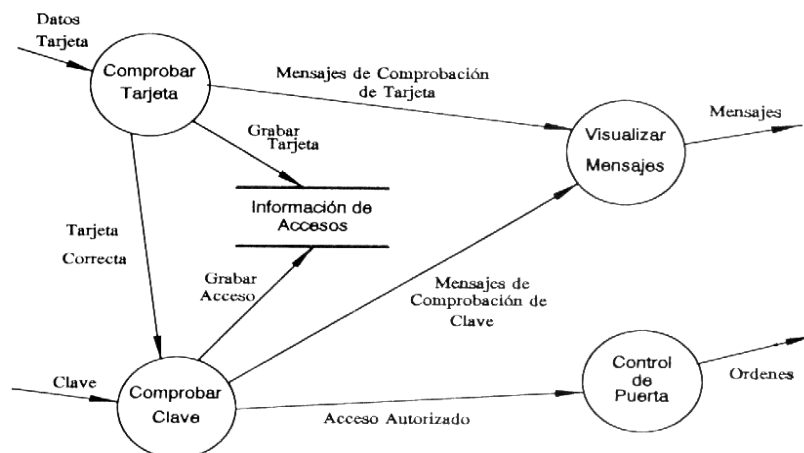
Los DFD se usan de forma jerarquizada por niveles.

El DFD de contexto se denomina nivel 0, cada proceso o burbuja se puede explotar y mostrar cómo se organiza interiormente, mediante otros DFD. En el DFD de contexto, sólo se puede explotar un único proceso que es el proceso global del sistema y su explosión dará lugar al único DFD de nivel 1.

Ejemplo de DFD global del sistema



Ejemplo de DFD de nivel 1



La identificación de los sucesivos DFD se realiza numerando de forma correlativa los distintos procesos antes de su refinamiento.

El DFD resultante de la explosión de un proceso x tendrá el número x del proceso explotado. Los procesos hijos que aparecen en él se numerarán de 1 en adelante, en la forma x.1, x.2, ...

El DFD de contexto no lleva numeración y el único diagrama resultante de nivel 1 se designa como DFD 0.

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.i.1 hasta DFD 1.i.z etc

Todos los flujos de datos de entrada y salida antes de la explosión del proceso deben coincidir con los flujos de entrada y salida del DFD resultado de la explosión o refinamiento.

La ventaja de esta notación es su simplicidad, fácilmente entendible por todos.

Los DFD sirven para establecer un modelo conceptual del sistema que facilita la estructuración de su especificación. El modelo así desarrollado es fundamentalmente estático dado que refleja los procesos necesarios para su desarrollo y la interrelación que existe entre ellos.

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

La única premisa de carácter dinámico que se puede establecer en un DFD es que en ellos se utiliza un modelo abstracto de cómputo del tipo flujo de datos.

- **Diagramas de transición de estados.**

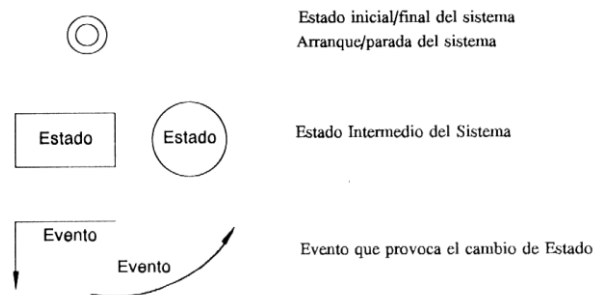
Cada vez que se modifica una variable, se evalúa una condición o el término de una expresión se produce un cambio de estado en el sistema.

Estos estados y las sucesivas transiciones entre ellos configuran la dinámica del sistema que se produce mientras se está ejecutando.

El diagrama de transición de estados es la notación específica para describir el comportamiento dinámico del sistema a partir de los estados elegidos como más importantes.

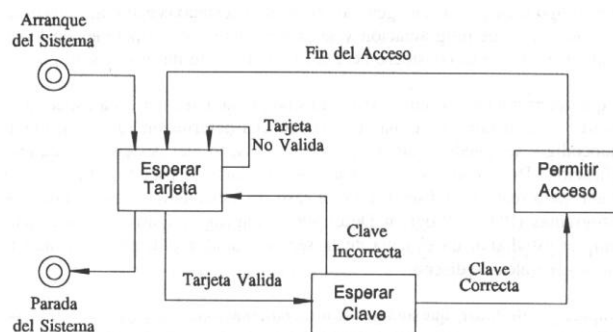
Se complementa perfectamente con los diagramas de flujo de datos y ofrece otro punto de vista del sistema.

Notación de diagramas de estado.



Cuando la complejidad así lo requiera, serán necesarios varios diagramas de estados.

Ejemplo



- **Descripciones funcionales. Pseudocódigo.**

Pseudocódigo. Notación basada en un lenguaje de programación estructurado del que se excluyen todos los aspectos de declaración de constantes, tipos, variables y subprogramas.

Se pueden incluir descripciones en lenguaje natural siempre que se considere necesario, como parte del pseudocódigo.

Si se detalla o refina demasiado una descripción funcional mediante pseudocódigo se puede estar realizando el diseño del sistema e incluso su codificación.

Cuando se especifica se trata de indicar cual es la naturaleza del problema a resolver sin detallar cómo se debe resolver.

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.

Estructuras básicas de pseudocódigo:

- Selección.
SI <Pseudocódigo de la condición> ENTONCES
 <Pseudocódigo>
SI-NO
 <Pseudocódigo>
FIN-SI
- Selección por casos
CASO <Especificación del elemento selector>
 SI-ES <Descripción del caso 1> HACER <Pseudocódigo>;
 SI-ES <Descripción del caso 2> HACER <Pseudocódigo>;
 ...
 SI-ES <Descripción del caso N> HACER <Pseudocódigo>;
 OTROS <Pseudocódigo>
FIN-CASO
- Iteración con pre-condición
MIENTRAS <Pseudocódigo de la condición> HACER
 <Pseudocódigo>
FIN-MIENTRAS
- Iteración con post-condición
REPETIR
 <Pseudocódigo>
HASTA <Pseudocódigo de la condición>
- Número de iteraciones conocido
PARA CADA <Especificación del elemento índice> HACER
 <Pseudocódigo>
FIN-PARA

- **Descripción de datos.**

Trata de detallar la estructura interna de los datos que maneja el sistema. No se debe descender a detalles de diseño o codificación.

En la metodología de análisis estructurado es lo que se conoce como **diccionario de datos**.

Existen diversos formatos posibles de diccionario de datos, pero todos deben de recoger al menos la siguiente información para cada dato:

- **Nombre o nombres.**
Denominación con la que se utilizará este dato en el resto de la especificación.
- **Utilidad**
Indicará todos los procesos, descripciones funcionales, almacenes de datos, etc. En los que se utilizará el dato.
- **Estructura**
Indicará los elementos de los que está constituido el dato, utilizando la siguiente notación:

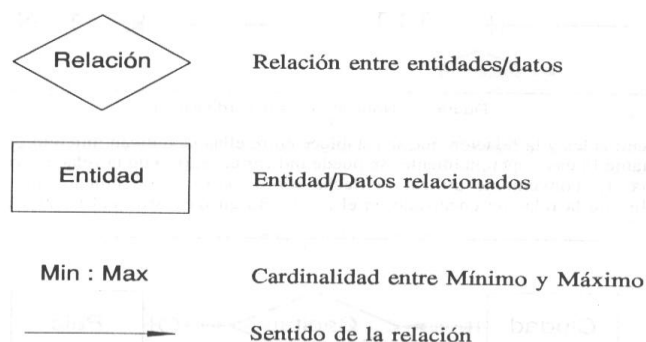
A+B	Secuencia de concatenación de los elementos A y B
[A B]	Selección entre los distintos elementos A o bien B
{A} ^N	Repetición N veces del elemento A. Se omite N si es indeterminado.
(A)	Opcionalmente se podrá incluir el elemento A
/ descripción /	Descripción en lenguaje natural como comentarios.
=	Separador entre el nombre de un elemento y su descripción

- **Diagramas de modelo de datos.**

Este modelo se conoce como modelo entidad-relación (modelo E-R) y permite definir todos los datos que manejará el sistema junto con las relaciones que se desea que existan entre ellos.

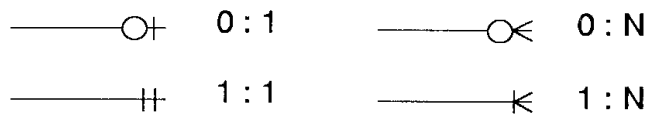
El modelo estará sujeto a revisiones durante las fases de diseño y codificación. Es un punto de partida imprescindible para comenzar cualquier diseño.

Notación básica de diagramas de datos.



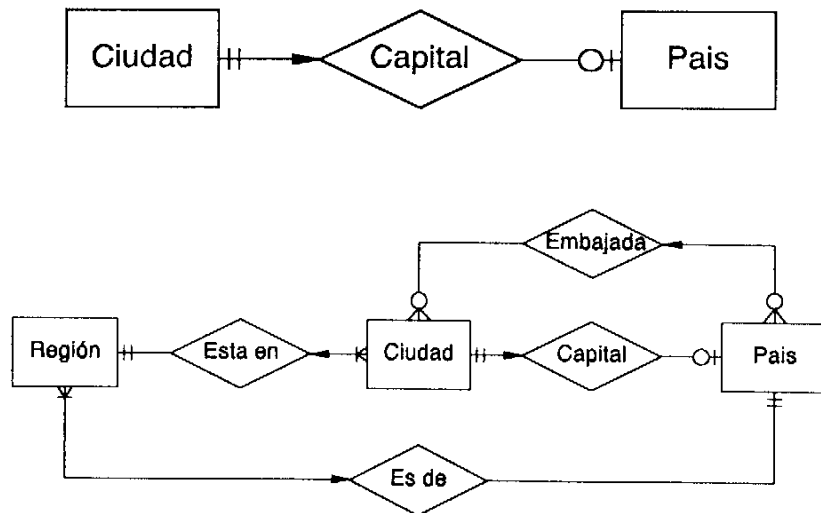
En los diagramas E-R la cardinalidad se muestra con la siguiente notación.

Cardinalidad



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

Ejemplos



2.4 DOCUMENTO DE ESPECIFICACIÓN DE REQUISITOS (SRD)

Software Requirements Document (SRD). Encargado de recoger todo el fruto del trabajo realizado durante la etapa de análisis del sistema de una forma integral en un único documento.

Documentos previos al SRD:

- Estudios de viabilidad.
- Estudios de alternativas.
- Historia del proyecto..
-

Todos estos documentos tienen una utilidad muy concreta pero no son imprescindibles.

El SRD es un documento fundamental y es el punto de partida del desarrollo de cualquier sistema software.

La mejor manera de redactar este documento es en forma de contrato con sus distintas cláusulas organizadas y agrupadas según el carácter de los requisitos.

1. INTRODUCCIÓN

- 1.1 Objetivo
- 1.2 Ámbito
- 1.3 Definiciones, siglas y abreviaturas
- 1.4 Referencias
- 1.5 Panorámica del documento

2. DESCRIPCIÓN GENERAL

- 2.1 Relación con otros proyectos
- 2.2 Relación con proyectos anteriores y posteriores
- 2.3 Objetivo y funciones
- 2.4 Consideraciones de entorno
- 2.5 Relaciones con otros sistemas
- 2.6 Restricciones generales
- 2.7 Descripción del modelo

3. REQUISITOS ESPECÍFICOS

- 3.1 Requisitos funcionales
- 3.2 Requisitos de capacidad
- 3.3 Requisitos de interfase
- 3.4 Requisitos de operación
- 3.5 Requisitos de recursos
- 3.6 Requisitos de verificación
- 3.7 Requisitos de pruebas de aceptación
- 3.8 Requisitos de documentación
- 3.9 Requisitos de seguridad
- 3.10 Requisitos de transportabilidad
- 3.11 Requisitos de calidad
- 3.12 Requisitos de fiabilidad
- 3.13 Requisitos de mantenibilidad
- 3.14 Requisitos de salvaguarda

4. APÉNDICES

1. INTRODUCCIÓN

Visión general de todo el documento SRD.

1.1 Objetivo

Exponer brevemente el objetivo del proyecto, a quién va dirigido, participantes y el calendario previsto.

1.2 Ámbito

Identificará y dará nombre al producto o los productos resultantes del proyecto.

Se explicará que hace y que no hace cada uno.

Se detallarán de manera precisa las posibles aplicaciones y beneficios del proyecto.

1.3 Definiciones, siglas y abreviaturas.

Glosario que contendrá una lista de definiciones de términos, siglas y abreviaturas particulares utilizados en el documento, que convenga reseñar para facilitar su lectura o evitar ambigüedades.

1.4 Referencias

Se dará una lista con la descripción bibliográfica de los documentos referenciados.

1.5 Panorámica del documento

Describirá la organización y el contenido del resto del documento.

2. DESCRIPCIÓN GENERAL

Visión general del sistema, ampliando el contenido de la sección de introducción.

2.1 Relación con otros proyectos

Analogías y diferencias de este proyecto con otros similares o complementarios.

Si no hay proyectos o sistemas relacionados, se indicará "No Aplicable"

2.2 Relación con proyectos anteriores y posteriores

Indicará si este proyecto es continuación de otro o si se continuará el desarrollo en proyectos posteriores.

Si no hay proyectos de esta clase, se indicará "No existen."

2.3 Objetivo y funciones

Describir el sistema en su conjunto con los objetivos y las funciones principales.

2.4 Consideraciones de entorno

Se describirán las características especiales que debe tener el entorno en que se utilice el sistema a desarrollar.

Si no se necesitan características especiales, se indicará "No existen".

2.5 Relaciones con otros sistemas

Describirá las conexiones del sistema con otros, si debe funcionar integrado en ellos o utilizando entradas y salidas indirectas de información.

Si el sistema no necesita intercambiar información con ningún otro, se indicará “No existen”.

2.6 Restricciones generales

Restricciones generales a tener en cuenta a la hora de diseñar y desarrollar el sistema.

- Empleo de determinadas metodologías.
- Lenguajes de programación.
- Normas particulares
- Restricciones hardware
- Restricciones del sistema operativo
- Etc..

2.7 Descripción del modelo

Probablemente el apartado más extenso de esta sección.

Describir el modelo conceptual que se propone para desarrollar el sistema en su conjunto y para cada una de sus partes más relevantes.

Se puede realizar utilizando todas las notaciones y herramientas.

3. REQUISITOS ESPECÍFICOS

Sección fundamental del documento.

Debe de contener una lista detallada y completa de los requisitos que debe cumplir el sistema a desarrollar.

Exponer de la forma más precisa posible, sin que la descripción de un requisito individual resulte demasiado extensa.

Importante: No incluir en los requisitos aspectos de diseño o desarrollo.

Enunciar los requisitos en forma de una lista numerada. Cada requisito debe ir acompañado de una indicación del grado de cumplimiento necesario.

- Obligatorio
- Recomendable
- Opcional

Los requisitos se agruparán en los apartados que se indican a continuación.

Si no hay requisitos en alguno de ellos, se indicará “No existen”.

3.1 Requisitos funcionales

Describen las funciones o QUÉ debe hacer el sistema y están muy ligados al modelo conceptual propuesto.

Concretar las operaciones de tratamiento de información, almacenamiento de información, generación de informes, cálculos, estadísticas, operaciones, etc.

3.2 Requisitos de capacidad

Referentes a los volúmenes de información a procesar, tiempo de respuesta, tamaños de ficheros o discos, etc.

Deben expresarse mediante valores numéricos. Cuando sea necesario se darán valores para el peor, el mejor y el caso más habitual.

3.3 Requisitos de interfase.

Referentes a cualquier conexión a otros sistemas. Incluye:

- Bases de datos
- Protocolos
- Formatos de ficheros
- Sistemas operativos
- Datos a intercambiar con otros sistemas o aplicaciones

3.4 Requisitos de operación

Referentes al uso del sistema en general.

3.5 Requisitos de recursos

Referentes a hardware, software, instalaciones, etc. Necesarios para el funcionamiento del sistema.

Importante, estimar los recursos con cierto coeficiente de seguridad en previsión de necesidades de última hora no previstas.

3.6 Requisitos de verificación

Los que debe cumplir el sistema para que sea posible verificar y certificar que funciona correctamente. Funciones de autotest, emulación, simulación, etc.

3.7 Requisitos de pruebas de aceptación

Los que deben cumplir las pruebas de aceptación a que se someterá el sistema.

3.8 Requisitos de documentación

Documentación que debe formar parte del producto a entregar

3.9 Requisitos de seguridad

Referentes a la protección del sistema contra cualquier manipulación o utilización.

3.10 Requisitos de transportabilidad

Referentes a la posible utilización del sistema en diversos entornos o sistemas operativos de una forma sencilla y barata.

3.11 Requisitos de calidad

Referentes a aspectos de calidad.

3.12 Requisitos de fiabilidad

Referentes al límite aceptable de fallos o caídas durante la operación del sistema.

3.13 Requisitos de mantenibilidad

Los que debe cumplir el sistema para que se pueda realizar adecuadamente su mantenimiento durante la fase de explotación.

3.14 Requisitos de salvaguarda

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 personas.

4. APÉNDICES

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 de especificaciones.