Bibliografía

Obligatoria (en orden alfabético)

- Elmasri, R. y Navathe, S.; **Sistemas de Base de Datos**. 2ª ed. USA: Addison-Wesley Iberoamericana; 1997
- Neil, C. G.; **Análisis de Sistemas – un enfoque conceptual**-. Buenos Aires, UAI, 2002.
- Neil, C. G.; **Modelo Entidad Interrelación –un caso práctico-.** UAI, 2005.
- R. S. Pressman. **Ingeniería de Software. Un enfoque Práctico**. 4ª ed. México: Mc Graw-Hill; 1998
- Yourdon, E.; **Análisis Estructurado Moderno**. México: Prentice-Hall Hispanoamericana; 1993.

Ampliatoria

- Batini, Ceri y Navathe; **Diseño Conceptual de Base de Datos – un enfoque de entidades interrelaciones.** Wilmington, Del. Addison-Wesley: 1994
- Date, C.; **Introducción a los Sistemas de Base de Datos**. Volumen 1. 5ª ed. USA: Addison-Wesley Iberoamericana; 1993

Dirección electrónica

http://biblioteca.vaneduc.edu.ar

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 2

Datos e Información

Los gerentes utilizan a la información para realizar sus actividades diarias de toma de decisiones. Las fuentes de información con las que se nutren, tanto como los formatos de su representación, pueden ser variados. En el proceso diario de las organizaciones, los datos se transforman en información cuando son interpretados y procesados por el usuario en la toma de decisión.

Es importante diferenciar datos de información. Los usuarios requieren información para la toma de decisiones; los diseñadores precisan tener la libertad de proponer distintas estructuras de datos para hacerlos persistentes en el tiempo sin redundancia y que, una vez recuperados, brinden la información solicitada por el usuario.

En los sistemas de información, la comunicación permite interrelacionar a los componentes para el logro de los objetivos propuestos. Para diseñar sistemas de información se deben conocer los fundamentos de los conceptos de datos, información y comunicación.

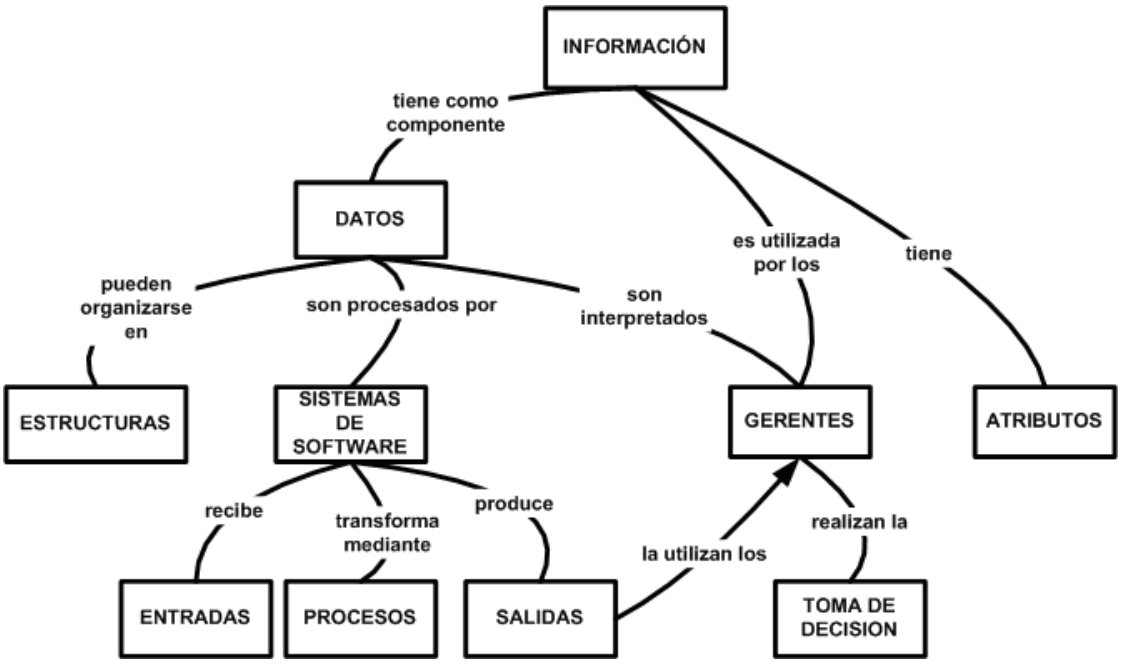


Figura 2.1: Mapa conceptual de “Datos e Información”

2.1. LOS DATOS

Los datos son símbolos que describen algo, éstos se organizan en estructuras para describir un objeto. Todas las aplicaciones de software pueden, conjuntamente, denominarse procesamiento de datos. El software se construye para procesar datos, para transformar datos de una forma a otra, es decir, para aceptar una entrada, transformarlos de alguna forma y producir una salida. En el ambiente del procesamiento de datos, se debe considerar tres aspectos: el flujo de datos, el contenido de los datos y la estructura de datos.

El flujo de datos representa la manera en la que los datos cambian conforme se mueven a través del sistema. El contenido de los datos representa los elementos de datos individuales que componen otros elementos mayores. Y, por último, la estructura de datos representa la organización interna de los distintos elementos de datos.

2.1.1. LAS ESTRUCTURAS DE DATOS

Las estructura de datos es una representación de la relación lógica existente entre los elementos de datos individuales. En los sistemas de información se materializan en archivos o bases de datos.

La complejidad de las estructuras de datos, si bien está limitada al ingenio del diseñador, parte de un número reducido de estructuras básicas que constituyen los bloques con los que se construyen estructuras más complejas.

El elemento básico es el dato elemental, representa un valor simple que puede ser referenciado mediante un identificador. Dentro de los lenguajes de programación, el tamaño y el formato de un dato elemental puede variar dentro de los limites impuestos por el lenguaje. Por ejemplo, “nombre” puede ser representado mediante un dato elemental variable de tipo texto con 20 posiciones. Se pueden agrupar datos elementales en una lista u organización continua denominada vector secuencial. Cuando el vector secuencial se amplia a “n” dimensiones (n>1) se crea un espacio n-dimensional denominado arreglo.

Estas tres estructuras, dato elemental, vector secuencial y arreglo, se pueden organizar de distintas formas. Una lista enlazada es una estructura de datos que organiza algunas de las estructuras anteriores no contiguas de modo que puedan ser procesados como una lista. Cada nodo contiene una organización de datos determinada y uno o más punteros que indican la posición de memoria del siguiente elemento de la lista. Se pueden añadir y quitar nodos en cualquier posición de la lista redefiniendo los punteros.

Cualquier otro tipo de estructura de datos se puede construir a partir de las descriptas. Por ejemplo una estructura de datos jerárquica se implementa utilizando listas multienlazadas que contienen datos elementales, vectores o espacios n-dimensionales.

2.1.2. LAS ANOMALÍAS DE ACTUALIZACIÓN

Las operaciones básicas que se realizan en las estructuras de datos almacenados son las consultas y las actualizaciones. Cuando se hacen persistentes los datos, se deben organizar de modo tal que las consultas puedan realizarse de manera eficiente. Las consultas no modifican los contenidos de las estructuras de datos, solamente inspeccionan su contenido.

En las bases de datos relacionales las estructuras de almacenamientos básicas son las tablas. Se pueden considerar como archivos planos, donde cada elemento que compone la estructura no tiene a su vez una estructura interna. Una base de datos se puede considerar como un conjunto de archivos compuestos por datos elementales relacionados y vinculados entre sí; esas vinculaciones epresentan las interrelaciones entre los mismos.

Las actualizaciones en el contenido de las estructuras de datos, esto es ingresar, modificar o eliminar uno o más datos, sí producen modificaciones en su contenido y el tipo de estructura condiciona estas operaciones.

Se denomina anomalías de actualización a los complicaciones que se producen en las operaciones de actualización de los datos en las estructuras de almacenamiento ineficientes. Estas anomalías se producen, principalmente, por la

redundancia de los datos almacenados que genera inconsistencia en el proceso de actualización.

2.1.3. LA NORMALIZACIÓN

Para solucionar los inconvenientes en las anomalías de actualización se propusieron distintas organización de los datos. La normalización es el proceso que permite transformar una estructura ineficiente en un conjunto de estructuras más eficientes que eviten esos problemas.

2.1.4. EL CICLO DE VIDA DE LOS DATOS

Dentro de un sistema de información administrativo se puede identificar un ciclo de vida de los datos. Comienza con la generación y captura de datos, ésta tiene lugar en virtud de una transacción interna o de un evento externo. Todo procesamiento ulterior, después de la generación y/o captura, se puede dividir en:

Almacenamiento o destrucción: la generación de datos es el resultado de ciertos fenómenos externos o internos, los cuales son observados y registrados en el sistema información. Si no resultan útiles se destruyen.

Recuperación: es el mecanismo que permite obtener los datos almacenados para su utilización. Mediante los sistemas administradores de base de datos, esta actividad no supone mayores inconvenientes.

Evaluación: cuando los datos son recuperados, normalmente es necesario evaluarlos para determinar si tienen que ser sometidos a un procesamiento posterior, almacenarlos nuevamente o destruirlos. Una vez que los datos se han transformado en información y por consiguiente han sido utilizados por el usuario, se vuelven a evaluar.

Análisis: los datos pueden analizarse antes de su uso mediante la identificación de áreas de interés.

Síntesis: a menudo se requiere agrupar muchos datos o sintetizarlos para estructurar un todo significativo.

Utilización: una vez transformados en un formato adecuado para que sean utilizados por los usuarios, los datos se convierten en información. Luego de ser utilizados, la información retorna al ciclo como datos, luego son evaluados nuevamente para saber si conviene guardarlos en su forma original o en otra.

Destrucción: terminada la evaluación de los datos, si no son útiles, pueden ser destruidos.

2.2. LA INFORMACIÓN

Desde la perspectiva de la teoría de la información, se define a ésta como un conjunto de datos que permiten aclarar algo sobre aquello que es desconocido. La llegada de una información se podría producir como consecuencia de la recepción de un mensaje, entendiendo como tal una emisión desde una fuente informativa. Cuando se sabe con seguridad que un hecho va a ocurrir, no contiene información alguna. Un suceso, por lo tanto, contendrá mayor cantidad de información cuanto menor sea la probabilidad de ocurrencia del mismo.

La teoría de la información, ciencia desarrollada por Claude Shannon plantea, entre otros, el objetivo de hacer lo más eficiente posible la transmisión de información de un punto a otro en una red de comunicación.

La medida de la información está relacionada con la posibilidad de que la fuente pueda elegir entre varios mensajes posibles. Luego, al poder la fuente seleccionar el que será transmitido entre varios mensajes, es evidente que el usuario tendrá incertidumbre respecto al mensaje que podrá recibir.

Se puede suponer que cada mensaje tiene asociada una probabilidad, de forma tal que cuanto mayor sea la probabilidad de que ese mensaje sea cierto, menor será la información que contiene para el usuario.

Se define como cantidad de unidades de información (figura 2.2) que un usuario recibe, al serle entregado un mensaje, como el logaritmo de la inversa de la probabilidad de ocurrencia del suceso contenido en él.

$$I_{(a)} = \log 1/p_{(a)}$$

Figura 2.2: Cantidad de unidades de información

Desde el punto de vista de la conducta, la información predispone a actuar de cierto modo basándose en el procesamiento de los datos recibidos. La información es un conjunto de estímulos que desencadenan el comportamiento, los datos, en cambio, son símbolos que describen a un objeto.

Entonces, otra definición de información (figura 2.3) de mayor utilidad en el ambiente de los sistemas de información, propone que la información es un conjunto de datos procesados por el usuario y utilizados para la toma de decisión. Se incluye en la definición tres aspectos: los datos, el usuario y la posibilidad de que esa información sea útil para la toma de decisiones.

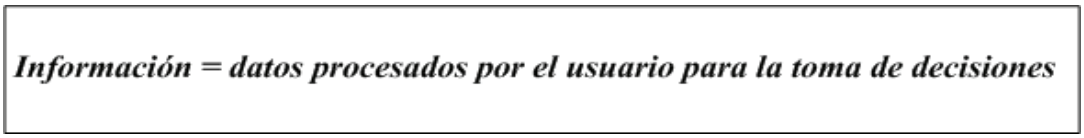


Figura 2.3: Definición de información

2.2.1. LOS ATRIBUTOS DE LA INFORMACIÓN

Para entender a la información como componente importante en la toma de decisión, Davis y Olson proponen y analizarán una serie de atributos que la caracteriza:

2.2.1.1. LOS OBJETIVOS

La información debe tener un objetivo o finalidad en el momento de ser trasmitida sino será solamente datos o ruido. El propósito básico de la información es informar, evaluar, convencer, tomar decisiones, controlar, buscar, etc.

2.2.1.2. LA FORMA DE REPRESENTACIÓN

La forma de representación de la información al ser humano es, esencialmente, sensorial, es decir, intervienen la vista, el oído, el tacto, el gusto y el olfato, pero en las organizaciones, el formato es esencialmente visual y auditivo. La forma de representación de la información es determinante para el usuario. En los sistemas informáticos los formatos de entrada y salida de datos constituye una de las tareas de diseño más relevantes.

2.2.1.3. LA REDUNDANCIA

La redundancia es el exceso de información. Si el costo de un error es alto, la redundancia no es necesariamente perjudicial en la medida que esté controlada y permita, además, reducir la incertidumbre. Es importante destacar una vez más la diferencia entre dato e información; en las estructuras de almacenamiento, la redundancia es perjudicial, ya que generan inconsistencia. En cambio, la información redundante puede reducir la incertidumbre del tomador de decisiones.

2.2.1.4. LA FRECUENCIA

La frecuencia de la información es la cantidad de veces, en una determinada unidad de tiempo, que ésta se transmite. La frecuencia repercute en el valor

de la información y está vinculada con la modificación que se realice en ella. Es tan perjudicial una baja frecuencia de la información que varíe constantemente como una alta frecuencia de información que varíe poco.

2.2.1.5. EL COSTO

El costo de la información es el precio de conseguir la información y es, indudablemente, un factor limitante en su obtención. Está íntimamente vinculada con el valor de información.

2.2.1.6. EL VALOR

El valor de la información representa la utilidad que ésta tiene para el proceso de toma de decisiones. El costo de obtener información, aunque no es una tarea simple, puede ser determinado. En cambio, el valor de la información se torna más difícil debido a que muchos de los beneficios obtenidos por ésta son de difícil medición, por ejemplo: la captación de nuevos clientes, la conformidad del usuario, etc.

2.2.1.7. LA DENSIDAD

La densidad de la información representa el "volumen" de información. Un informe escrito tendrá menor densidad que un gráfico o una tabla, La utilización de modelos gráficos, en lugar de textuales, en la representación de los sistemas de información, se justifica considerando este atributo.

2.3. LA COMUNICACIÓN

El modelo básico de la comunicación se compone de los siguientes elementos: la fuente, que es el emisor y contiene una población de mensajes posibles; el codificador, que actúa sobre el mensaje proveniente de la fuente para convertirlo en señales que acepte el canal; el canal de señales, que es el medio por donde se trasmite la información; el decodificador, que actúa sobre las señales recibidas extrayendo el mensaje en un formato reconocible por el receptor; el receptor, que es el que recibe el mensaje y por último, el ruido, que son hechos externos que introducen interferencias en los sistemas de información.

2.4. EL HOMBRE COMO PROCESADOR DE INFORMACIÓN

El hombre es, en definitiva, el objetivo y fin de los sistemas de información, él será el que utilice la información para la toma de decisiones. Se analizarán algunos conceptos generales de cómo el ser humano procesa la información.

El punto de arranque en el procesamiento de información por el hombre es la recepción de la energía ambiental que afecta a la actividad nerviosa. La sensación no es un mero proceso mecánico, la mente participa en la captación e interpretación de percepciones sensoriales.

El ser humano es capaz de formar varios patrones con la misma entrada ambiental. Algunos factores que influyen en el reconocimiento de patrones tanto visuales como auditivos, son el tipo de estímulo, su duración, el retrazo entre dos representaciones del mismo estímulo, el procesamiento después de producirse el mismo, el aprendizaje y la memoria.

En el diseño de sistemas de información administrativo, el reconocimiento de patrones tiene una utilidad directa. La representación gráfica de los informes y el desarrollo de los caracteres que leerán las personas, se basa en el reconocimiento humano de patrones.

La sintaxis es el conocimiento sobre la forma de un lenguaje. En los sistemas de información administrativa se supone que el lenguaje usado en los informes y formularios ha de expresarse en oraciones o frases que se comprendan dentro del contexto de la aplicación. Es posible formar oraciones sintácticamente correctas pero que carezcan de significado, o de significado ambiguo. Además, los conceptos se expresan con palabras y estas tienen diferentes significados dependiendo de las estructuras cognitivas de las personas que la utilizan.

Estos son los problemas semánticos. Las definiciones y significados se facilitan gracias a los procesos cognitivos que colocan las palabras e imágenes en relaciones significativas entre sí y con el contexto en donde se emplean.

Las herramientas de modelado en los sistemas de información utilizan componentes gráficos y convenciones que intentan representar en un formato carente de ambigüedad la realidad. El significado de los elementos representativos está acotado a una sintaxis más simple para, de esa forma, restringir los significados semánticos.

En el procesamiento de información, la memoria es indispensable aunque ésta sea breve. De la misma manera, la computadora debe hacer persistente sus datos para utilizaciones ulteriores. El sistema de memoria en el ser humano se compone de memoria a corto plazo, denominada primaria, y memoria a largo plazo, denominada secundaria. La memoria a corto plazo es limitada en exceso, se puede retener apenas 7 2 elementos en ella. La memoria a largo plazo en el ser humano es muy grande.

En términos de entrada, se distinguen dos aspectos: la memoria episódica o aquellos eventos específicos que se vivencian o se perciben y la memoria semántica que es la estructuración de las relaciones, asociaciones y representaciones espaciales.

Los sistemas de información deben diseñarse de modo tal que compensen lo más posible la incapacidad de los administradores de recordar totalmente y con exactitud los patrones de la memoria a largo plazo.

La forma en que se presentan los estímulos externos al administrador puede afectar en gran parte la eficiencia con la que se procese la información.

2.5. RESUMEN

Los gerentes utilizan a la información para realizar sus actividades diarias de toma de decisiones. Los datos son símbolos que describen algo, estos se estructuran para describir un objeto. La información, en cambio, son datos procesados por el usuario para la toma de decisiones. Los usuarios requieren información para la toma de decisiones; los diseñadores precisan tener la libertad de proponer distintas estructuras de datos para hacerlos persistentes en el tiempo sin redundancia y que, una vez recuperados, brinden la información solicitada por el usuario.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 3

Introducción al Concepto de Sistemas

Un sistema es un conjunto de componentes que se interrelacionan con el fin de lograr un objetivo común. La simple colección de componentes, por lo tanto, no constituye un sistema, éstos, además, deben estar relacionados entre sí y tener un propósito que los reúna.

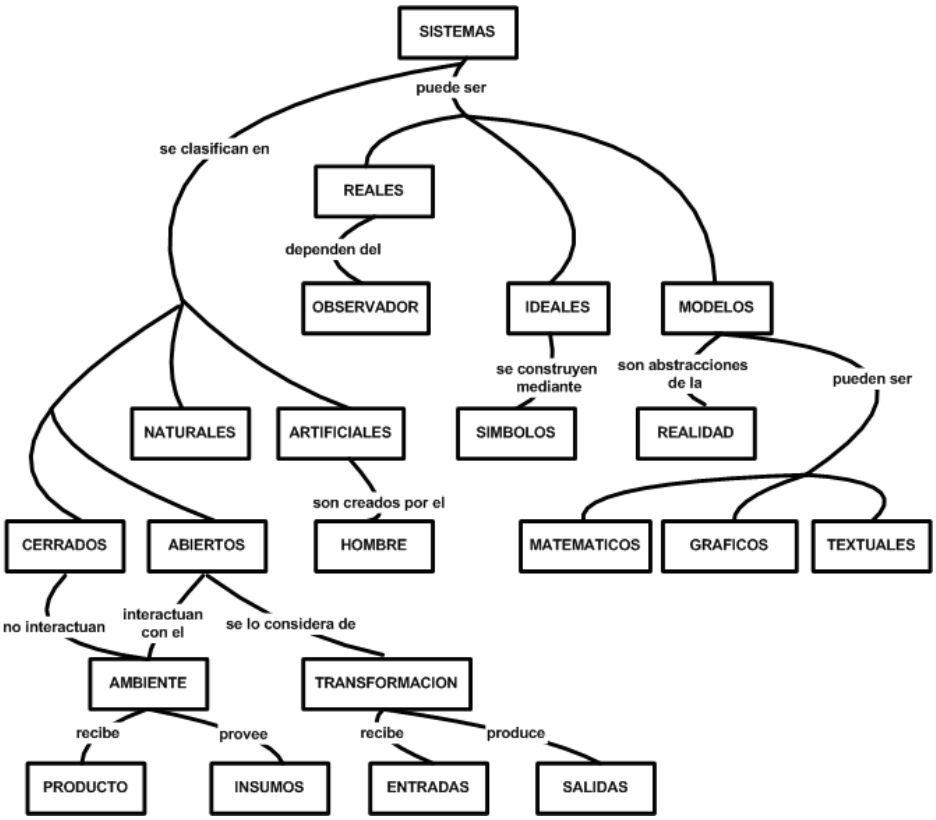


Figura 3.1: Mapa conceptual de “Sistemas”

Las propiedades de los sistemas en general son también aplicables a los sistemas de información en particular, por tal motivo, el conocimiento de esas propiedades será de utilidad en el diseño y construcción de sistemas de información.

Comprender el funcionamiento de un sistema como un todo puede ser una tarea muy compleja. La descomposición funcional esto es, la división de un sistema en partes, es uno de los mecanismos que se utilizan para dominar esa complejidad.

El esfuerzo para resolver un problema es mayor que la suma de los esfuerzos parciales que se debe realizar para resolver cada una de las partes en que se divide ese problema. Este mecanismo de descomposición, si bien muy útil para comprender el funcionamiento de los sistemas, no debe aplicarse de forma arbitraria; existen dos criterios generales, aunque no cualitativos, que auxilian en el proceso de la descomposición, estos son la cohesión y el acoplamiento.

La cohesión es la relación interna que existe entre los componentes de un sistema, esta ligazón debe ser alta. El acoplamiento, por otro lado, es la relación que existe entre los procesos después de la descomposición y es recomendable que sea baja.

Cuando ser descompone un sistema en partes, cada una de éstas debe realizar una función clara y fácilmente identificable, es decir una alta cohesión, y la relación entre éstas partes debe ser mínima, lo que se denomina un bajo acoplamiento. Estos dos criterios se utilizarán en el proceso de análisis y en las actividades de diseño de sistemas de información.

3.1. LAS DEFINICIONES DE SISTEMAS

Todo sistema tiene, cuanto menos dos elementos, y éstos deben estar interconectados. Un concepto fundamental en el pensamiento sistémico es la relación jerárquica entre los sistemas, con excepción del universo total, un sistema está integrado por subsistemas de menor orden y éste también es parte de un sistema mayor. Por lo tanto, los conceptos de sistema y subsistema son intercambiables, dependiendo de la ubicación del observador.

Yourdon detalla en su libro diversas definiciones de sistemas. Entre ellas:

1. Un grupo de elementos interdependientes o que interactúan regularmente formando un todo. Por ejemplo, un sistema numérico.
2. Un grupo de cuerpos que interactúan entre sí bajo las influencias de fuerzas relacionadas. Por ejemplo, el sistema gravitacional.
3. Una mezcla de substancias que tiendan al equilibrio. Por ejemplo, un sistema termodinámico.
4. Un grupo de fuerzas y objetos naturales. Por ejemplo, un sistema de ríos.
5. Un grupo de aparatos o una organización que forma una red para distribuir algo o para servir a un propósito común. Por ejemplo, una red telefónica, un sistema de calefacción, las autopistas.

6. Un grupo de órganos del cuerpo que juntos llevan a cabo una o más funciones vitales. Por ejemplo, el sistema digestivo, o el mismo cuerpo considerado como una unidad funcional.
7. Un juego organizado de doctrinas, ideas o principios con la intención de explicar el trabajo de un todo sistemático. Por ejemplo, el sistema newtoniano de la mecánica.
8. Un procedimiento organizado o establecido. Por ejemplo, el sistema mecanografía al tacto.
9. Un sistema de clasificar, simbolizar o esquematizar. Por ejemplo, el sistema decimal.

3.2. LAS CARACTERÍSTICAS GENERALES DE LOS SISTEMAS

Todos los sistemas tienen objetivos, estos propósitos son las metas o fines hacia los cuales pretende llegar. El ambiente, en donde esta inmerso el sistema, es todo lo que está fuera de él, engloba lo que está fuera del control del sistema. De todos modos, el ambiente actúa sobre el sistema cuando le provee insumos y cuando recibe de él sus salidas.

Los sistemas tienen límites que los separan de su medio ambiente. El concepto de frontera ayuda a entender la distinción entre sistema abierto y cerrado. Los límites se definen con relativa facilidad en los sistemas naturales y físicos, pero son muy difíciles de delinear en los sistemas sociales, tales como las organizacionales.

Los sistemas abiertos intercambian información, energía o materia con su medio ambiente. Un sistema relativamente cerrado tiene límites rígidos e impermeables, mientras que el sistema abierto no, ya que produce intercambios con su medio ambiente.

El sistema abierto puede ser considerado como un modelo de transformación. Es una relación dinámica con su medio ambiente, recibe varias entradas, las transforma de alguna manera y exporta productos como salida del sistema.

El concepto de retroalimentación es importante para entender de qué manera un sistema mantiene un estado estable. En lo referente a la información, los productos o el proceso del sistema, es retroalimentado en forma de entrada al sistema, quizá con cambios en el proceso de transformación y/o en los productos futuros. La retroalimentación negativa es una entrada informativa que indica que el sistema se está desviando de un curso preestablecido y debe reajustarse hacia un nuevo estado estable.

Un sistema abierto, para sobrevivir, debe alcanzar un estado en el cual consuma suficientes insumos de su medio ambiente para compensar sus productos más la energía y materiales usados en la operación del sistema.

3.3. LA TEORÍA GENERAL DE SISTEMAS

La teoría general de sistemas es una forma sistemática de aproximación y representación de la realidad. Cuando se hace referencias a los sistemas se piensa en una totalidad cuyas propiedades no son atribuibles a la simple suma de las propiedades de sus partes o componentes. En las definiciones más corrientes se identifican los sistemas como conjuntos de elementos que guardan estrechas relaciones entre sí, que mantienen al sistema directa o indirectamente unido de modo más o menos estable y cuyo comportamiento global persigue, normalmente, algún tipo de objetivo.

Esas definiciones, que referencian principalmente a los procesos sistémicos internos deben ser, necesariamente, complementadas con una concepción de sistemas abiertos, en donde quede establecido un flujo de relaciones con el ambiente como condición para la continuidad sistémica.

Existen dos visiones estrategias para la investigación de sistemas generales:

1. La visión de sistemas, en donde las distinciones conceptuales se concentran en una relación entre el todo, esto es, el sistema completo y sus partes o elementos y,
2. La visión de sistemas en donde las distinciones conceptuales se concentran en los procesos de la frontera. Es decir, entre el sistema y su medio ambiente.

En el primer caso, la principal característica de un sistema está dada por la interdependencia de las partes que lo integran y el orden subyacente entre ellas. En el segundo, lo central son los flujos de entradas y de salidas mediante los cuales se establece una relación entre el sistema y su ambiente. Ambos enfoques son complementarios.

3.4. LOS CONCEPTOS BÁSICOS DE LA TEORÍA GENERAL DE SISTEMAS

La teoría general de sistemas propone un conjunto de conceptos básicos que pueden utilizarse cuando se diseñan sistemas de información.

3.4.1. LA ADAPTABILIDAD

La adaptabilidad es la propiedad que tienen los sistemas de aprender y modificar alguna característica de acuerdo a las modificaciones que sufre el contexto. Esto se logra a través de un mecanismo de adaptación que permita responder a los cambios, tanto internos como externos a través del tiempo. Para que un sistema pueda ser adaptable debe tener un fluido intercambio con el medio en el que se desarrolla.

3.4.2. EL CONCEPTO DE CAJA NEGRA

El concepto de caja negra se utiliza para representar a los sistemas cuando no se sabe, o no interesa en ese momento, qué elementos lo componen, pero sí se sabe que a determinadas entradas corresponden determinadas salidas y con ello se puede inducir que, a partir de ciertos estímulos, las variables internas funcionarán en forma previsible.

3.4.3. LA CENTRALIZACIÓN

Un sistema se lo denomina centralizado cuando tiene un núcleo que comanda a todos los demás componentes y estos, además, dependen para su activación del primero, ya que por sí solos no son capaces de generar ningún proceso.

3.4.4. LA DESCENTRALIZACIÓN

Los sistemas descentralizados son aquellos donde el núcleo de comando y decisión está formado por varios subsistemas. En este caso el sistema no es tan dependiente, sino que puede llegar a contar con subsistemas que actúan de reserva y que sólo se pongan en funcionamiento cuando falle el sistema que debería actuar en dicho caso.

3.4.5. LA COMPLEJIDAD

Este concepto, por un lado, indica la cantidad de elementos de un sistema, esto es la complejidad cuantitativa y, por el otro, sus potenciales interacciones, es decir la conectividad y el número de estados posibles que se producen.

3.4.6. EL CONTEXTO

Un sistema siempre estará relacionado con el contexto que lo rodea, es decir, el conjunto de objetos exteriores al mismo que lo influyen directamente y a su vez el sistema, aunque en una menor proporción, influye también sobre el contexto; se trata de una relación mutua de contexto-sistema. Para determinar los límites de un sistema se deben considerar el contexto de interés y la determinación del alcance del límite de interés entre el contexto y el sistema.
Se suele representar al contexto como un círculo que encierra al sistema, dejando afuera del límite de interés los elementos que no interesan al analista.

En lo que hace a las relaciones entre el contexto y los sistemas y viceversa, es posible que sólo interesen algunas de estas relaciones, con lo que habrá un límite de interés relacional.

Determinar el límite de interés es fundamental para marcar el foco de análisis, puesto que sólo será considerado lo que quede dentro de ese límite. Entre el

sistema y el contexto existen infinitas relaciones. Es lógico, por lo tanto, que no se tengan en cuenta todas, sino aquellas que interesan al análisis.

3.4.7. LAS ENTRADAS

Las entradas son los ingresos al sistema, estos que pueden ser recursos materiales, humanos o información. Las entradas constituyen la fuerza de arranque que suministra al sistema sus necesidades operativas. Las entradas pueden ser en serie, esto es, el resultado o la salida de un sistema anterior con el cual el sistema en estudio está relacionado en forma directa. Puede ser una salida aleatoria, es decir al azar, donde el termino "azar" se utiliza en el sentido estadístico. Las entradas aleatorias representan las entradas potenciales para un sistema. Por último, la entrada puede ser retroacción, que es la reintroducción de una parte de las salidas del sistema en sí mismo.

3.4.8. LA ESTABILIDAD

Un sistema se dice estable cuando puede mantenerse en equilibrio a través del flujo continuo de materiales, energía e información. La estabilidad de los sistemas ocurre mientras los mismos pueden mantener su funcionamiento y trabajen de manera efectiva.

3.4.9. LA ALIMENTACIÓN DELANTERA

Es una forma de control de los sistemas, donde dicho control se realiza a la entrada del mismo, de tal manera que él no tenga entradas corruptas o malas, de esta forma al no haber entradas defectuosas, las fallas no serán consecuencia de las entradas sino de los procesos mismos que componen al sistema.

3.4.10. LA FRONTERA

En algunos sistemas, sus fronteras o límites coinciden con las discontinuidades estructurales que existen entre éstos y sus ambientes, pero normalmente la demarcación de los límites de los sistemas queda en manos de un observador. En términos operacionales puede decirse que la frontera del sistema es aquella línea que separa al sistema de su entorno y que define lo que le pertenece y lo que queda fuera de él.

3.4.11. LOS LÍMITES

Todo sistema tiene elementos interiores y exteriores a él, así mismo lo que es externo al sistema, forma parte del ambiente y no al propio sistema. Los límites se encuentran íntimamente vinculados con la cuestión del ambiente, se puede definir como la línea que forma un círculo alrededor de variables seleccionadas de forma tal que existe un menor intercambio de energía a través de esa línea con el interior del círculo que delimita.

3.4.12. LA MANTENIBILIDAD

La mantenibilidad es la propiedad que tiene un sistema de permanecer constantemente en funcionamiento. Para ello utiliza un mecanismo que asegure que los distintos subsistemas están balanceados y que el sistema total se mantiene en equilibrio con su medio.

3.4.13. LOS MODELOS

Los modelos son construcciones diseñadas por un observador que tiene como objetivo identificar y mensurar las relaciones sistémicas complejas. Todo sistema real tiene la posibilidad de ser representado en más de un modelo. La decisión, en este punto, depende tanto de los objetivos del modelador como de su capacidad para distinguir las relaciones relevantes con relación a tales objetivos. La esencia de la modelización sistémica es la simplificación. El metamodelo sistémico más conocido es el esquema entrada-transformación-salida.

3.4.14. EL PROCESO

El proceso es lo que transforma una entrada en salida, como tal puede ser una máquina, un individuo, una computadora, un producto químico, una tarea realizada por un miembro de la organización, etc. En la transformación de entradas en salidas es importante comprender cómo se efectúa la misma.

3.4.15. LA RETROALIMENTACIÓN

La retroalimentación son el conjunto de procesos mediante los cuales un sistema abierto recoge información sobre los efectos que sus decisiones internas tienen en el medio ambiente y a partir de ellas dirige las decisiones sucesivas. La retroalimentación se produce cuando las salidas del sistema o la influencia de las salidas del sistema en el contexto, vuelven a ingresar al mismo como recursos o información. Permite, además, el control de un sistema y que el mismo tome medidas de corrección sobre la base de la información retroalimentada.

3.4.16. LAS SALIDAS

Las salidas de los sistemas son los resultados que se obtienen de procesar las entradas. Al igual que las entradas éstas pueden adoptar la forma de productos, servicios e información. Las mismas son el resultado del funcionamiento del sistema o, alternativamente, el propósito para el cual existe el sistema.

Las salidas de un sistema se convierten en entrada de otro, que las procesará para convertirla en otra salida, repitiéndose este ciclo en forma indefinida.

3.4.17. LA SINERGIA

Todo sistema es sinérgico en tanto el examen de sus partes, en forma aislada, no puede explicar o predecir su comportamiento. La sinergia es, en consecuencia, un fenómeno que surge de las interacciones entre las partes o componentes de un sistema.

3.4.18. LOS SISTEMAS ABIERTOS

Son sistemas que importan y procesan elementos, tales como energía, materia o información, de sus ambientes siendo ésta una característica propia de todos los sistemas vivos. Que un sistema sea abierto significa que establece intercambios permanentes con su ambiente, intercambios que determinan su equilibrio, capacidad reproductiva o continuidad.

3.4.19. LOS SISTEMAS CERRADOS

Un sistema es cerrado cuando ningún elemento exterior entra y ninguno sale fuera del sistema. Estos alcanzan su estado máximo de equilibrio al igualarse con el medio. En ocasiones el término sistema cerrado es también aplicado a sistemas que se comportan de una manera fija, rítmica o sin variaciones, como el caso de los circuitos cerrados.

3.4.20. LOS SUBSISTEMAS

Se entiende por subsistemas a un conjunto de elementos y relaciones que responden a estructuras y funciones que están especializadas dentro de un sistema mayor. En términos generales, los subsistemas tienen las mismas propiedades que los sistemas y su delimitación es relativa a la posición del observador y al modelo que tenga de éstos. Desde este punto de vista se puede hablar de subsistemas, sistemas o supersistemas, en tanto éstos posean las características sistémicas.

3.5. LA CLASIFICACIÓN DE LOS SISTEMAS

Los sistemas, en general, pueden clasificarse en reales, ideales y modelos. Los primeros suponen una existencia que es independiente del observador, los segundos son construcciones simbólicas, como el caso de la lógica y las matemáticas, mientras que el tercer tipo corresponde a abstracciones de la realidad, en donde se combina lo conceptual con las características de los objetos.
Con relación a su origen los sistemas pueden ser artificiales o naturales, según éstos sean, o no, creados por el hombre. Con relación al ambiente o grado de

aislamiento, los sistemas pueden ser cerrados o abiertos, según el tipo de intercambio que establecen con su entorno.

Existen, dentro del ámbito de los sistemas de información, distintas clasificaciones.

3.5.1. LOS SISTEMAS EN LÍNEA

Los sistemas en línea son aquellos que aceptan la entrada directamente del área donde se creó. También es el sistema en el que la salida, o el resultado del proceso, se devuelve directamente a donde es requerido.

Los datos, en este tipo de sistemas, pueden ser modificados o recuperados o ambas cosas en forma rápida y sin tener que efectuar accesos a otros componentes de información del sistema.

3.5.2. LOS SISTEMAS EN TIEMPO REAL

Un sistema computacional de tiempo real puede definirse como aquel que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho contexto en ese momento.

Entre otros se puede nombrar a los sistemas de control de procesos, éstos controlan refinerías, procesos químicos y operación de máquinas; los sistemas de cajeros automáticos; los sistemas de alta velocidad para adquisición de datos; los sistemas de guía de proyectiles, que debe ajustar y orientar continuamente los propulsores; los sistemas de conmutación telefónica, que controlan la transmisión de voz y datos en miles de llamados telefónicos, detectando los números marcados, condiciones de ocupado etc.; los sistemas de vigilancia de pacientes que ajustan la dosis de los medicamentos al detectar cambios en los signos vitales, etc.

3.5.3. LOS SISTEMAS DE PROCESAMIENTO DE DATOS

Los sistemas de procesamientos de datos se caracterizan por procesar grandes volúmenes de datos, por el almacenamiento y recuperación de esos datos, por la realización de cálculos, clasificación y ordenamiento. Este tipo de sistema mejora las actividades rutinarias, los procesos están bien estructurados, generan resúmenes que serán utilizados por otros sistemas.

3.5.4. LOS SISTEMAS DE INFORMACIÓN GERENCIAL

Los sistemas de información gerencial son sistemas de procesamiento que no toman decisiones por sí mismo, pero que auxilian a los gerentes y otros profesionales de una organización a tomar decisiones en varios aspectos de las operaciones de la organización.

En este tipo de sistemas, el usuario debe identificar los criterios que se utilizarán para tomar la decisión, Algunos de ellos son binarios, otros deben transformarse a binarios en el diseño. De acuerdo con la prioridad se le pueden dar "pesos" para que vayan definiendo criterios que conduzcan a alternativas que puedan ser evaluadas y analizadas.

Estos sistemas soportan una amplia gama de tareas organizacionales, más que los sistemas procesamiento de datos, incluyendo análisis y toma de decisión. Se basa en hechos pasados y distribuye la información relevante para una correcta toma de decisión. Los procesos están bien estructurados y, además, son periódicos.

3.5.5. LOS SISTEMAS DE APOYO A LA TOMA DE DECISION

Los sistemas de apoyo a la toma de decisión son utilizados por los gerentes para evaluar y analizar la gestión de la organización. Estos sistemas ofrecen indicaciones o sugerencias más amplias y generales acerca de, por ejemplo, la naturaleza del mercado, las preferencias de los consumidores, el comportamiento de la competencia, etc.

Estos sistemas realizan procesos semi o no estructurados. Los factores que afectan a la toma de decisión son desconocidos o poco accesible. Son interactivos, es decir, puede conducir a otros requerimientos. Ayuda a la toma de decisión, no tiene una base datos como los anteriores o si los tiene son insuficientes. Se utiliza la información según las necesidades del usuario.

3.5.6. LOS SISTEMAS BASADOS EN EL CONOCIMIENTO

Los sistemas basados en el conocimiento, también denominados sistemas expertos, son sistemas que tienen embebido el conocimiento y una máquina de inferencia para interactuar con el sistema, esto le permite funcionar como especialistas basándose en el conocimiento de especialistas humanos.

3.6. RESUMEN

Todo sistema tiene, cuando menos dos elementos, y estos están interconectados. Un concepto fundamental en el pensamiento de sistemas es la relación jerárquica entre los componentes, con excepción del universo total, un sistema esta integrado por subsistemas de menor orden y es también parte de un suprasistema.

Los sistemas tienen límites que los separan de su medio ambiente. El concepto de límites ayuda a entender la distinción entre sistema abierto y cerrado.

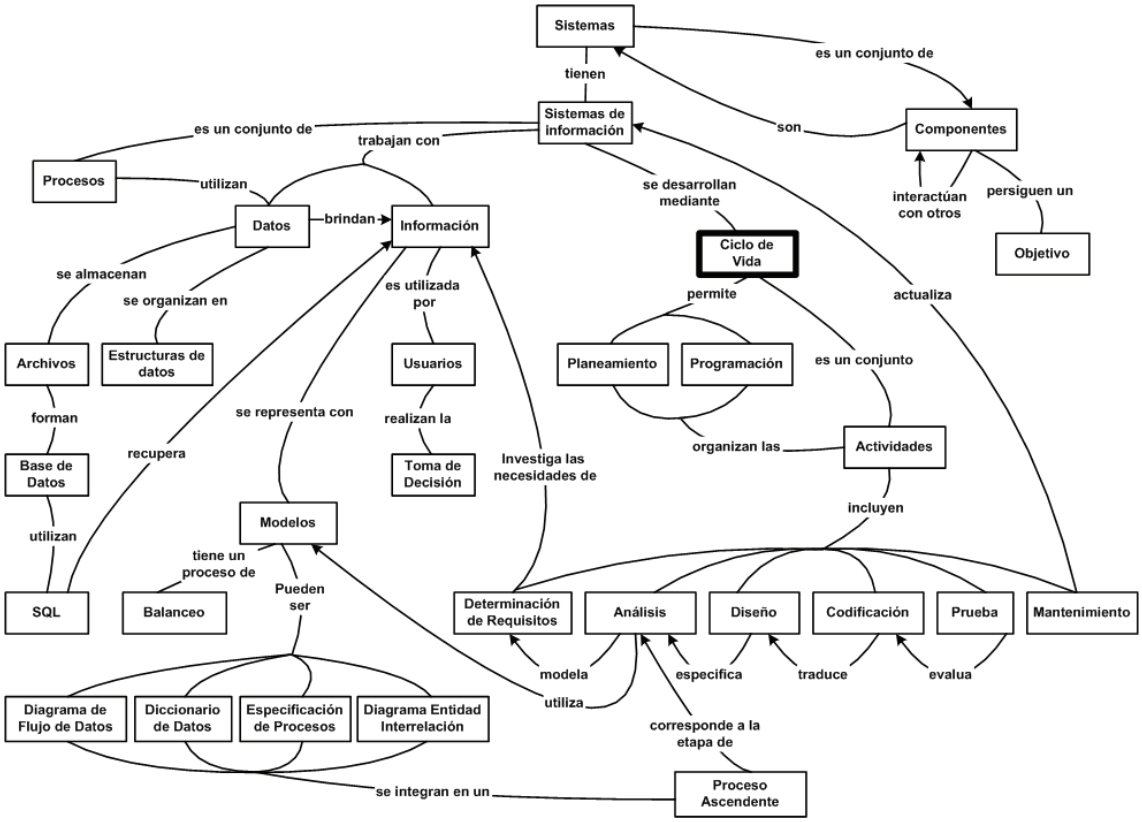
El sistema relativamente cerrado tiene límites rígidos e impermeables, mientras que el sistema abierto tiene límites permeables entre sí mismo y un sistema más amplio. Los límites se definen con relativa facilidad en los sistemas

naturales y físicos, pero son muy difíciles de establecer en los sistemas sociales, tales como las organizacionales. Los sistemas abiertos intercambian información, energía o materia con su medio ambiente.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 4



Ciclo de Vida de Desarrollo de Sistemas

Independientemente del modelo de desarrollo de sistemas que se utilice, del área de aplicación y del tamaño y complejidad del proyecto, el proceso de desarrollo de sistemas contiene siempre una serie de fases genéricas, que existen en todos los paradigmas, denominada ciclo de vida de desarrollo de sistemas.

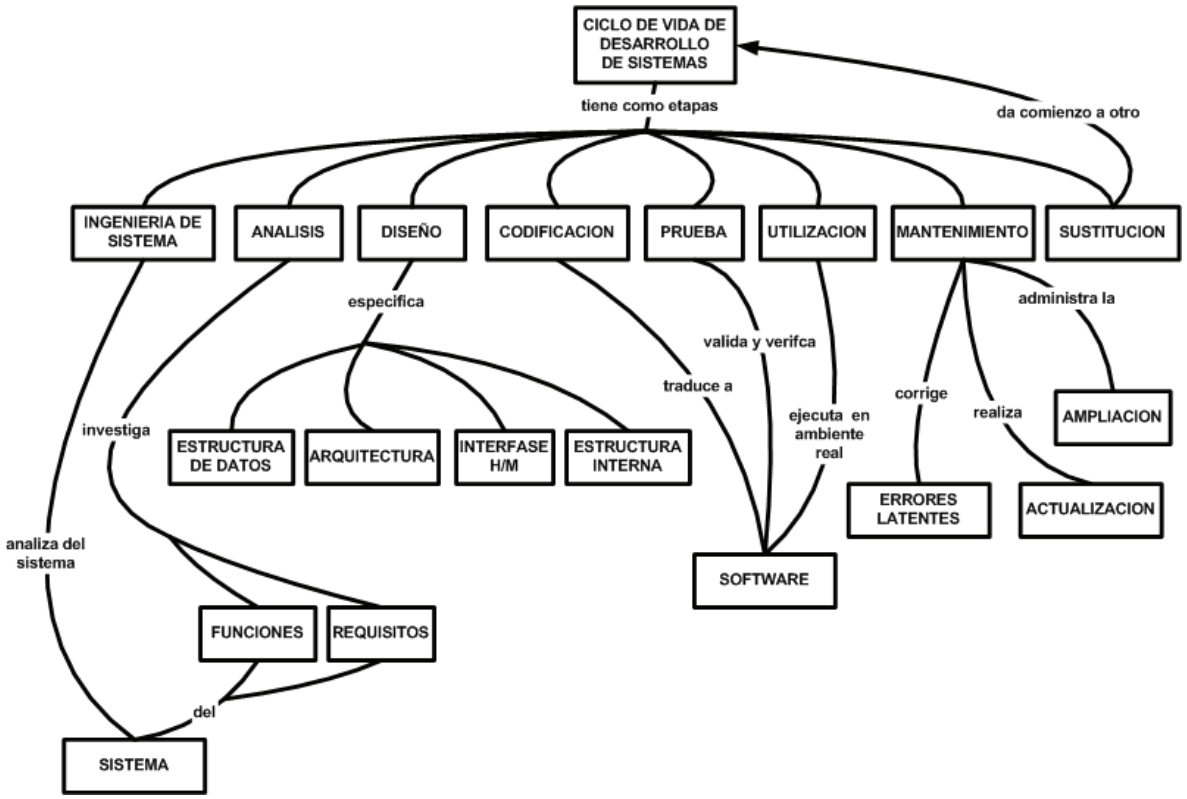


Figura 4.1: Mapa conceptual de “Ciclo de Vida de Desarrollo de Sistemas”

El ciclo de vida de desarrollo de sistemas es una sucesión de etapas por las que atraviesa el software, expresado en un sentido amplio: la documentación, desde que comienza un nuevo proyecto hasta que éste se deja de utilizar.

Cada una de estos pasos puede concebirse como un proceso que contiene flujos de datos de entrada y de salida. El origen del flujo de entrada es la etapa anterior y el destino, la posterior. Además, cada etapa tiene asociada tareas que deben realizarse, y una serie de documentos que producir. Las etapas pueden repetirse en forma secuencial o cíclica, dependiendo del tipo de ciclo de vida utilizado.

Existen diversas formas de realizar el proceso de desarrollo de sistemas de información y cada uno de ellos está asociado a un paradigma determinado de la ingeniería del software, es decir, a una serie de métodos, herramientas y procedimientos que se deben utilizar a lo largo de un proyecto.

La elección de un modelo de ciclo de vida se realiza de acuerdo con la naturaleza del proyecto y de la aplicación, los métodos a utilizar y los controles y entregas requeridos.

4.1. LAS ETAPAS DEL CICLO DE VIDA

Aunque las etapas del ciclo de vida tienen una alta dependencia del modelo de desarrollo utilizado, existen una serie de actividades que les son comunes a todos ellos.

4.1.1. EL ANÁLISIS GENERAL DEL SISTEMA

Debido a que el software es siempre parte de un sistema mayor, el análisis general de sistemas busca establecer cuáles son los requisitos generales del mismo, determinando cuál es el papel específico del software dentro del ámbito más amplio del sistema.

4.1.2. EL ANÁLISIS DE REQUISITOS DEL SOFTWARE

El análisis general del sistema proporciona el ámbito donde actuará el software y su relación con el resto de componentes del sistema, pero antes de comenzar a desarrollarlo es necesario hacer una definición más detallada de la función del software.

El análisis y la definición de los requisitos es una tarea que debe llevarse a cabo conjuntamente por el desarrollador de software y por el cliente. La especificación de requisitos del software es el documento que se produce como resultado de esta etapa.

4.1.3. LA PLANIFICACIÓN DEL PROYECTO SOFTWARE.

Durante esta etapa se realiza el análisis de riesgos, debido a que el desarrollo de cualquier proyecto complejo lleva implícito una serie de riesgos, unos relativos al propio proyecto y otros a las decisiones que deben tomarse durante su desarrollo. Además, se definen los recursos necesarios para desarrollar el software y se establecen las estimaciones de tiempo y costos.

El propósito de esta etapa de planificación es proporcionar una indicación preliminar de la viabilidad del proyecto de acuerdo con el costo y con el cronograma que se haya establecido.

4.1.4. EL DESARROLLO

En esta fase se determina la arquitectura de la aplicación, las estructuras de datos, las interfaces entre los distintos módulos de la arquitectura y entre los componentes y los usuarios y las características procedimentales de los módulos de programación. Los pasos concretos de esta etapa dependen del modelo de ciclo de vida utilizado, en general se realizarán cuatro tareas específicas.

4.1.4.1. EL DISEÑO

El diseño del software traduce los requisitos analizados anteriormente a un conjunto de representaciones, éstas pueden ser gráficas o basadas en algún lenguaje apropiado y describen cómo van a estructurarse los datos, cuál va a ser la arquitectura de la aplicación, cuál va a ser la estructura de cada programa y cómo serán las interfaces. Es necesario seguir criterios de diseño que permitan asegurar la calidad del producto. Una vez finalizado el diseño es necesario revisarlo para asegurar la completitud y el cumplimiento de los requisitos del software.

4.1.4.2. LA CODIFICACIÓN

En esta fase, el diseño se traduce a un lenguaje de programación, entregando como resultado un programa ejecutable. La buena calidad de los programas desarrollados depende en gran medida de la calidad del diseño.

Una vez codificados los programas deben revisarse su estilo y claridad, y se debe comprobar que exista una correspondencia con la estructura de los mismos definida en la fase de diseño.

4.1.4.3. LA PRUEBA

Terminada la etapa de programación, es preciso probarlo para detectar errores de codificación, de diseño o de especificación. Las pruebas son necesarias para encontrar el mayor número posible de errores antes de entregar el sistema de información al cliente.

Es necesario probar cada uno de los componentes por separado, esto es, cada uno de los módulos o programas, para comprobar el rendimiento funcional de cada una de estas unidades. A continuación se procede a integrar los componentes para probar toda la arquitectura del software y, además, probar su funcionamiento y las interfaces. En este punto hay que comprobar, además, si se cumplen todos los requisitos de la especificación.

4.1.5. EL MANTENIMIENTO

La fase de mantenimiento se centra en los cambios que va a sufrir el software a lo largo de su vida útil. Estos cambios pueden deberse a la corrección de errores, a cambios en el entorno inmediato del software o a cambios en los requisitos del cliente, con el objetivo de ampliar el sistema.

En la fase de mantenimiento se aplican nuevamente los pasos de las fases de definición y de desarrollo, pero en el contexto de un software ya existente y en funcionamiento. Es quizá, en esta etapa, donde todas las consideraciones sobre calidad en el desarrollo de sistemas se hace más evidente. El costo del manteniento será menor cuanto mayor haya sido el esfuerzo en desarrollar las etapas anteriores con criterios de calidad.

4.2. LA UTILIZACIÓN DEL CICLO DE VIDA

Los modelos de ciclo de vida de sistemas son utilizados para la organización, planificación, contratación de personal, realización de presupuestos y para administrar el proyecto según las características organizacionales particulares de tiempo, espacio y recursos computacionales.

Se utiliza, además, como una guía de qué documentos destinados al cliente se deben generar. Como un marco para analizar o estimar puntos dentro del ciclo de vida en los que es necesaria una mayor cantidad de recursos. Además, se puede utilizar como base para dirigir estudios empíricos hacia el estudio de los factores que afectan la productividad, los costos y calidad del software.

4.3. EL CICLO DE VIDA EN CASCADA

El ciclo de vida en cascada fue derivado de modelos utilizados en ingeniería con el objetivo de introducir una metodología de trabajo en el desarrollo de grandes productos de software. Este modelo está formado por varia etapas que son procesadas en una manera lineal (figura 4.2). El ciclo de vida en cascada es un modelo básico pero muy importante y ha sido el origen de muchos otros, sin embargo, para varios proyectos modernos, ha quedado un poco desactualizado. La versión original del modelo fue levemente mejorada a lo largo del tiempo. Sigue siendo, actualmente, un modelo muy utilizado.

Este modelo representa un ciclo de vida en un sentido amplio, incluye no sólo las etapas de ingeniería sino toda la vida del producto, es decir, la vida útil del software y el mantenimiento, hasta que llega el momento de sustituirlo.

El ciclo de vida en cascada exige un enfoque sistemático y secuencial del desarrollo de software, que comienza en el nivel de la ingeniería de sistemas y avanza a través de fases secuenciales sucesivas. Estas fases son las siguientes:

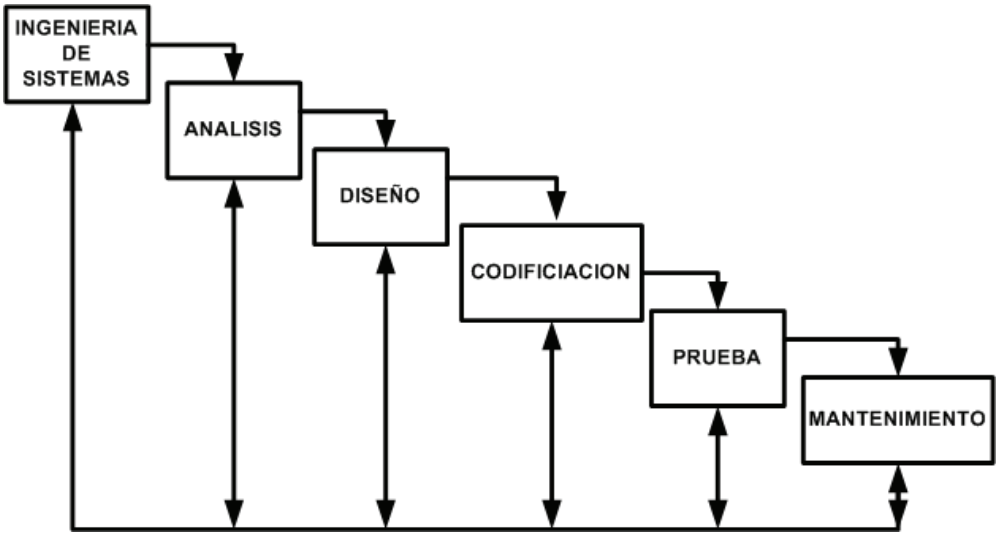


Figura 4.2: Modelo de ciclo de vida en cascada

4.3.1. LA INGENIERÍA DE SISTEMA

El software es siempre parte de un sistema mayor, por lo que siempre va a interrelacionarse con otros elementos, ya sean éstos hardware, máquinas o personas. Por tal razón, el primer paso del ciclo de vida de un proyecto consiste en el análisis de las características y el comportamiento del sistema del cual el software va a formar parte.

La ingeniería del sistema comprende, por tanto, los requisitos globales a nivel del sistema, así como una cierta cantidad de análisis y de diseño en el ámbito superior, sin entrar en mucho detalle.

4.3.2. EL ANÁLISIS DE REQUISITOS DEL SOFTWARE.

El análisis de requisitos debe ser más detallado para aquellos componentes del sistema que se van a implementar mediante software. Se deben comprender cuáles son los datos que se van a utilizar, cuál va a ser la función que tiene que cumplir el software, cuáles son las interfaces requeridos y cuál es el rendimiento, además de otros requisitos no funcionales que se esperan lograr.

Los requisitos, tanto del sistema como del software, deben documentarse y revisarse con el cliente. Como resultado de la fase de análisis, se obtendrá la especificación de requisitos del software. En esta etapa se utilizan las herramientas de modelado que permitirán traducir las especificaciones funcionales

textuales, producidas en la etapa de requerimientos, a un formato carente de ambigüedad.

4.3.3. EL DISEÑO

El diseño se aplica a cuatro características distintas del software, ellas son, las estructuras de datos, la arquitectura de las aplicaciones, la estructura interna de los programas y las interfaces. El diseño es el proceso que traduce los requisitos en una representación del software de forma que pueda conocerse la arquitectura, funcionalidad e incluso la calidad del mismo antes de comenzar la codificación.

4.3.4. LA CODIFICACIÓN

La codificación consiste en la traducción del diseño a un formato que sea comprensible para la computadora. Si el diseño es lo suficientemente detallado, la codificación será relativamente sencilla, y podrá hacerse, al menos en parte, de forma automática, usando generadores de código.

4.3.5. LA PRUEBA

Una vez terminada la etapa de codificación, se obtiene el programa ejecutable y comienza la fase de pruebas. El objetivo de esta etapa es detectar los errores producidos en alguna de las fases de traducción anteriores, especialmente en la codificación. Para ello deben probarse todas las sentencias y, también, todos los módulos que forman parte del sistema.

4.3.6. EL MANTENIMIENTO

Terminada la fase de pruebas, el software se entrega al cliente y comienza la vida útil del mismo. La fase de utilización se solapa con las posteriores, es decir, el mantenimiento y la sustitución y continua hasta que el software, ya reemplazado por otro, deje de utilizarse.

El software sufrirá cambios a lo largo de su vida útil. Estos cambios pueden ser debidos a variadas causas.

Durante la utilización, el cliente puede detectar errores en el software, estos se denominan errores latentes, es decir, errores que no fueron descubiertos en la etapa de prueba pero sí encontrados por el usuario.

Cuando se producen cambios en alguno de los componentes del sistema informático, por ejemplo cambios en la computadora, en el sistema operativo o en los periféricos, se debe adaptar el software a ellos.

El cliente habitualmente requiere modificaciones funcionales, normalmente ampliaciones, que no fueron contempladas inicialmente en el proyecto.

En cualquier caso, el mantenimiento supone volver atrás en el ciclo de vida, a las etapas de codificación, diseño o análisis dependiendo de la magnitud del cambio encarado.

4.3.7. LOS PROBLEMAS CON LOS CICLOS DE VIDA EN CASCADA

Los proyectos de desarrollo de sistemas, en general, no siguen un ciclo de vida estrictamente secuencial tal como lo propone el modelo, siempre hay iteraciones.

El modelo en cascada, a pesar de ser lineal, contiene flujos que permiten la vuelta atrás. Así, desde el mantenimiento se vuelve al análisis, el diseño o la codificación, y también desde cualquier fase se puede volver a la anterior, si se detectan fallos. Estas vueltas atrás no son controladas, ni quedan explícitas en el modelo, y éste es uno de los problemas que presenta este paradigma

El principal inconveniente del modelo, no obstante, es la dificultad para establecer, inicialmente, todos los requisitos del sistema. Normalmente los usuarios no tienen conocimiento de la importancia de la fase de análisis o bien no han pensado en detalle qué es lo que quieren que haga el software.

Los requisitos se van aclarando y refinando a lo largo de todo el proyecto, según se plantean dudas concretas en el diseño o la codificación. Sin embargo, el ciclo de vida clásico requiere la definición inicial de todos los requerimientos y no es fácil acomodar en él las incertidumbres que suelen existir al comienzo de todos los proyectos. Hasta que se llega a la fase final del desarrollo, esto es, la codificación, no se dispone de una versión operativa del programa. Como la mayor parte de los errores se detectan cuando el cliente está en condiciones de probar el sistema, no se detectan hasta el final del proyecto, cuando son más costosos de corregir y más presiones hay para que el programa se ponga definitivamente en marcha.

4.3.8. EL ÁMBITO DE APLICACIÓN DEL CICLO DE VIDA EN CASCADA

Si bien todos estos problemas son ciertos, de todas formas es mucho mejor desarrollar software siguiendo el modelo de ciclo de vida en cascada que hacerlo sin ningún tipo de guías. Además, este modelo describe una serie de pasos genéricos que son aplicables a cualquier otro paradigma.

4.4. EL CICLO DE VIDA MEDIANTE LENGUAJES DE CUARTA GENERACIÓN

Se denomina lenguajes de cuarta generación a un conjunto muy diverso de métodos y herramientas que tiene por objeto facilitar el desarrollo de software. Todas estas herramientas tienen en común permitir al desarrollador especificar algunas características del mismo a un alto nivel de abstracción para luego generar automáticamente el código fuente a partir de la especificación.

Los tipos más habituales de generadores de código cubren uno o varios de los siguientes aspectos:

Generación de código. Son lenguajes de alto nivel que, a partir de la especificación de los requisitos del sistema, generan automáticamente toda la aplicación.

Generación de pantallas y de informes. Son herramientas que permiten diseñar la pantalla dibujándola directamente, incluyendo además el control del cursor y la gestión de errores de los datos de entrada.

Gestión de entornos gráficos. Proporcionan un entorno que facilita el uso de ventanas, además de la edición, control del cursor, movimiento, cambio de tamaño, etc.

Herramientas de acceso a bases de datos. Son lenguajes no procedimentales de alto nivel, derivados normalmente de SQL, que permiten rápidas consultas a los sistemas administradores de base de datos.

La ventaja principal de estas herramientas, esto es, la generación automática de código a partir de especificaciones de alto nivel de abstracción, permite reducir la duración de las fases del ciclo de vida, especialmente la de codificación.

Al igual que con otros modelos, el proceso comienza con la determinación de requisitos de información, que pueden ser traducidos directamente a código fuente usando un generador de código. Sin embargo el problema que se plantea es el mismo que en el ciclo de vida clásico, es muy difícil que se puedan establecer todos los requerimientos desde el comienzo, ya que el cliente puede no estar seguro de lo que necesita o, aunque lo sepa, puede ser difícil expresarlo de la forma en que precisa la herramienta de cuarta generación para poder entenderla.

Si la especificación es pequeña, se puede pasar directamente del análisis de requisitos a la generación automática de código, sin realizar ningún tipo de diseño. Pero si la aplicación es grande, se producirán los mismos problemas que suceden si no se usan los lenguajes de cuarta generación, esto es, mala calidad, dificultad de mantenimiento y poca aceptación por parte del cliente. Es necesario, por lo tanto, realizar un cierto grado de diseño, al menos lo que se denomina una estrategia de diseño, puesto que el propio generador se encarga de parte de los detalles del diseño tradicional, es decir, la descomposición modular, la estructura lógica y la organización de los archivos, etc.

Las herramientas de cuarta generación se encargan también de producir automáticamente la documentación del código generado, pero esta documentación es normalmente muy limitada y, por ello, difícil de seguir. Es necesario completarla hasta obtener una documentación que sea útil.
Debido a que el código generado debería ser correcto y no contener los típicos errores de la codificación manual, es necesaria la fase de pruebas, en primer lugar para comprobar la eficiencia del código generado ya que, por

ejemplo, la generación automática de los accesos a bases de datos puede producir código muy ineficiente cuando el volumen de datos es grande, también para detectar los errores en la especificación a partir de la cual se generó el código y, por último, para que el cliente compruebe si el producto final satisface sus necesidades.

El resto de las fases del ciclo de vida, utilizando estas técnicas, es igual a las del modelo del ciclo de vida en cascada, de la que este no es más que una adaptación a las nuevas herramientas de producción de software.

4.4.1. LOS PROBLEMAS CON LOS LENGUAJES DE CUARTA GENERACIÓN

Muchos de los lenguajes de especificación que se utilizan pueden considerarse como lenguajes de programación de un nivel algo más alto que los anteriores, pero que no logran prescindir de la codificación en sí. Otro inconveniente es que, normalmente, el código generado es ineficiente.

4.4.2. EL ÁMBITO DE APLICACIÓN DE LOS LENGUAJES DE CUARTA GENERACIÓN

El ámbito de aplicación de esta técnica está restringido, casi exclusivamente, al software de gestión. La mayoría de las herramientas de cuarta generación están orientadas a la generación de informes a partir de grandes bases de datos. Esto es debido a que las mejores prestaciones de estos lenguajes está relacionada con la creación de interfaces hombre - máquina y con las consultas a bases de datos.

4.5. LOS PROTOTIPOS

En el ciclo de vida en cascada se dificulta la obtención clara de todos los requisitos del sistema al inicio del proyecto, además, no se dispone de una versión operativa del programa hasta las fases finales del desarrollo, lo que dificulta la detección de errores, dejando para el final y muchas veces por parte del usuario, el descubrimiento de los requisitos inadvertidos en las fases de análisis.

Un modelo de ciclo de vida basado en la construcción de prototipos puede disminuir estos inconvenientes (figura 4.3). En primer lugar, hay que ver si el sistema que se desarrollará es un buen candidato a utilizar el paradigma de ciclo de vida de construcción de prototipos.

En general, cualquier aplicación que presente mucha interacción con el usuario, o que necesite algoritmos que puedan construirse de manera evolutiva, yendo de lo más general a lo más específico es una buena candidata. No obstante, hay que tener en cuenta la complejidad; si la aplicación necesita que se

desarrolle una gran cantidad de código para poder tener un prototipo que enseñar al usuario, las ventajas de la construcción del mismo se verán superadas por el esfuerzo de desarrollar un prototipo que al final habrá que desechar o modificar mucho. También hay que tener en cuenta la disposición del cliente para probarlo y sugerir modificaciones de los requisitos.

El prototipo servirá, principalmente, para modelar y poder mostrar al cliente cómo va a realizarse la entrada y salida de datos en la aplicación, de modo que éste pueda hacerse una idea será el sistema final, pudiendo entonces detectar deficiencias o errores en la especificación aunque el modelo no sea más que eso.

La construcción de un prototipo comienza con la realización de un modelo del sistema, a partir de los requisitos que se conocen. No es necesario realizar una definición completa de los requisitos, pero sí es conveniente determinar, al menos, las áreas donde será necesaria una definición posterior más detallada. Luego se procede a diseñar el prototipo. Se tratará de un diseño rápido, centrado sobre todo en la arquitectura del sistema y la definición de la estructura de las interfaces más que en aspectos procedimentales de los programas, se hará hincapié más en la forma y en la apariencia que en el contenido.

A partir del diseño preliminar se construirá el prototipo. Existen herramientas especializadas en generar prototipos ejecutables a partir del diseño. Otra opción sería utilizar los lenguajes de cuarta generación. En cualquier caso, el objetivo es siempre que la codificación sea rápida, aunque sea en detrimento de la calidad del software generado.

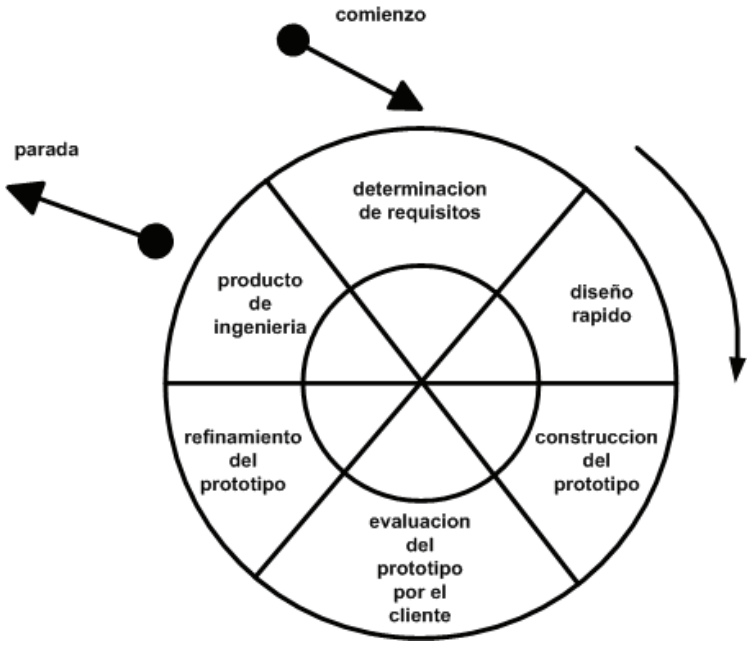


Figura 4.3: El modelo de prototipos

Una vez que el prototipo está terminado, en una etapa inicial, se lo presenta al usuario para que lo pruebe y sugiera modificaciones. En este punto el usuario puede ver una implementación de los requisitos que ha definido inicialmente y sugerir las modificaciones necesarias en las especificaciones para que satisfagan mejor sus necesidades.

A partir de los comentarios del usuario y los cambios que se consideren necesarios en los requisitos, se procederá a construir un nuevo prototipo y así sucesivamente hasta que éstos queden totalmente formalizados y se pueda entonces empezar con el desarrollo del producto final.

Por lo tanto, el prototipo es una técnica que sirve, principalmente, para la fase de análisis de requisitos, pero lleva consigo la obtención de una serie de subproductos que serán útiles a lo largo del desarrollo del proyecto. No obstante, hay que tener en cuenta que el prototipo no es el sistema final, ya que por sus características de desarrollo, normalmente, apenas es utilizable. Será demasiado lento, demasiado grande, ineficiente, inadecuado para el volumen de datos necesario, contendrá errores, principalmente debido al diseño rápido, será demasiado general o estará codificado en un lenguaje o para una máquina inadecuada, o a partir de componentes de software previamente existente.

De todas maneras, no hay que preocuparse por el tiempo y el esfuerzo realizado construyendo prototipos que luego habrán de ser desechados, si con ello se ha conseguido tener más clara la especificación del sistema, puesto que el tiempo perdido se ahorrará en las fases siguientes, que podrán realizarse con menos esfuerzo y en las que se cometerán menos errores que obliguen a volver atrás en el ciclo de vida.

4.5.1. LOS PROBLEMAS CON EL PROTOTIPO

Uno de los principales problemas con este método es que, con demasiada frecuencia, el prototipo pasa a ser parte del sistema final, ya sea por presiones del cliente, que quiere tener el sistema funcionando lo antes posible o bien porque los desarrolladores se han acostumbrado a la computadora, el sistema operativo o el lenguaje con el que se lo desarrolló. Se olvida aquí que el prototipo ha sido construido de forma acelerada, sin tener en cuenta consideraciones de eficiencia, calidad del software o facilidad de mantenimiento.

Al utilizar el prototipo como producto final conducirá a que éste contenga numerosos errores latentes, además de ser ineficiente, poco fiable, incompleto o difícil de mantener. En definitiva a que tenga poca calidad, y eso es precisamente lo que se quiere evitar aplicando la ingeniería del software.

Esto normalmente sucede por que el cliente ve lo que parece ser una versión en funcionamiento del software, sin tener conocimiento de que el prototipo es una estructura frágil e inestable, sin considerar que en el apuro por obtenerlo, no se tuvieron en cuenta temas fundamentales como calidad total del software o mantenibilidad a largo plazo.

A pesar de que pueden ocurrir problemas, el prototipo es un método efectivo de ingeniería de software. El usuario y el desarrollador deben coincidir en que el prototipo es construido para servir como un mecanismo cuyo propósito, principalmente, es la definición de los requerimientos.

4.5.2. EL ÁMBITO DE APLICACIÓN DEL PROTOTIPO

El prototipo provee una retroalimentación para evaluar y desarrollar nuevos requerimientos. El uso del ciclo de vida por prototipos es válido cuando el usuario no está dispuesto a examinar modelos abstractos en papel o no puede o no está dispuesto a especificar sus requerimientos de ninguna forma y sólo se pueden determinar mediante un proceso de tanteo, o ensayo y error. O el usuario se preocupa más por el formato y distribución de los datos de entrada y salida en la pantalla y por los mensajes de error, que por los cómputos que realiza el sistema para lograrlo.

También pueden utilizarse cuando se tiene intención de que el sistema sea en línea y con operación total por la pantalla. O cuando el sistema no requiere de especificación de grandes cantidades de detalles algorítmicos, ni de muchas especificaciones de procesos para describir algoritmos con los cuales se obtienen resultados.

4.6. EL MODELO EN ESPIRAL

El problema con los modelos de desarrollo de sistemas es que no tienen en cuenta la incertidumbre inherente a los procesos de software. Importantes proyectos de sistemas fallaron porque los riegos del proyecto se despreciaron sin estar nadie preparado para algún imprevisto.

Barry Boehm reconoció esto y trató de incorporar el factor de “riesgo del proyecto” al modelo de ciclo de vida, agregándoselo a las mejores características de los modelos de Cascada y de Prototipo. El resultado fue el Modelo en Espiral, presentado en 1986. La característica principal de la nueva propuesta fue incorporar los puntos fuertes y evitar las dificultades de otros modelos, desplazando el énfasis de la administración hacia la evaluación y resolución del riego. De esta manera se permite tanto a los desarrolladores como a los clientes detener el proceso cuando se lo considere conveniente.

El modelo en espiral proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos, mucho más realista que el ciclo de vida clásico y permite la utilización de prototipos en cualquier etapa de la evolución del proyecto.

Una característica distintiva de este modelo es la incorporación, en el ciclo de vida, del análisis de riesgos, permitiendo finalizar el proyecto antes de haberse embarcado en el desarrollo del producto final, si el riesgo es demasiado grande.

El modelo en espiral define, en su versión original, cuatro tipos de actividades. La planificación, el análisis de riesgo, la ingeniería y la evaluación del cliente. (figura 4.4).

4.6.1. LA PLANIFICACIÓN

La planificación consiste en la determinación de los objetivos del proyecto, las posibles alternativas y las restricciones. Esta fase equivale a la de recolección de requisitos del ciclo de vida clásico e incluye, además, la planificación de las actividades a realizar en cada iteración.

4.6.2. EL ANÁLISIS DE RIESGO

Una de las actividades de la gestión de proyectos es el análisis de riesgos. El desarrollo de cualquier proyecto complejo lleva implícito una serie de riesgos, unos relativos al propio proyecto, esto es, los riesgos que pueden hacer que el proyecto fracase y otros relativos a las decisiones que deben tomarse durante su desarrollo, es decir, los riesgos asociados a que una de estas decisiones sea errónea.

El análisis de riesgos consiste en cuatro actividades principales.

Identificar los riesgos. Pueden ser riesgos del proyecto, por ejemplo presupuestarios, de organización, del cliente, etc.; riesgos técnicos, por ejemplo problemas de diseño, codificación, mantenimiento; riesgos del negocio, por ejemplo riesgos de mercado, en donde podría suceder que se adelante la competencia o que el producto no se venda bien.

Estimación de riesgos. Consiste en evaluar, para cada riesgo identificado, la probabilidad de que ocurra y sus consecuencias, es decir, el costo que tendrá en caso de que suceda.

Evaluación de riesgos. Consiste en establecer niveles de referencia para el incremento de costo, de duración del proyecto y para la degradación de la calidad que, si se superan, harán que se interrumpa el proyecto. Luego hay que relacionar cuantitativamente cada uno de los riesgos con estos niveles de referencia, de forma que en cualquier momento del proyecto se pueda calcular si se ha superado alguno de los niveles de referencia establecidos.

Gestión de riesgos. Consiste en supervisar el desarrollo del proyecto, de forma tal que se detecten los riesgos tan pronto como aparezcan, se intenten minimizar sus daños y que exista, además, un apoyo previsto para las tareas críticas, es decir, aquéllas que más riesgo encierran.

4.6.3. LA INGENIERÍA

En esta etapa se desarrolla el sistema o un prototipo del mismo.

4.6.4. LA EVALUACIÓN DEL CLIENTE

Consiste en la valoración, por parte del cliente, de los resultados de la ingeniería. En la primera iteración se definen los requisitos del sistema y se realiza la planificación inicial del mismo. A continuación se analizan los riesgos del proyecto, basándose en los requisitos iniciales y se procede a construir un prototipo del sistema. Entonces el cliente procede a evaluar el prototipo y, a partir de sus comentarios, se refinan los requerimientos y se reajusta la planificación inicial, volviendo a empezar el ciclo.

En cada una de las iteraciones se realiza el análisis de riesgos, teniendo en cuenta los requisitos y la reacción del cliente ante el último prototipo. Si los riesgos son demasiado grandes se terminará el proyecto, aunque lo normal es que se siga avanzando a lo largo de la espiral.

Con cada iteración, se construyen sucesivas versiones del software, cada vez más completas, y aumenta la duración de las operaciones del cuadrante de ingeniería, obteniéndose, al final, el sistema completo.

La diferencia principal con el modelo de construcción de prototipos, es que en éste los prototipos se usan para perfilar y definir los requisitos. Al final, el prototipo se desecha y comienza el desarrollo del software siguiendo el ciclo de vida clásico. En el modelo en espiral, en cambio, los prototipos son sucesivas versiones del producto final, cada vez más detalladas, siendo la última versión el producto en sí, constituyéndose en el esqueleto del producto final. Por lo tanto, los prototipos deben construirse siguiendo estándares de calidad.

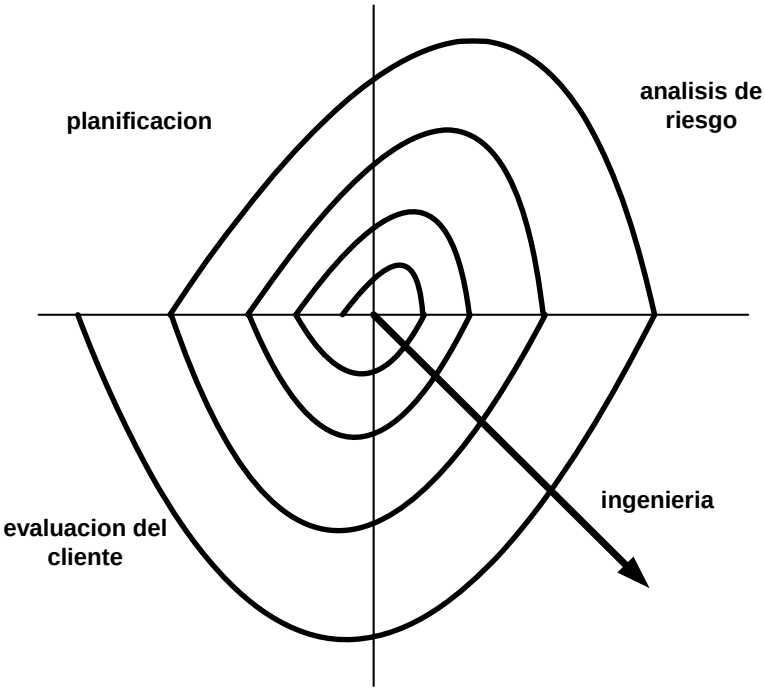


Figura 4.4: El modelo en espiral

4.6.5. LAS CARACTERISTICAS DEL CICLO DE VIDA EN ESPIRAL

El ciclo de vida en espiral tiene una serie de características que lo distinguen de los demás modelos. Por una lado, evita las dificultades de los modelos existentes a través de un proceso conducido por el riesgo, intentando eliminar errores en las fases tempranas. Además, es el mismo modelo para el desarrollo y el mantenimiento. Provee mecanismos para asegurar la calidad del software. Se adapta bien a proyectos complejos, dinámicos e innovadores. La evaluación, después de cada fase, permite cambios en las percepciones de los usuarios, avances tecnológicos o perspectivas financieras. Y, por último, la focalización en los objetivos y limitaciones ayuda a asegurar la calidad.

4.6.6. LOS PROBLEMAS CON EL CICLO DE VIDA EN ESPIRAL

Los problemas en este modelo están relacionados con la falta de un proceso de guía explícito para determinar objetivos, limitaciones y alternativas. Provee, además, más flexibilidad que la conveniente para la mayoría de las aplicaciones. La pericia para la tasación del riesgo no es una tarea fácil, además, es necesaria mucha experiencia en proyectos de software para realizar esta tarea exitosamente.

4.6.7. EL ÁMBITO DE APLICACIÓN DEL CICLO DE VIDA EN ESPIRAL

El paradigma del modelo en espiral para la ingeniera del software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala. El ámbito de aplicación apropiado de este modelo es en proyectos complejos, dinámicos e innovadores.

4.7. LA SELECCIÓN DE UN CICLO DE VIDA

Los modelos presentados suministran una guía que permite ordenar las diversas actividades técnicas en el proyecto de desarrollo de software e intentan suministrar un marco para la administración en el desarrollo y el mantenimiento.

Entre los criterios utilizados para la elección de un modelo en particular se encuentran la madurez de la aplicación, relacionado a la probabilidad de que muchos requerimientos comenzarán a conocerse sólo después del uso del sistema; la complejidad del problema y de la solución; las frecuencias y magnitudes esperadas de los cambios de requerimientos; el financiamiento disponible; el acceso de los desarrolladores a los usuarios y la certeza de requerimientos conocidos.

Considerando la importancia de la planificación, se recomienda realizar el desarrollo de un proyecto de software bajo el modelo en espiral insertando en él, cualquier otro modelo que se requiera dependiendo de las necesidades que se presenten. Esto permite realizar una planificación del proceso de desarrollo del software considerando los riesgos asociados en cada etapa identificada. El identificar los riesgos en proyectos, evaluar su impacto, monitorear y controlar el avance del desarrollo del mismo, permite al administrador aumentar las posibilidades de éxito de un proyecto o minimizar las posibilidades de fracaso de éste.

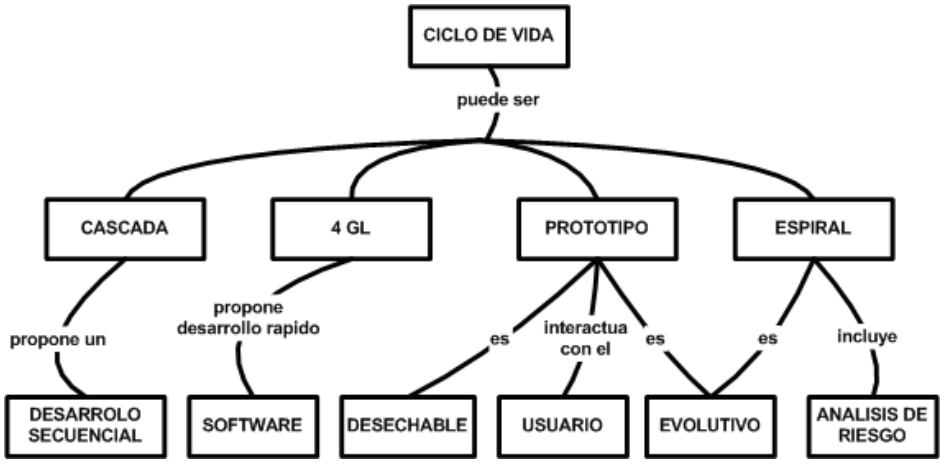


Figura 4.5: Mapa conceptual “Tipos de Ciclo de Vida”

Uno de los factores que más influyen en el proceso de desarrollo de software y que prácticamente acompaña a toda aplicación es el hecho de que en su mayoría, no hay forma de tener todos los requerimientos corregidos antes del desarrollo del software. El prototipo es ampliamente recomendado para realizar la especificación de requerimientos, se debe notar que la idea del prototipo es capturar por retroalimentación los objetivos, necesidades y expectativas del cliente por lo cual no se debe caer, como ya se señaló, en una utilización de éstos como partes finales del sistema, ya que en su desarrollo generalmente no se consideran aspectos de calidad, ni otros asociados con facilitar la etapa de mantenimiento del sistema. El prototipo trata de minimizar los cambios en los requerimientos, mientras que el diseño modular trata de minimizar el impacto de los cambios en los requerimientos.

El cambio es una propiedad intrínseca del software. Hoy en día el software debe poseer un enfoque evolutivo, un sistema debe evolucionar para acomodar la naturaleza evolutiva de los grandes sistemas. El software cambia constantemente, debido a la necesidad de reparar el software, eliminando errores no detectados anteriormente, como a la necesidad de apoyar la evolución de los sistemas a medida que aparecen nuevos requerimientos o cambian los antiguos.

En muchos casos, los modelos pueden y deben combinarse de forma que puedan utilizarse las ventajas de cada uno en un único proyecto. El paradigma del modelo en espiral lo hace directamente, combinando la creación de prototipos y algunos elementos del ciclo de vida clásico, en un enfoque evolutivo para la ingeniería de software.

No hay necesidad por lo tanto, de ser dogmático en la elección de los paradigmas para la ingeniería de software, la naturaleza de la aplicación debe dictar el método a elegir.

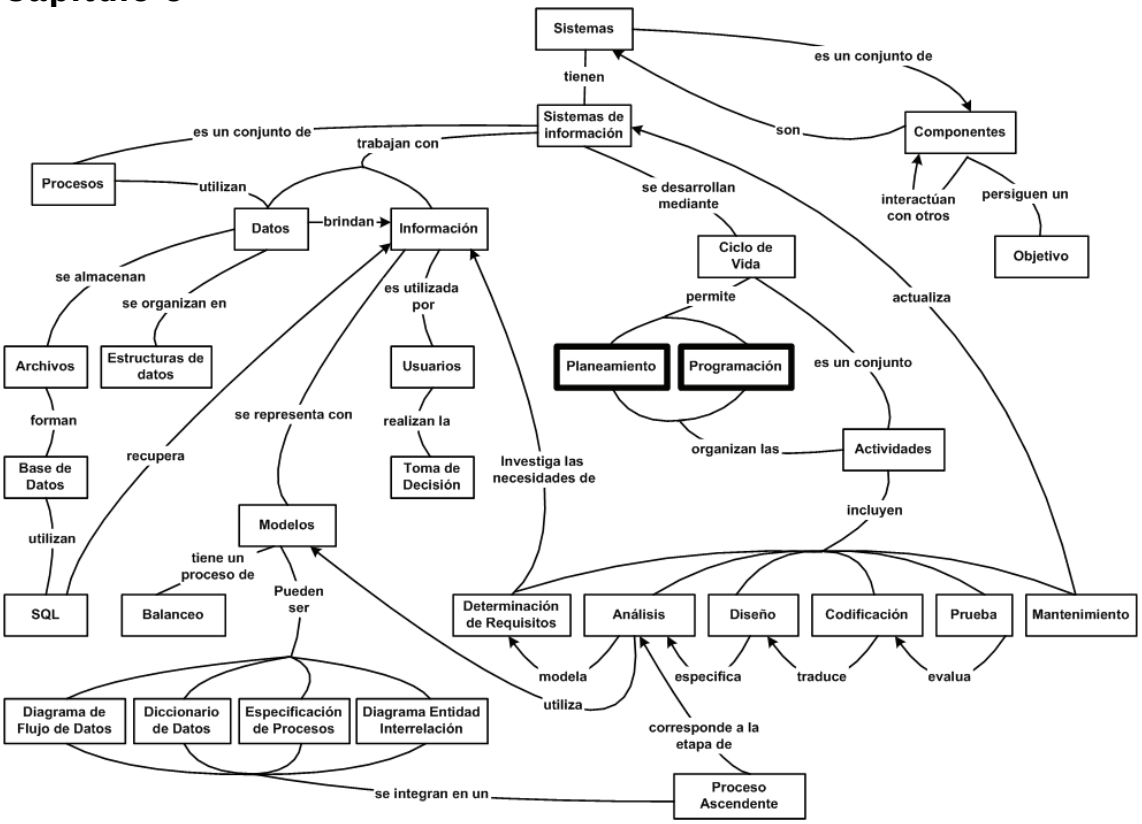
4.8. RESUMEN

El ciclo de vida de desarrollo de sistemas es una sucesión de etapas por las que transita el software desde que comienza un nuevo proyecto hasta que éste se deja de utilizar. Existen diversos modelos de ciclo de vida, cada uno de ellos propone distintas maneras de enfocar el proceso de desarrollo de software y va asociado a un paradigma de la ingeniería del software, es decir, a una serie de métodos, herramientas y procedimientos que se debe utilizar a lo largo de un proyecto.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 5



Planeamiento y Programación de Proyectos

Un proyecto, cualquiera sea su característica, se puede considerar como la sucesión de un conjunto de tareas interrelacionadas que deben ejecutarse en un orden determinado y con el fin de alcanzar un objetivo. Cuando se emprende la realización de un proyecto se reconoce en su desarrollo tres etapas bien diferenciadas: el planeamiento, la programación y el control.

El planeamiento establece qué debe hacerse y en qué secuencia. La programación determina cuándo debe hacerse, esto es, acota en el tiempo lo pla-

neado. Y, por último, el control se encarga de verificar si se cumple con lo planeado y lo programado anteriormente.

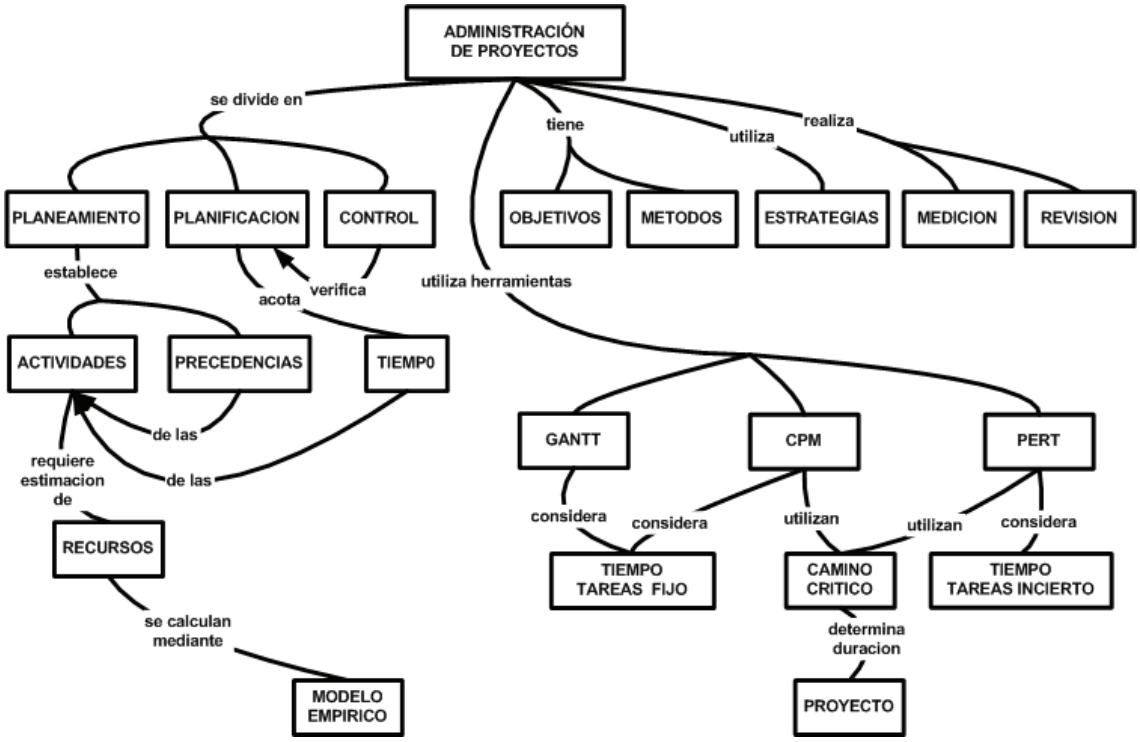


Figura 5.1: Mapa conceptual de “Planeamiento y Programación de Proyectos”

El análisis y diseño de sistemas de información involucra una serie de actividades diversas que al integrarse constituyen un proyecto informático. Los recursos críticos que se utilizarán no siempre estarán disponibles en el momento en que se los requiere. Esto incluye a los miembros importantes del personal que, por distintas razones, pueden no estar disponibles y la estimación original de la cantidad de trabajo necesario que es muy difícil de mensurar.

El administrador no sólo administra tareas sino personas. Debe asegurarse, por lo tanto, que todos los analistas, diseñadores y programadores estén realizando lo que deben hacer cuando deban hacerlo.

Por tal motivo, se precisan técnicas que permitan distribuir los esfuerzos en el tiempo, a fin de concluir exitosamente el proyecto en el momento proyectado y con los recursos previstos.

5.1. LA ADMINISTRACIÓN DE PROYECTOS

La administración de proyectos consiste en la planificación, la programación y el control de los recursos humanos, técnicos y económicos para alcanzar un objetivo. Esto se logra a través de una organización específica, de la definición de roles, funciones y responsabilidades de todos los miembros participantes, del uso de herramientas y de la aplicación de técnicas adecuadas de estimación de recursos, de la programación de las actividades, de la distribución de recursos en el tiempo y de la comparación entre la ejecución real y la programada.

5.2. LA PLANIFICACIÓN

Una planificación adecuada de un proyecto tiene características distintivas. Debe ser creativa en la utilización de herramientas y en la adaptación de las mismas a los requerimientos del proyecto. Tiene que ser flexible al observar el impacto del cambio y adaptarse a él. Debe ser monitoreada constantemente mediante controles. Además, debe ser analítica, esto es, explorar los factores externos e internos que permitan identificar variables e incertidumbres.

En cambio, un proyecto mal planificado se lo reconoce, entre otras cosas, por un inicio inapropiado, por la pérdida del entusiasmo de los responsables, por la confusión e incertidumbre y por la búsqueda de los culpables.

Los pasos a seguir para la realización de una planificación adecuada, incluyen la determinación de los trabajos a realizar, la estructuración de los mismos, la estimación del esfuerzo requerido, la programación de la secuencia lógica, la determinación de las duraciones sin tener en cuenta las restricciones y la administración de los recursos a partir de las limitaciones.

5.3. EL DIAGRAMA DE GANTT

El diagrama de Gantt es una de las técnicas más simples utilizadas en la administración de proyectos y consiste en representar las tareas por medio de barras, cuyas longitudes son proporcionales a la duración de las tareas. Fue uno de los intentos de mayor aceptación para el seguimiento de un proyecto y aún se lo sigue utilizando como complemento de los métodos posteriores. Como ventaja principal, se destaca la sencillez y la facilidad de comprensión al integrar gráficamente la planificación, la programación y el progreso del proyecto. Permite visualizar rápidamente los elementos principales, su programación en el tiempo y, además, el progreso en cada uno de ellos.

Como desventaja se puede mencionar que no permite determinar el impacto del atraso o adelanto de alguna de las tareas sobre la fecha de terminación del proyecto, esto es debido a que las tareas no están interrelacionadas, el nivel de detalle no es suficiente para detectar a tiempo los retrasos en el programa

y no permite la evaluación de diferentes alternativas. Además, se complica su lectura por su extensión en proyectos de envergadura.

Para su realización se debe estructurar el proyecto en sus principales tareas, estimar el tiempo requerido para efectuar cada una de ellas, listar las tareas y representar su duración por una barra o rectángulo teniendo en cuenta la secuencia de cada una de ellas indicando, periódicamente, el progreso de cada actividad.

En forma práctica, una vez establecidas las tareas, su duración y precedencias, se grafica, a partir de la primera tarea y de acuerdo a su precedencia y en longitud proporcional a su duración, todas las demás actividades en un grafico bidimensional. En el eje de las abscisas se establece una escala temporal y en el de las ordenadas las tareas.

A partir del siguiente ejemplo (tabla 5.1), se ilustrará la realización de un diagrama de Gantt (figura 5.2). La simplicidad de su construcción ahorra innecesarias explicaciones.

Actividad	Duración	Precedencia
A	2	-----
B	3	----
C	1	B
D	1	A, C
E	2	D
F	1	E

Tabla 5.1: Listado de actividades

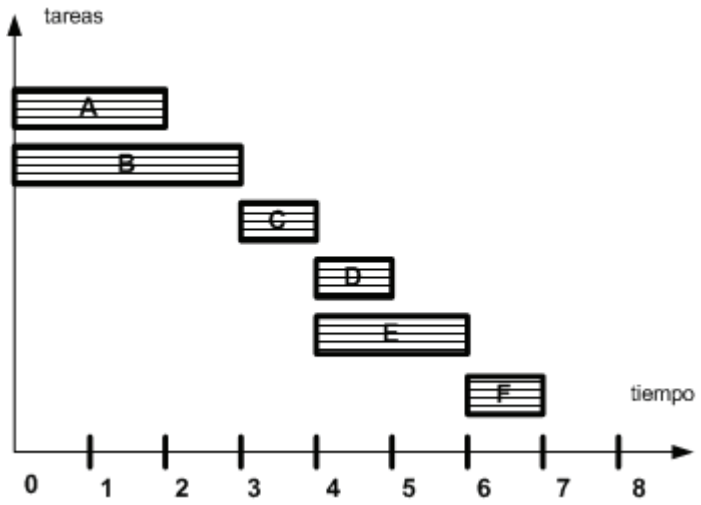


Figura 5.2: Diagrama de Gantt

5.4. EL MÉTODO DEL CAMINO CRÍTICO

El método de Gantt tiene como principal ventaja su simplicidad pero en proyectos más complejos es necesaria una mayor información. Se han propuesto otros métodos para la administración de proyectos, las técnicas llamadas PERT (Program Evaluation and Review Technique) y CMP (Critical Path Method) permiten mayor detalle en el seguimiento de un proyecto que el diagrama de Gantt.

En general, estas técnicas resultan útiles para una gran variedad de proyectos que contemplen la investigación y el desarrollo de nuevos productos y procesos, el diseño e implantación de sistemas informáticos, etc.

En proyectos con las características antes detalladas, los administradores deben programar y coordinar las diversas tareas, de manera que se pueda terminar a tiempo el proyecto completo. Por lo general las actividades a desarrollar en un proyecto no son necesariamente secuenciales, y aún en ese caso estas actividades son interdependientes. Si bien es cierto que algunas tareas dependen de la terminación de otras para su inicio, se puede dar el caso de actividades en paralelo que originan el inicio de una tercera.

Las preguntas esenciales en la elaboración de un proyecto son:
¿Cuál es el tiempo total que se requiere para terminar el proyecto?
¿Cuáles son las fechas programadas de inicio y de terminación para cada actividad específica?
¿Qué actividades son "críticas" y deben terminarse exactamente según lo programado para poder mantener el proyecto dentro del programa?
¿Cuánto se pueden demorar las actividades "no críticas" antes de que ocasionen demoras en el proyecto total?

Las técnicas en administración de proyectos, denominadas CPM y PERT, ayudan a responder las preguntas anteriores.

Aunque ambas técnicas tienen el mismo propósito general y utilizan en buena medida la misma terminología, se desarrollaron en forma independiente. Se presentó el método PERT a finales de los años 1950 para especificar las tareas de planear, programar y controlar el proyecto de desarrollo de los misiles Polaris. Como muchas de las tareas relacionadas con el proyecto nunca se habían considerado antes, resultaba difícil pronosticar el tiempo necesario para terminar las diversas tareas. En consecuencia, se desarrolló la técnica PERT con el objetivo de permitir administrar las incertidumbres en los tiempos de terminación de las actividades.

Por otro lado, el CPM se ideó para desarrollar y controlar proyectos industriales en donde se consideraba que se conocían los tiempos de las tareas o actividades.

5.4.1. LA MALLA CPM

La primera fase del proceso de programación de proyectos con CPM consiste en determinar las tareas específicas que lo conforman.

Se ilustrará la construcción del diagrama con un ejemplo referido a un proyecto de desarrollo de sistemas. Se considerará una lista simple de cuatro actividades que se muestran en la tabla 5.2.

Actividad	Descripción	Duración	Precedencia
A	Determinación de requerimientos	2	-----
B	Análisis	3	-----
C	Diseño	1	B
D	Codificación	1	A, C

Tabla 5.2: Listado de actividades

La precedencia de una actividad específica son todas las actividades que, cuando terminan, permiten el inicio de la actividad en cuestión. Por ejemplo, en la lista de actividades anterior se señala que se puede comenzar el trabajo correspondiente a las actividades A y B en cualquier momento, puesto que ninguna tarea depende de la terminación de las actividades anteriores. Sin embargo, no puede iniciarse la actividad C hasta que haya terminado la activi-

dad B, y no puede iniciarse la actividad D hasta que hayan concluido las actividades A y C. Como se verá, es necesario conocer la información sobre los antecedentes inmediatos de cada actividad, con objeto de describir las interdependencias entre las tareas del proyecto.

A continuación se dibuja una malla (figura 5.3) que, no sólo ilustra las actividades que se listan en la tabla anterior, sino que también muestra las relaciones de precedencia entre las tareas. A esta representación gráfica se le denomina malla CPM para el proyecto. A diferencia de los gráficos del método de Gantt, la longitud del arco o línea que una dos nodos es independiente de la duración de la tarea que representa.

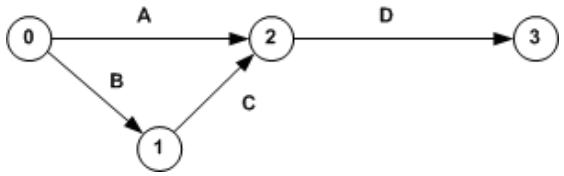


Figura 5.3: Malla CPM

Actividad	Duración	Precedencia
A	1	----
B	2	----
C	1	B
D	2	A, C
E	2	C
F	3	C
G	1	D, E, F

Tabla 5.3: Listado de actividades

Los nombres de las actividades, en este caso letras, se muestran sobre los arcos de la malla. Los círculos o nodos de la malla, corresponden al inicio o terminación de las actividades. La conclusión de todas las actividades que conducen a un nodo se denomina evento. Por ejemplo, el nodo 1 de la figura 5.3, corresponde al evento de la terminación de la actividad B y el nodo 2 corresponde al evento de la terminación de la actividad A y C.

Se elaborará una malla para un proyecto genérico que tiene las siguientes actividades (tabla 5.3), con sus respectivos antecedentes y duraciones.

Una fragmento de la malla CPM que podría utilizarse para las primeras cuatro actividades es la siguiente (figura 5.4):

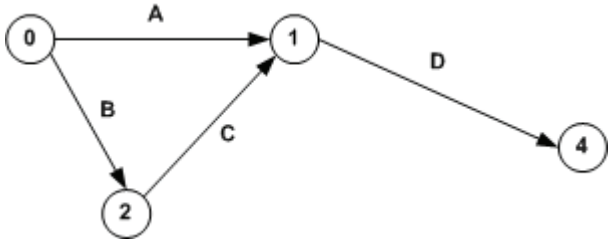


Figura 5.4: Malla CPM

Esta porción de malla no ocasiona ningún problema específico para la actividad D debido a que muestra, correctamente, que las actividades A y C son los antecedentes inmediatos. Sin embargo, cuando se intenta añadir la actividad E a la malla (figura 5.4), se presenta un problema. En primer lugar, se podría intentar incluir la actividad E comenzando en el nodo 1. Sin embargo, esto significaría que las actividades A y C son antecedentes inmediatos para la actividad E, lo cual es incorrecto. Con referencia al programa original de actividades para el proyecto, se observa que el único antecedente inmediato de la actividad E es la actividad C.

Se puede evitar el problema anterior incluyendo una actividad ficticia la cual, como su nombre lo indica, no es una actividad real, sino una actividad supuesta que tiene duración nula y sólo sirve para asegurar que se grafica en la malla las relaciones adecuadas de precedencia entre las actividades. Por ejemplo, puede añadirse el nodo 3 e insertar una actividad ficticia, señalada mediante una línea punteada del nodo 3 al nodo 1, lo cual conforma la malla que se muestra a continuación (figura 5.5):

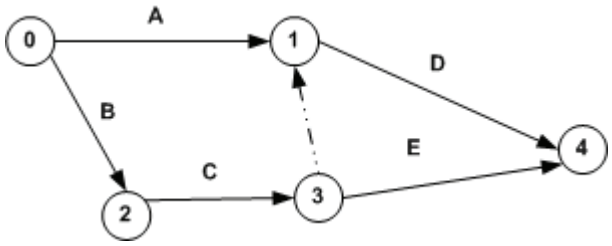


Figura 5.5: Malla CPM

Haciendo este cambio en la malla, y comenzando la actividad E en el nodo 3, se tiene que su única actividad antecedente es la C, lo cual es correcto. La actividad ficticia no tiene requisito de tiempo, porque sólo se utiliza para conservar las relaciones adecuadas de precedencia en la malla. Obsérvese que la inclusión de la actividad ficticia muestra también correctamente que las actividades A y C son antecedentes inmediatos de la actividad D.

La terminación de la malla de siete actividades se puede ilustrar de la siguiente manera (figura 5.6):

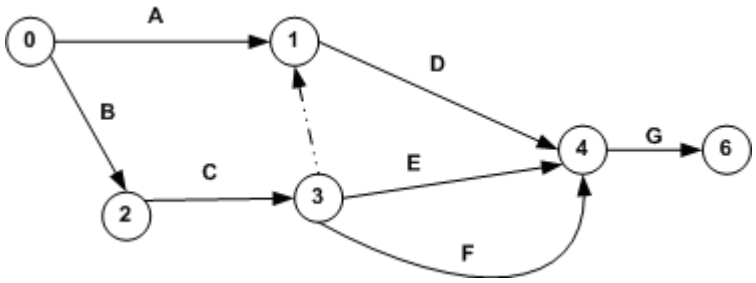


Figura 5.6: Malla CPM

Nótese la forma en que la malla identifica correctamente que las actividades D, E y F son los antecedentes inmediatos de la actividad G (figura 5.6). Sin embargo, las actividades E y F comienzan, ambas, en el nodo 3 y terminan en el nodo 4. Esta situación ocasiona problemas para ciertos programas de computación que utilizan nodos inicial y final para identificar las actividades de una malla CPM. En estos programas, el procedimiento computacional consideraría que las actividades E y F son la misma, puesto que tienen los mismos nodos inicial y final. Cuando se presenta esta condición, pueden añadirse actividades ficticias a la malla para asegurarse de que ningún par de actividades tengan los mismos nodos inicial y final. Utilizando el nodo 5 y una actividad ficticia (figura 5.7), se elimina este problema para las actividades E y F.

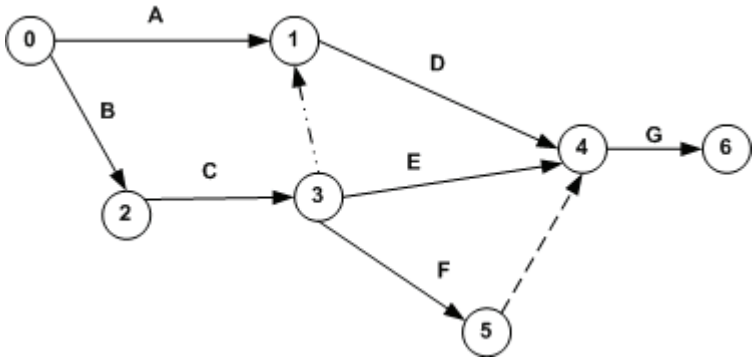


Figura 5.7: Malla CPM

Se pueden utilizar actividades ficticias para identificar en forma correcta las relaciones de precedencia así como también eliminar la posible confusión que pudiera ocasionar el hecho de que dos o más actividades tengan los mismos nodos inicial y final. Aunque es posible que no se requieran actividades ficticias para todas las mallas CPM, los proyectos más grandes o más complejos

pueden requerir de muchas de estas actividades para poder esquematizar o trazar en forma apropiada la malla CPM.

5.4.2. EL CAMINO CRÍTICO

Dentro de la malla CPM existen diversos caminos, estos recorridos están formados por las distintas tareas que van desde la tarea correspondiente al nodo inicial, hasta las que concluyen en el nodo final. Por ejemplo, los caminos A-D-G; B-C-E-G; B-C-D-G y B-C-F-G de la figura 5.7. Existen en la malla CPM uno o más de estos caminos, denominados críticos, en donde un retraso en cualquiera de las tareas que pertenecen a esa ruta implicará un inevitable retraso de todo el proyecto.

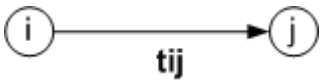
Se definirán algunos conceptos para la determinación del camino crítico.

5.4.2.1. LA FECHA TEMPRANA DE UN SUCESO

Un suceso indica la finalización de una actividad y con ello la posibilidad de comenzar la siguiente tarea, estos sucesos o nodos no insumen tiempo, son instantáneos. En cambio las tareas consumen un tiempo determinado para su ejecución.

Un suceso no puede comprobarse antes de que finalicen todas las tareas que lleguen a él. Puede suceder, sin embargo, que no todas las tareas que concurren a un nodo finalicen al mismo tiempo. El suceso, entonces, se verifica en el momento en que finaliza la tarea concurrente a él que finalice última. Las fecha de ocurrencia de un suceso indican cuando pueden comenzar la o las tareas siguientes a ese nodo. Se dice que pueden comenzar pero no, necesariamente, que deban comenzar. Esa fecha se la denomina fecha temprana de un suceso.

La determinación de la fecha temprana del nodo destino, se calcula sumando a la fecha temprana del nodo origen la duración de la tarea (figura 5.8).



$$Tej = Tei + tij$$

Figura 5.8: Fecha temprana de un suceso

Siendo Tej, la fecha temprana del nodo destino; Tei, la fecha temprana del nodo origen y tij, la duración de la tarea. Para el nodo inicial, la fecha temprana es cero.

Si existe sólo una tarea que concurre a ese nodo, el valor del cálculo resulta la fecha temprana. Si hay más de una tarea concurrente al nodo, la fecha temprana será la mayor de todas las que concurren a ese nodo. Esto es debido a que la fecha temprana de un suceso es aquella en la que concluyen todas las tareas que concurren a él, en consecuencia debe tomarse la mayor.

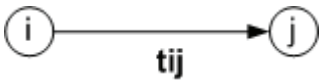
5.4.2.2. LA FECHA TARDÍA DE UN SUCESO

La verificación de un suceso implicaba la finalización de una o más tareas y ese, por consiguiente, era el momento en que podrían comenzar la o las tareas siguientes, ese instante se denomina fecha temprana de un suceso. Pero, si no comienzan la o las tareas en ese momento, ¿hasta cuándo puede retrazarse su comienzo sin alterar la fecha de finalización del proyecto?.

Es evidente que debe existir una fecha en la que, inevitablemente, debe comenzar una tarea para que no se alargue la duración total del proyecto, a esa fecha se la denomina fecha tardía de un suceso o nodo.

La determinación de la fecha tardía de un nodo se calcula restando a la fecha tardía del nodo destino la duración de la tarea (figura 5.9).

Siendo Tai, la fecha tardía del nodo origen; Taj, la fecha tardía del nodo destino y tij, la duración de la tarea. Se toma como fecha tardía del suceso final del proyecto su fecha temprana. Es decir, que en el suceso final del proyecto coinciden la fecha temprana y la fecha tardía. A partir de ahí se va retrocediendo hasta el nodo inicial de la malla. En modo inverso a la determinación de la fecha temprana de un suceso, si de un nodo parte más de una tarea, para el cálculo de la fecha tardía del suceso se utiliza el valor menor.



Tai = Taj - tij

Figura 5.9: Fecha tardía de un suceso

5.4.2.3. EL MARGEN TOTAL

Las fechas tardías y las tempranas se calculan para los nodos o sucesos. En cambio, el margen total se calcula para las tareas y se determina restándole a la fecha tardía del nodo destino la fecha temprana del nodo origen y la duración de la tarea (figura 5.10)

$$MT = Taj - Tei - tij$$

Figura 5.10:Margen total

El margen total establece, para cada tarea, la flexibilidad que tiene esa actividad con relación al retrazo en su comienzo. Cuando el margen total es cero, esa tarea se la denomina critica, cualquier retrazo en esas actividades, retraza irremediablemente el proyecto. Por tal motivo, es en ese conjunto de tareas donde la atención debe ser dirigida constantemente. A partir de un ejemplo se detallará la construcción del camino crítico (Tabla 5.4).

Se determinará, a partir del nodo inicial de la malla CPM del ejemplo, las fechas tempranas para cada suceso (figura 5.11)

Suceso 0 : $Te_0 = 0$
Suceso 1 : $Te_1 = Te_3 + t_{31} = 3 + 0 = 3$
Suceso 2 : $Te_1 = Te_0 + t_{02} = 0 + 2 = 2$
Suceso 3 : $Te_1 = Te_2 + t_{23} = 2 + 1 = 3$
Suceso 4 : $Te_1 = Te_5 + t_{54} = 6 + 0 = 6$
Suceso 5 : $Te_1 = Te_3 + t_{35} = 3 + 3 = 6$
Suceso 6 : $Te_1 = Te_4 + t_{46} = 6 + 1 = 7$

Figura: 5.11: Fechas tempranas

Actividad	Duración	Precedencia
A	1	----
B	2	----
C	1	B
D	2	A, C
E	2	C
F	3	C
G	1	D, E, F

Tabla 5.4: Listado de Actividades

Se determinará, a partir del nodo final de la malla CPM del ejemplo, las fechas tardías para cada suceso (figura 5.12).

Suceso 0 : $Ta_0 = Ta_2 - t_{02} = 2 - 2 = 0$
Suceso 1 : $Ta_1 = Ta_4 - t_{14} = 6 - 2 = 4$
Suceso 2 : $Ta_2 = Ta_3 - t_{23} = 3 - 1 = 2$
Suceso 3 : $Ta_3 = Ta_5 - t_{35} = 6 - 3 = 3$
Suceso 4 : $Ta_4 = Ta_6 - t_{46} = 7 - 1 = 6$
Suceso 5 : $Ta_5 = Te_4 - t_{45} = 6 - 0 = 6$
Suceso 6 : $Ta_6 = 7$

Figura 5.12:Fechas tardías

El margen total para cada tarea de la malla CPM se muestra en la figura 5.13

Tarea A : $MT_{A\,[01]} = Ta_1 - Te_0 - t_{01} = 4 - 0 - 1 = 3$
Tarea B : $MT_{B\,[02]} = Ta_2 - Te_0 - t_{02} = 2 - 2 - 0 = 0$
Tarea C : $MT_{C\,[23]} = Ta_3 - Te_2 - t_{23} = 3 - 2 - 1 = 0$
Tarea D : $MT_{D\,[14]} = Ta_4 - Te_1 - t_{14} = 6 - 4 - 2 = 0$
Tarea E : $MT_{E\,[34]} = Ta_4 - Te_3 - t_{34} = 6 - 3 - 2 = 1$
Tarea F : $MT_{F\,[35]} = Ta_5 - Te_3 - t_{35} = 6 - 3 - 3 = 0$
Tarea G : $MT_{G\,[46]} = Ta_6 - Te_4 - t_{46} = 7 - 6 - 1 = 0$

Figura 5.13: Margen total

En la figura 5.14, se muestra la malla CPM con las fechas tempranas y tardías para cada nodo

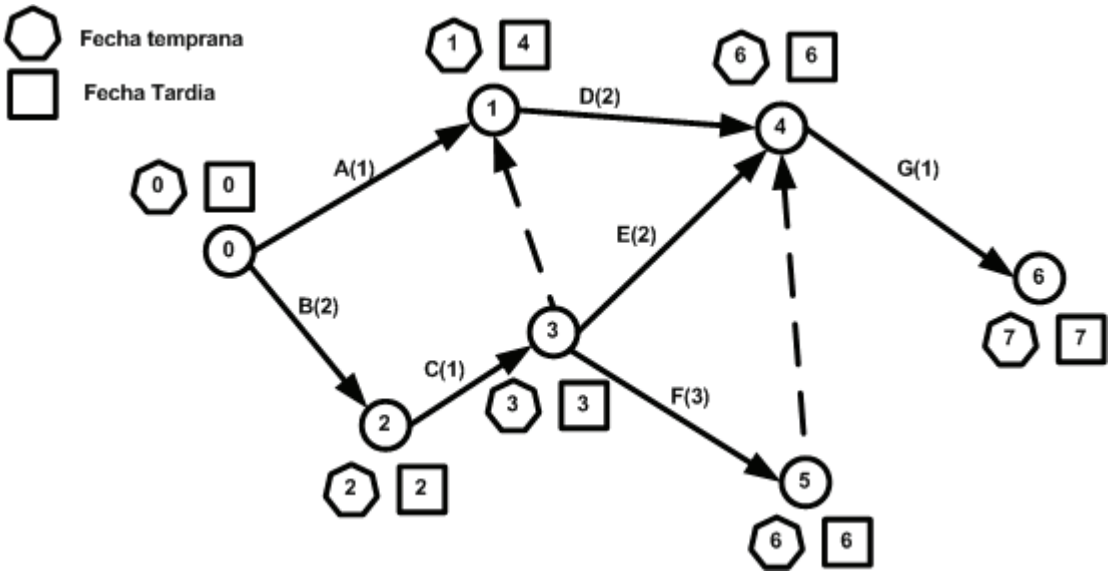


Figura 5.14: Malla CPM del ejemplo

En la figura 5.15, se presenta el camino crítico para el ejemplo, está compuesto por todas las tareas críticas. Las actividades que lo conforman son las B-C-D-F-G.

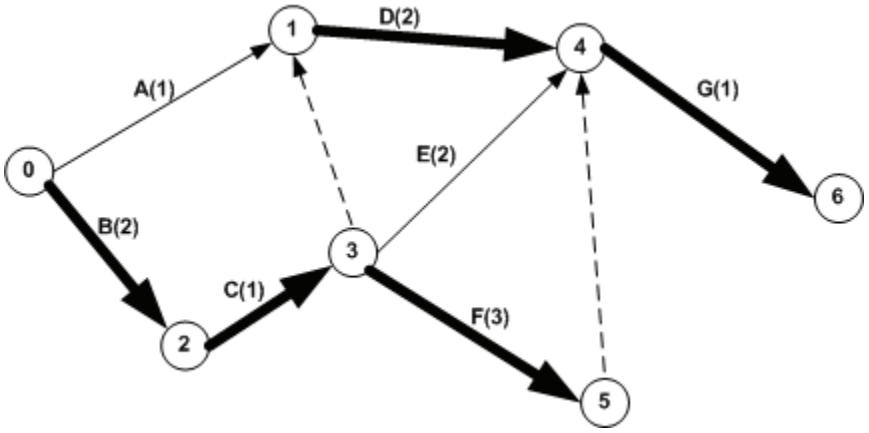


Figura 5.15: El camino crítico

5.5. RESUMEN

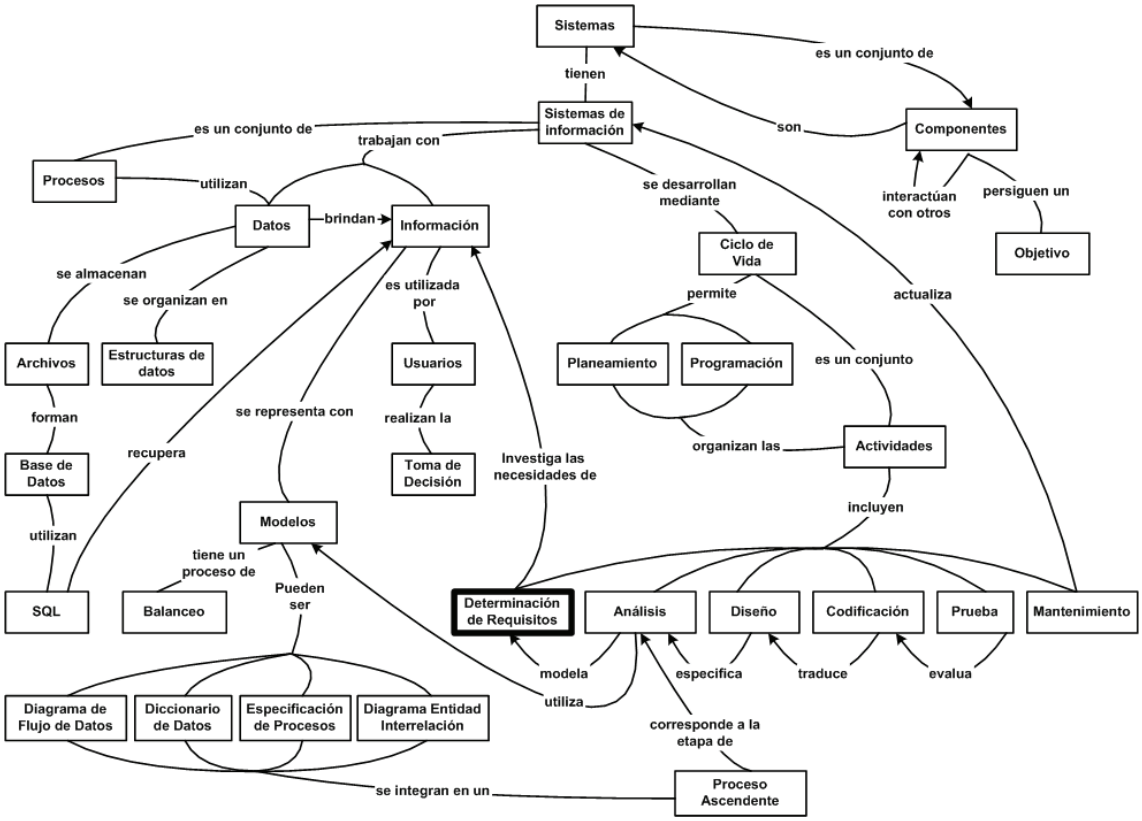
En el análisis y diseño de sistemas de información se distinguen, en su desarrollo, tres etapas bien diferenciadas, el planeamiento, la programación y el control. El planeamiento establece qué tareas deben hacerse y en qué secuencia. La programación determina cuándo debe hacerse. Por último, el control verifica si se cumple con lo planeado y lo programado.

La administración de proyectos utiliza recursos humanos, técnicos y económicos para alcanzar un objetivo. Esto se logra a través de la definición de roles, funciones y responsabilidades de todos los miembros participantes, del uso de herramientas y la aplicación de técnicas adecuadas de estimación de recursos, de la programación de actividades, la distribución de recursos en el tiempo y la comparación entre la ejecución real y la programada.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 6



Determinación de los Requisitos de Información

La determinación de los requisitos de información tiene como objeto desarrollar una representación del sistema que pueda ser revisada y aprobada por el usuario y es, además, una actividad que sirve de enlace entre la asignación de funciones que se ha hecho en el análisis del sistema y la etapa de diseño del software.

Desde la perspectiva del analista de sistemas, el análisis de requisitos define con mayor precisión las funciones y rendimiento del software, las interfaces que ha de tener con otros componentes del sistema y las restricciones que debe cumplir.

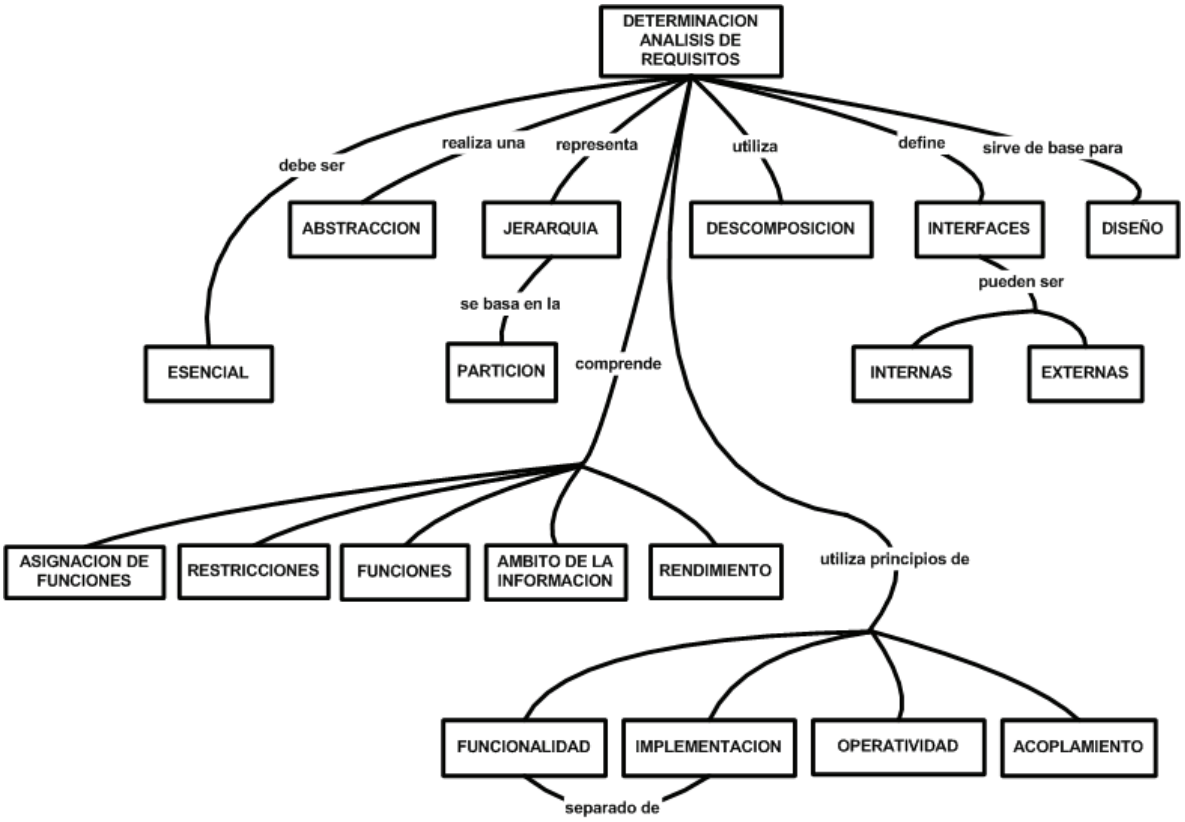


Figura 6.1: mapa conceptual de “Determinación de los Requisitos de Información”

Desde el punto de vista del diseñador, el análisis de requisitos proporciona una representación de la información y de las funciones del sistema que él se encargará de traducir en un diseño de estructura de datos y programas.

El análisis de requisitos, incluido en la especificación del proyecto, permite a todas las personas involucradas en el mismo, incluido el usuario, valorar la calidad del software una vez que haya sido construido.

La labor del analista en esta actividad debe centrarse en el qué, no en el cómo. Esto es, ¿qué datos debe manejar el sistema software?, ¿qué función debe realizar?, ¿qué interfaces debe tener?, y ¿qué restricciones tiene que cumplir?

El análisis de requisitos, por lo tanto, puede entenderse como una tarea de análisis y síntesis, se debe analizar el problema y los objetivos del cliente y, a partir de esto, sintetizar posibles soluciones.

6.1. LA DETERMINACIÓN INICIAL DE REQUISITOS

Uno de los principales inconvenientes que surgen en la fase de determinación inicial de requisitos es cómo organizar toda la información que obtiene el analista en sus entrevistas con las personas involucradas en el proyecto y cómo, además, poner de acuerdo a todas ellas sobre cuál es la solución más adecuada.

Mientras que los métodos clásicos se basan en entrevistas bilaterales, entre el analista y cada una de las partes, con lo que dejan para el investigador toda la labor de organización y obtención de un consenso, últimamente se tiende a prácticas más relacionadas con el "brainstorming", en el que cada un de las partes expone sus ideas y propuestas y se produce, luego, un debate con el objetivo de que las posiciones de cada uno de los actores vayan acercándose sucesivamente hasta llegar, finalmente, a un consenso.

En este método de trabajo, inicialmente se llevan a cabo unas reuniones preliminares con el usuario con vistas a aclarar el ámbito del problema y los objetivos generales de una solución del mismo. Como resultado de estas reuniones, el analista y el usuario redactan una descripción breve del problema, los objetivos del proyecto y el esquema general de la solución.

A continuación se convoca una reunión en la que deben estar representados el analista, el usuario y el equipo de desarrollo. Los invitados a la reunión deben conocer previamente la descripción del problema y su solución redactada anteriormente y se les pide que elaboren una lista preliminar de los objetos o entidades que pueden identificar en el sistema o su entorno, las operaciones que se realizan o sirven para interrelacionar estos objetos, las restricciones que debe cumplir el sistema y los rendimientos que se esperan del mismo. Para conducir la reunión se nombra a un moderador neutral.

La reunión comienza discutiendo la necesidad y justificación del proyecto, es decir, se debate el documento preliminar. Una vez que todos estén de acuerdo en la necesidad de desarrollar el proyecto, se presenta cada una de las listas de objetos, operaciones, restricciones y rendimientos realizadas individualmente y, a partir de éstas, se crean listas combinadas, en las que se agrupa lo redundante pero no se elimina nada.

Puede entonces comenzar la discusión sobre la lista combinada, en la que se pueden añadir nuevos elementos, eliminar otros o rescribirla de modo tal que refleje correctamente el sistema a desarrollar. Se realiza, además, una mini especificación de cada uno de los elementos de las listas, donde se define brevemente cada uno de dichos elementos. El desarrollo de estas mini especificaciones puede descubrir nuevos elementos o demostrar que alguno de ellos es-

tá incluido en otro. Se pueden dejar en el aire ciertos aspectos siempre y cuando se registre que deben ser tratados posteriormente.

Una vez concluida la reunión, el analista elaborará un borrador de la especificación del proyecto, combinando las mini especificaciones de cada elemento, que servirá de base para todo su trabajo posterior.

Este enfoque en equipo de la determinación inicial de requisitos proporciona numerosas ventajas, entre las que se pueden citar la comunicación multilateral de todos los involucrados en el proyecto, el refinamiento instantáneo y en equipo de los requisitos a partir de las ideas previas individuales y la obtención de un documento que sirva de base para el proceso de análisis.

6.2. LA ESPECIFICACIÓN PRELIMINAR

La especificación preliminar debe indicar qué es lo que debe hacer el sistema, pero no cómo hay que hacerlo. Debe ser una descripción de las necesidades y no una propuesta de solución. El cliente debe indicar qué características son imprescindibles y cuáles opcionales, y debe evitar describir la estructura interna del sistema, para no reducir la flexibilidad en la implementación. Las decisiones de diseño, tales como la utilización de un determinado algoritmo, no son tareas de análisis y, por lo tanto, no son propiamente requisitos del sistema.

La especificación preliminar no es un documento inmutable. Normalmente será ambiguo, incompleto, incorrecto e inconsistente. Incluso, aunque estén expresados de forma precisa, algunos de los requisitos reflejados pueden causar efectos secundarios no deseados, por ejemplo una gran cantidad de datos almacenados, tiempos de carga o de ejecución muy grandes, o que impliquen costos de implementación demasiado grandes, frente a unos beneficios pequeños o innecesarios. Es necesario tomar la decisión de cumplir o no estos requisitos.

6.3. LOS PRINCIPIOS DEL ANÁLISIS DE REQUISITOS

Se han ido desarrollando diversos métodos de análisis, y diversas herramientas para facilitar el uso de estos métodos en los sistemas de información. Cada uno de ellos tiene sus particularidades y un campo de aplicación distinto, pero todos utilizan una notación gráfica o textual y se basan en un conjunto de principios fundamentales.

1. Identificar y representar el ámbito de información del problema.
2. Modelar la información, la función y el comportamiento del sistema.
3. Descomponer el problema de forma que se reduzca la complejidad.
4. Avanzar desde lo más general a lo más detallado

Los dos primeros puntos se refieren, por lo tanto, a conseguir representar el sistema y la información que maneja mediante un diagrama o una representa-

ción textual que permita comprender fácilmente qué información se maneja y qué funciones se realizan con esta información.

Los dos últimos se refieren a un método de aproximación descendente que permita la división del problema en sub problemas, avanzando de lo más general a lo más específico, de forma tal que la síntesis de la solución siga una estructura jerárquica.

6.3.1. EL ÁMBITO DE INFORMACIÓN

La tarea que realiza el software consiste siempre en el procesamiento de los datos, cualquier aplicación responde a una estructura de entrada-procesamiento-salida, donde el software se encarga de procesar datos de entrada para producir, mediante una transformación, datos de salida. Por otro lado, estos datos pueden ser lógicos, numéricos, cadenas de caracteres, imágenes, etc.

Pero en un sistema de software, la información no está representada sólo por datos sino también por sucesos o eventos. Los eventos son normalmente datos de tipo lógico, por ejemplo, la activación o desactivación de la señal y su procesamiento está ligado con el control del sistema.

En un sentido general, se puede decir que la información que maneja un sistema de software se divide en datos y eventos. Los datos se refieren a aquellos que, utilizados por el tomador de decisiones, se transforman en información y que son procesados por el sistema y, por otro lado, los eventos establecen cuándo deben procesarse esos datos.

El ámbito de información del sistema admite, por tanto, dos puntos de vista, por un lado, el flujo de datos que establecen cómo se mueven los datos por el sistema y cómo se van transformando estos datos mediante los procesos, y por otro, el flujo de control, es decir, cómo se controla el sistema y cuándo hay que realizar cada procesamiento.

6.3.2. EL MODELADO DEL SISTEMA SOFTWARE

Los modelos se utilizan en el campo de la ingeniería de software para poder entender mejor lo que se quiere construir. Se utilizan modelos para los datos que transforman el sistema, esto son los modelos de datos, para las funciones que transforman esos datos, es decir, modelos de procesos y también para definir el comportamiento del sistema, los denominados modelos de control. Todos los métodos de análisis utilizados son, en realidad, métodos de modelado de sistemas.

6.3.2.1. LA UTILIDAD DE LOS MODELOS

Los modelos que se realizan en la fase de análisis sirven para ayudar al analista a entender los datos, la información, la función y el comportamiento del sistema, con lo que el análisis puede hacerse de forma más fácil y sistemática.

Pueden ser utilizados de base para el trabajo del diseñador. La arquitectura de las aplicaciones y los datos debe corresponderse con los modelos del análisis. Y, además, permiten realizar la validación del producto una vez desarrollado.

Por una parte, el sistema final debe comportarse como indica el modelo. Por otra, el sistema final permite comprobar la consistencia y la eficacia de la especificación.

Para realizar un modelo se puede utilizar una notación gráfica con lo cual éste estará representado mediante una serie de diagramas o bien una notación textual con lo cual se tendrá, por consiguiente, modelos en lenguaje natural o en un lenguaje de especificación más formal.

6.3.3. LA DESCOMPOSICIÓN DEL SISTEMA

El proceso de descomposición de sistemas se utiliza para abordar problemas que son demasiado grandes o demasiado complejos para resolverlos directamente. Mediante la descomposición, se lo divide en subproblemas más sencillos (figura 6.2), que realizan una función más clara y que pueden entenderse más fácilmente. Los criterios de cohesión y acoplamiento asisten al analista en este proceso. La descomposición se puede aplicar a los ámbitos de la información, de la función o del comportamiento.

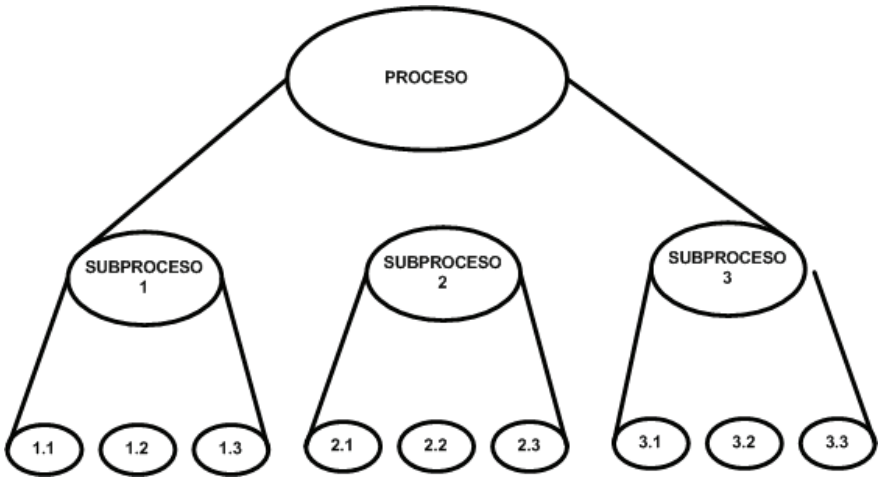


Figura 6.2: El proceso de la descomposición

6.3.4. EL ANÁLISIS DE LO GENERAL A LO ESPECÍFICO

Descomponiendo un problema en subproblemas, y definiendo modelos de cada uno de éstos, se establece una jerarquía de representaciones, que irán desde lo más general, ubicado en el ápice de la jerarquía, a las más específicas, las de nivel inferior.

Por lo tanto, cada modelo de la jerarquía produce, por descomposición de las funciones que realiza, una serie de modelos, cada uno de éstos realiza un conjunto de funciones que tendrán un mayor nivel de detalle.

6.4. LOS PRINCIPIOS DE ESPECIFICACIÓN

La especificación, independientemente del modo en que se realice, puede ser vista como un proceso de representación. Los requisitos se representan de forma tal que conduzcan finalmente a una correcta implementación del sistema. Pressman detalla en su libro una serie de principios que deben seguirse para poder obtener una buena especificación.

6.4.1. LA SEPARACIÓN DE LA FUNCIONALIDAD DE LA IMPLEMENTACIÓN

Una especificación es, por definición, una descripción de lo que se quiere realizar, no de cómo se va a realizar o implementar. Estas especificaciones se centran exclusivamente en el qué y no en el cómo. Este principio se basa en la necesidad de, por un lado, no tomar prematuras decisiones de implementación y, por otro lado, permitirle al diseñador tener la libertad de elegir, entre muchas, la mejor alternativa de elección.

6.4.2. EL LENGUAJE DE ESPECIFICACIÓN DEBE ESTAR ORIENTADO AL PROCESO

Debe emplearse una descripción orientada al proceso, en la cual la especificación del qué se obtenga mediante la especificación de un modelo de comportamiento que describa, en términos de respuestas funcionales, como responde el proceso a los distintos estímulos del entorno.

6.4.3. LA ESPECIFICACIÓN DEBE ABARCAR TODO EL SISTEMA DEL QUE EL SOFTWARE ES PARTE

Un sistema está compuesto por un conjunto de partes que interactúan. El comportamiento de un componente específico del mismo, como es el software, sólo puede ser definido en el contexto del sistema completo. Por lo tanto, la especificación debe describir todas los partes del sistema, estableciendo el comportamiento no sólo del componente software, sino también de las interfaces entre éste y los demás componentes del sistema.

6.4.4. LA ESPECIFICACIÓN DEBE ABARCAR TAMBIÉN EL ENTORNO DEL SISTEMA

Hay que especificar, también, el entorno en el que opera el sistema. La especificación del entorno permite describir la interfaz del sistema de la misma forma que las interfaces de los componentes del mismo. De esta forma se pueden representar sistemas dinámicos cuyo comportamiento varía dependiendo de los estímulos que reciban del entorno.

Esta especificación del entorno no se hace con vistas a implementarlo, puesto que ya viene dado y no se puede modificar, sino que sirve para definir los límites del sistema y su interfaz, permitiendo además probarlo.

6.4.5. LA ESPECIFICACIÓN DEBE MODELAR EL DOMINIO DEL PROBLEMA

La especificación del sistema debe ser una descripción del dominio del problema, en vez de ser un modelo de diseño o implementación. Debe describir el sistema tal como es percibido por los expertos en el dominio de aplicación. Los elementos del sistema

deben corresponderse con objetos reales de dicho dominio, ya sean individuos, máquinas u organizaciones, y las acciones que se realicen deben corresponderse, también, con lo que realmente ocurre en ese dominio. Además, deben poder describirse en la especificación las reglas o leyes que gobiernan

los objetos del dominio de aplicación. Para que esto sea posible, el formato de la especificación y su contenido deben ser adecuados al problema.

6.4.6. LA ESPECIFICACIÓN DEBE SER OPERATIVA

La especificación debe servir para determinar si una implementación concreta la satisface. Esto puede hacerse bien mediante la verificación de la corrección del sistema implementado, cosa que normalmente no puede demostrarse, o bien mediante una serie de casos de prueba elegidos arbitrariamente.

A partir de los resultados de una implementación sobre un conjunto arbitrario de datos de entrada, debe ser posible usarla para validar estos resultados, a pesar de que la especificación no describa el cómo sino solamente el qué.
Este principio no establece que la especificación tenga que ser ejecutable, sino más bien que pueda ser utilizada para demostrar su validez.

6.4.7. LA ESPECIFICACIÓN DEBE SER AMPLIABLE Y TOLERANTE A LA INCOMPLETITUD

Una especificación es un modelo o abstracción de un sistema real, por lo tanto nunca será completa y puede desarrollarse, de acuerdo a las necesidades, a distintos niveles de detalle. Normalmente el desarrollo de las especificaciones se hace de forma incremental, por tal motivo los elementos del sistema estarán, en cada momento, parcialmente especificados.

6.4.8. LA ESPECIFICACIÓN DEBE ESTAR LOCALIZADA Y DÉBILMENTE ACOPLADA

Durante su desarrollo, una especificación sufre continuas modificaciones. Por este motivo, la estructura de la especificación debe permitir que estos cambios se realicen lo más fácilmente posible. El principio de localización, relacionado con el concepto de cohesión, establece que si es necesario modificar un elemento de la especificación, esta modificación sólo se realice en un punto de la misma. El principio del bajo acoplamiento establece que se puedan añadir y quitar partes de la especificación fácilmente sin que esto implique la necesidad de modificar otras especificaciones del sistema.

Par cumplir estos dos principios la especificación, ésta ha de ser modular y no redundante, con interfaces pequeñas y bien definidas.

6.5. RESUMEN

La determinación de los requisitos de información es la sistematización del proceso de definición de los requerimientos en el proceso de desarrollo de software. Su tarea es la de entender correctamente el problema antes de avanzar hacia la etapa de diseño.

El objetivo principal es generar especificaciones que describan el comportamiento del sistema en forma no ambigua, consistente y completa, transformando las necesidades del usuario, que generalmente son incompletas, expresándolas en términos informales.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 7

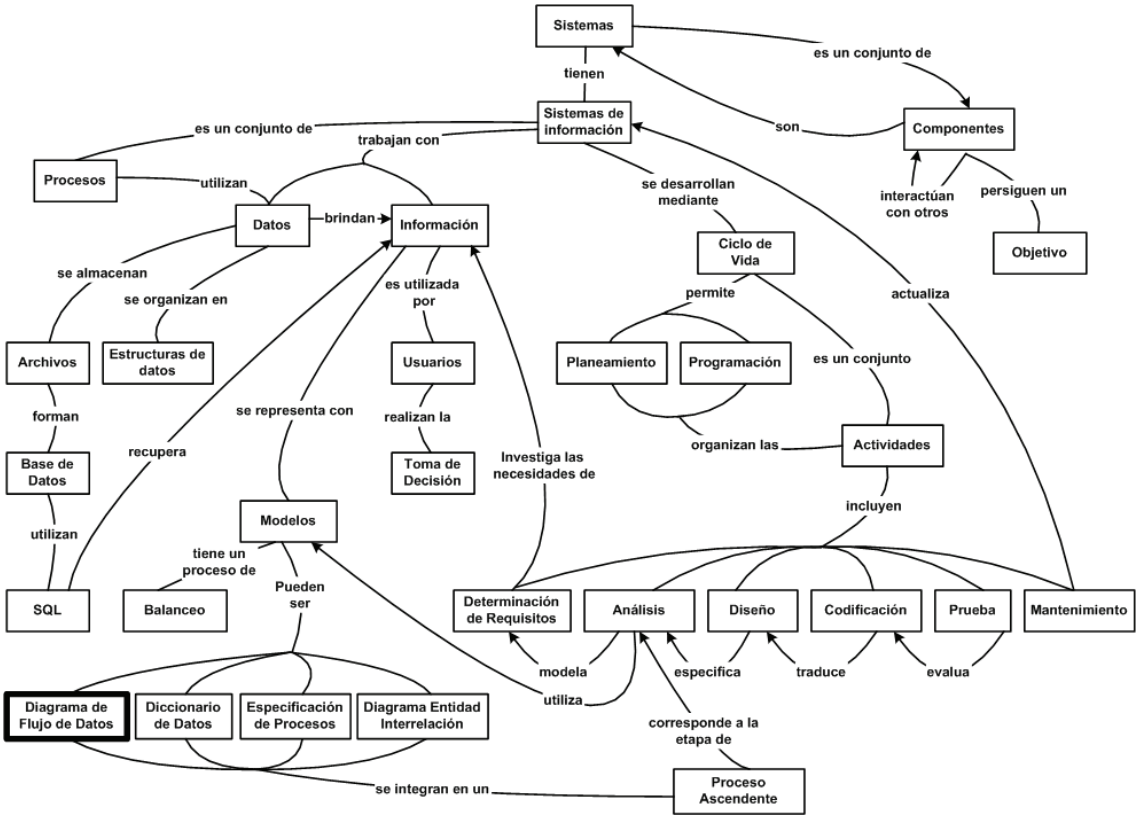


Diagrama de Flujo de Datos

Todos los métodos de análisis de los requisitos de información se basan en la construcción de un conjunto de modelos del sistema que se pretende desarrollar. Utilizando alguna notación, propia de cada método, se crean modelos que reflejen el sistema y aplicando las técnicas de descomposición mediante un proceso de construcción descendente o ascendente, se establece la esencia del sistema a desarrollar.

La construcción de modelos presenta ventajas claras, tales como centrarse en determinadas características del sistema, dejando de lado otras menos signifi-

cativas. Esto, además, permite enfocar las discusiones con el usuario en los aspectos más importantes del sistema, sin distracciones en características que sean, por el momento, irrelevantes.

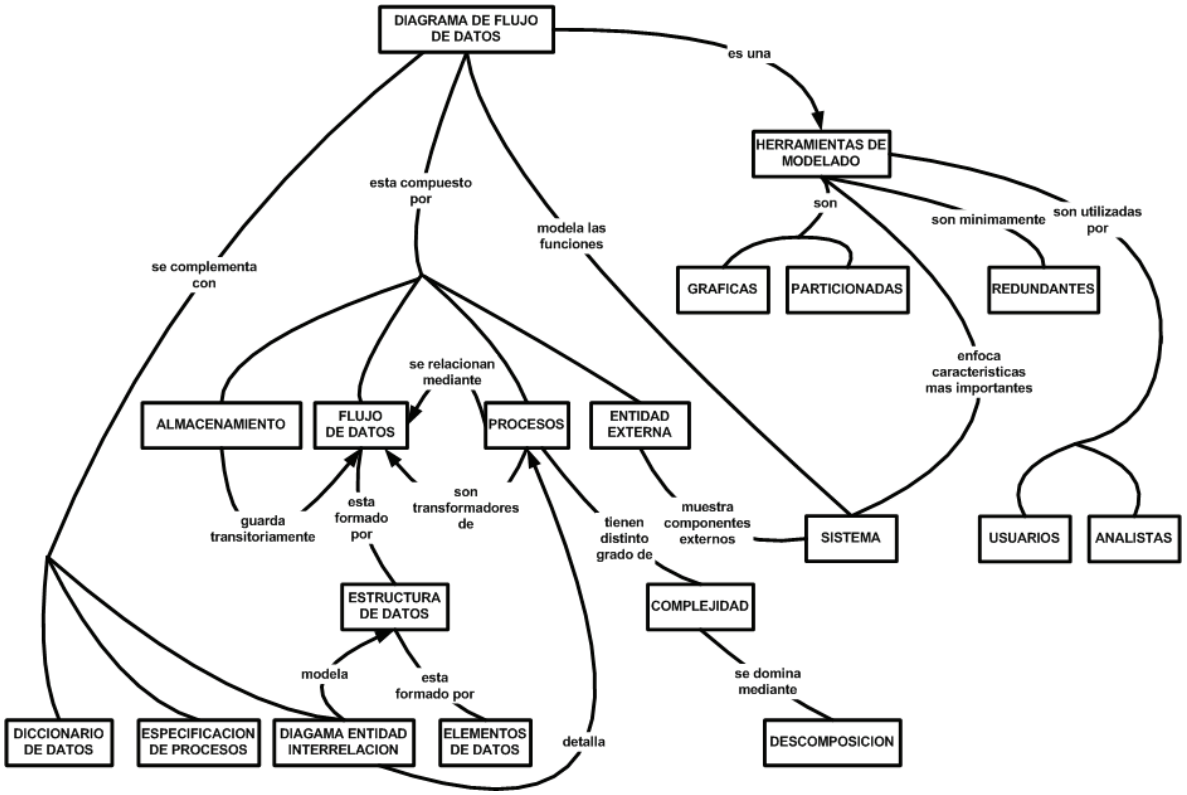


Figura 7.1: Mapa conceptual de “Diagrama de Flujo de Datos”

Los modelos, además, permiten realizar cambios y correcciones en los requisitos a un bajo costo y sin correr ningún riesgo. Si se perciben desentendimientos con el usuario o si éste ha cambiado de idea acerca de los requisitos del sistema, se puede cambiar el modelo o incluso desecharlo y empezar de nuevo. Si no se hiciesen modelos, los cambios en los requisitos sólo se efectuarían después de construir el software, y el costo sería mucho mayor.

Permite, también, verificar que el analista haya entendido correctamente las necesidades del usuario y que las documente de forma tal que los diseñadores y programadores pueden construir el software a partir de él.

7.1. EL ENFOQUE CLÁSICO

No todas las técnicas de análisis logran los objetivos antes descriptos. Una descripción del sistema de cientos de páginas constituye un modelo, pero ocul-

ta las características del sistema, tanto las relevantes como las irrelevantes, su desarrollo puede ser muy costoso, es difícil de modificar y no permite verificar si se han establecido correctamente los requerimientos del usuario.

El análisis de requisitos de información clásico, utilizado en los albores de esta disciplina, consistía en redactar especificaciones funcionales en forma de documentos textuales. Estos tenían una serie de características indeseables.

Por un lado, para entender el sistema había que leer la especificación completa. No había posibilidad de que ni el analista ni el usuario pudiesen centrarse en una determinada parte de la especificación sin tener que leer el resto.

Las especificaciones, con frecuencia, eran redundantes. La misma información se podía encontrar en diferentes partes del documento. Debido a esto, cualquier cambio que se hiciese en la especificación debía reflejarse en varios puntos del documento. Esta situación producía, con frecuencia, inconsistencias en la medida que, si no se realizaban cambios en todas las partes del documento, se tendrían distintas definiciones para los mismos objetos del sistema.

Otra característica indeseable era la ambigüedad, al estar escritas en lenguaje natural, podían ser interpretadas de forma distinta por analistas, usuarios, diseñadores o programadores. Estudios realizados muestran que la mayoría de los errores encontrados en el sistema final con un alto costo asociado a la corrección de esos yerros surgía de este tipo de malentendidos.

Debido a esto, la mayor parte del software de esa época carece de una documentación fiable, incluso aunque se hubiese realizado análisis y redactado una especificación.

Por estos motivos fueron surgiendo nuevos métodos de análisis, cuyo objetivo era obtener especificaciones gráficas, formadas por una colección de diagramas, acompañados de información textual detallada, que sirviese de material de referencia; particionadas, de forma que fuese posible leerse o trabajar sobre partes individuales de la especificación sin tener que leerla completamente; mínimamente redundantes, de forma que los cambios en los requisitos se realicen en un sólo punto de la especificación y, además, transparentes, tal que fuesen fáciles de leer y de entender. Los sistemas son complejos, las especificaciones tienen que ser claras y sencillas.

Bajo el nombre genérico de análisis estructurado, se engloban una serie de propuestas de diversos autores, con las características antes nombradas. Entre otros se destacan De Marco, Yourdon, Gane y Sarson. Cada uno de ellos ha desarrollado su propio método de análisis, mejorando, ampliando o adaptando los anteriores a algún campo de aplicación específico.

Existen diferentes técnicas y notaciones de modelado de sistemas que puede utilizar un ingeniero del software. Combinando todas ellas se puede establecer un modelo completo del sistema, pero lo importante es usar aquellos diagra-

mas que sirvan en una situación determinada. Habrá escenarios en que lo importante sea modelar el flujo de datos, mientras que el control sea obvio. En otras situaciones los datos serán lo suficientemente complejos como para justificar un modelo de datos. En otras, por ejemplo, en sistemas de tiempo real, será necesario especificar el comportamiento del sistema en función del tiempo.

En general, la mayoría de los sistemas requerirán utilizar varios modelos para representarlos. Cada uno de ellos se centra en un conjunto limitado de aspectos del sistema, dejando de lado otros. Combinando los diversos modelos se puede tener una visión detallada de todas las características del sistema. Esto es importante ya que los sistemas utilizan estructuras de datos complejas, realizan funciones complejas y tienen pautas de comportamiento complejas.

7.2. EL DIAGRAMA DE FLUJO DE DATOS

Un sistema de procesamiento de datos incluye tanto datos como procesos, y cualquier análisis de un sistema debe incluir ambos aspectos. Se necesita una técnica para modelar sistemas que describa qué funciones son las que realiza, qué interacción se produce entre esas funciones, qué transformaciones de datos realiza el sistema y qué datos de entrada se transforman en qué datos de salida.

A medida que la información se mueve a través del sistema, va siendo modificada mediante una serie de transformaciones. El diagrama de flujo de datos es una técnica gráfica que representa el flujo de datos y las transformaciones que se aplican a ellos.

Representa, esencialmente, qué funciones o qué transformaciones se realizan sobre los datos pero no cuándo se realizan o en qué secuencia.

7.3. LOS COMPONENTES DEL DIAGRAMA DE FLUJO DE DATOS

Para representar el sistema mediante diagrama de flujo de datos se utilizará la notación propuesta por Yourdon, posiblemente la más extendida. Se detallará a continuación los elementos que se utilicen en el diagrama de flujo de datos.

7.3.1. LOS PROCESOS

Los procesos o funciones son, básicamente, transformadores de flujos que muestran una parte del sistema que transforma datos de entrada en datos de salida. Se representa gráficamente con un círculo (figura 7.1).

Figura 7.1: Representación de un proceso

Son los componentes que realizan cada una de las funciones del sistema. Los procesos en el diagrama de flujo de datos evitan mostrar detalles procedimentales, estos se aclaran mediante las especificaciones de procesos.

7.3.2. LAS ENTIDADES EXTERNAS

Las entidades externas (figura 7.2), representan a los elementos externos al sistema. Estos pueden ser personas, organizaciones u otros sistemas que interactúan con él y que están fuera de los límites del sistema. Las entidades externas proporcionan datos que serán transformados por el sistema o son las que consumen los datos que fueron transformados por el sistema.

Figura 7.2: Representación de una entidad externa

Los flujos de datos que comuniquen el sistema con las entidades externas representan las interfaces del mismo. Las entidades externas aparecen, principalmente, en el diagrama de contexto y pueden repetirse en los niveles inferiores.

La forma de reconocerlas es preguntando qué harán esas entidades con los datos que les ofrece el sistema o de donde obtiene los datos que la entidad externa entrega al sistema. Si la respuesta a ambas preguntas es “no interesa”, se está en presencia de una entidad externa.

Es evidente que ni el analista ni el diseñador están en posibilidades de cambiar los contenidos de una entidad externa o la forma en que trabaja. Sin duda existirán muchas entidades externas, pero solamente se deben incluir, como objetos de estudio, aquellas que se conectarán con el sistema.

7.3.3. LOS FLUJOS DE DATOS

Los flujos de datos representan datos o estructuras de datos que fluyen a través del sistema (figura 7.3). La flecha indica el sentido de flujo. Los flujos de datos conectan procesos entre sí, con entidades externas o con almacenamientos de datos.

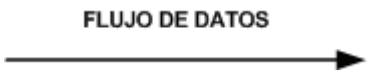


Figura 7.3: Representación de un flujo de datos

Los flujos de datos representan datos en movimiento. Puede tratarse de un elemento de dato simple o compuesto o incluso de una colección de datos de estructura compleja y van etiquetados con un nombre que identifica a los datos que transportan y, posiblemente, con el estado de dicha información. Por ejemplo, número de teléfono, número de teléfono correcto, número de teléfono erróneo.

7.3.4. LOS ALMACENAMIENTOS DE DATOS

Mientras que los flujos de datos representan datos en movimiento los almacenamientos representan datos estáticos o en reposo.

Los almacenamientos describen a los datos persistentes que serán utilizados por el sistema (figura 7.4). Permiten guardar, temporalmente, datos que luego podrán ser procesados por el mismo proceso que los creó o por otro distinto. En la mayoría de los casos, se utilizarán almacenamientos de datos cuando dos procesos intercambien datos pero sus funciones no se ejecuten en forma simultánea.

Figura 7.4: Representación del un almacenamiento

Los almacenamientos se comunican solamente con los procesos mediante flujos de datos, de modo tal que éstos pueden entrar o salir de él (figura 7.5). Si un flujo de datos ingresa a un almacenamiento se interpreta como una modificación del contenido de su estructura, esto es, una alta, baja o una actualización de uno o más valores de los elementos de datos que lo componen. Un flujo de datos que parte desde un almacenamiento a un proceso expresa la lectura de uno o más elementos, no hay modificación del contenido de su estructura de datos.

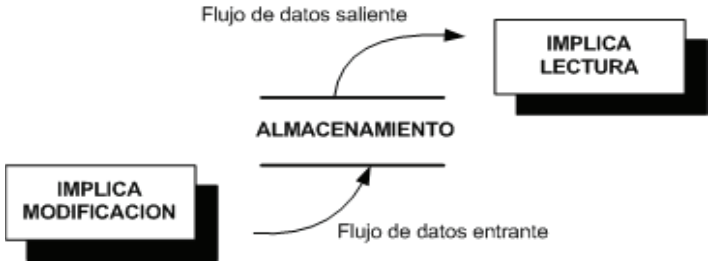


Figura 7.5: Flujos entrantes y salientes a un almacenamiento

Los almacenamientos de datos no detallan ningún tipo de implementación ulterior y van etiquetados con un nombre significativo que haga referencia a los datos que contienen, generalmente en plural.

7.4. LAS RELACIONES ENTRE COMPONENTES DEL MODELO

No todos los componentes del modelo pueden relacionarse entre sí. Los procesos pueden comunicarse con almacenamientos, otros procesos o entidades externas. Los almacenamientos y las entidades externas sólo pueden comunicarse con procesos (figura 7.6).

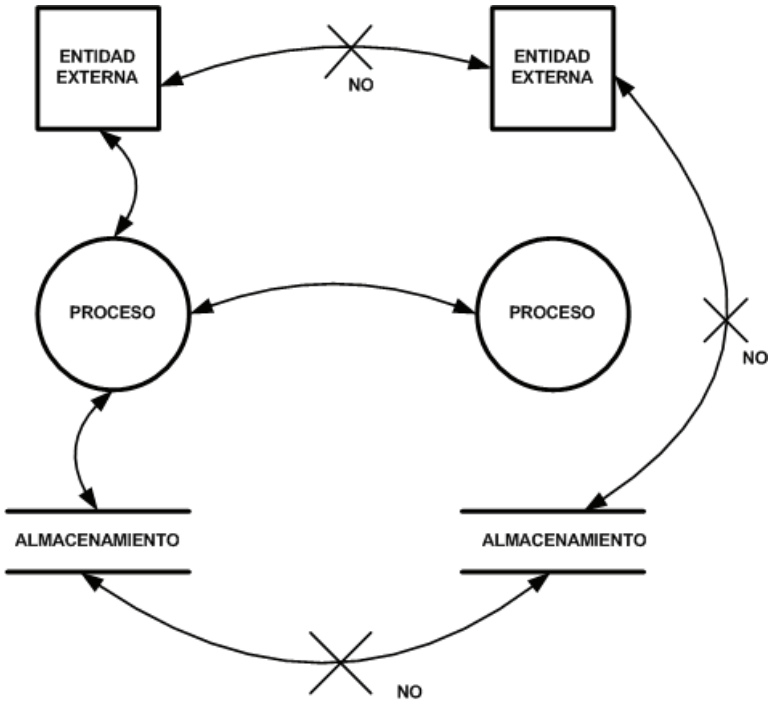


Figura 7.6: Relaciones entre componentes del modelo

7.5. EL DIAGRAMA DE CONTEXTO

Se puede utilizar el diagrama de flujo de datos para representar el sistema en cualquier nivel de abstracción. El diagrama de flujo de datos de nivel 0 se lo denomina diagrama de contexto (figura 7.7) y en él el sistema está representado por un sólo proceso, que identifica cuál es la función principal del sistema, mostrando además los flujos de datos que lo relacionan con las entidades externas.

El diagrama de contexto tiene una gran importancia ya que, aunque simple, resume el requisito principal del sistema, esto es, recibir entradas, luego las procesa de acuerdo con determinadas funciones y genera, por último, las salidas. A partir del diagrama de contexto se pueden ir construyendo nuevos diagramas, en un proceso de descomposición descendente, que vaya definiendo con mayor nivel de detalle los flujos de datos y procesos de transformación que ocurren en el sistema, de forma tal que al final se obtendrá una jerarquía de diagramas.

En general, cualquier proceso que aparezca en un diagrama de flujo de datos puede ser descripto, más detalladamente, en un nuevo diagrama. A esto se denomina explosión de un proceso. Cada proceso que se está describiendo aparece descompuesto en una serie de subprocesos o subsistemas, cada uno de los cuales se encarga de detallar un aspecto determinado del proceso original.

Los flujos de datos que entran y salen del proceso que se está describiendo deben entrar y salir del diagrama de flujo de datos que lo desarrolla. Además de estos flujos, el diagrama contendrá, por lo general, nuevos flujos que comunican los procesos que figuran en él y posiblemente también almacenamientos de datos.

Figura 7.7: El diagrama de contexto

7.6. LA CONSTRUCCIÓN DE UN DIAGRAMA DE FLUJO DE DATOS

Dado que en el diagrama de contexto el sistema estará representado por un sólo proceso, este diagrama de flujo de datos de nivel 0 sólo dará lugar, luego del proceso de explosión, a un diagrama de flujo de datos de nivel 1. A su vez, éste puede dar lugar a tantos diagrama de flujo de datos de nivel 2 como

procesos contenga, y así sucesivamente hasta que se haya alcanzado un nivel en el que los procesos sean lo suficientemente simples como para no necesitar una descripción más detallada en un diagrama de flujo de datos. De esta forma, se construye una jerarquía de diagrama de flujo de datos.

7.6.1. LOS PROCESOS POR NIVEL

Un diagrama de flujo de datos debe contener, por simplicidad, no más de 9 procesos. Es razonable tener 7 ± 2 procesos por nivel. Una de las características más importantes de las herramientas de modelado en la facilidad de su lectura. Un diagrama de flujo de datos con muchos procesos sería complicado para su lectura y comprensión,

7.6.2. LOS NOMBRES DE LOS PROCESOS

Cada proceso de un diagrama de flujo de datos debe tener un nombre corto y significativo. Normalmente un verbo más un sustantivo. Si bien la descripción de los componentes es una de las tareas encomendada al diccionario de datos, es conveniente que los nombres que se propongan a los procesos del diagrama de flujo de datos sean significativos en el contexto y cortos en su extensión.

7.6.3. LA NUMERACIÓN DE LOS PROCESOS

Los procesos deben ser numerados. Todo proceso, en cualquier nivel, debe tener un número que lo identifique. La numeración de los procesos en el diagrama de flujo de datos no implica secuencialidad, solamente es una referencia para identificarlo a él y a sus procesos descendientes.

7.6.4. LA EXPLOSIÓN DE LOS PROCESOS

Para modelar sistemas complejos se utiliza la descomposición, ésta da como resultado un diagrama de flujo de datos a distintos niveles de detalle. La descomposición o explosión en subprocesos es uno de los mecanismos que se utilizan para administrar la complejidad. El limite de la descomposición de los procesos está dado por la complejidad del mismo.

Los diagrama de flujo de datos de niveles inferiores desarrollan, de forma más concreta, los procesos de niveles superiores. La explosión de procesos implica la especificación más detallada de los procesos más complejos y se realiza hasta alcanzar un nivel de especificación mínimo y sencillo.

7.6.5. EL BALANCEO EN LAS EXPLOSIONES

En cada diagrama de flujo de datos proveniente de la explosión de un proceso de orden superior debe representarse los mismos flujos de datos que en el proceso padre. La explosión implica el detalle de un proceso complejo dividido en subprocesos, por tal motivo los flujos de entrada y de salida en el proceso superior debe corresponderse con los de los niveles inferiores.

7.6.6. LA NUMERACIÓN EN LA EXPLOSIÓN DE PROCESOS

Normalmente, el sistema que se está modelando será lo suficientemente complejo como para necesitar varios diagramas de flujo de datos. Se precisará construir, entonces, una jerarquía de diagrama de flujo de datos.

Cada diagrama de flujo de datos de un nivel “n” será el resultado de la explosión de un proceso de un diagrama de flujo de datos de nivel “n-1” (figuras 7.8 y 7.9). Es necesario que el nombre del diagrama de flujo de datos sea el nombre del proceso que desarrolla, que la numeración de los procesos en el diagrama de flujo de datos “n” derive del número del diagrama de flujo de datos “n-1”, por ejemplo, 3, 3.1, 3.2, etc., además debe mantenerse la consistencia entre los flujos de datos en ambos niveles.

La regla de equilibrio dice que los flujos de datos que entran o salen del proceso de orden superior deben aparecer en el diagrama de flujo de datos de orden inferior, manteniendo, además, el nombre y el sentido.

En los diagramas de flujo de datos de niveles superiores generalmente es necesario, cuando los flujos de datos pasan al nivel inferior, descomponerlos. Para realizar esta descomposición se puede hacer divergir el flujo en dos o más o, al menos, dejar reflejada esta descomposición en el diccionario de datos.

El diagrama de flujo de datos debe ser leído y comprendido, no sólo por el analista que construyó el modelo, sino también por los usuarios que son los expertos en la materia de aplicación.

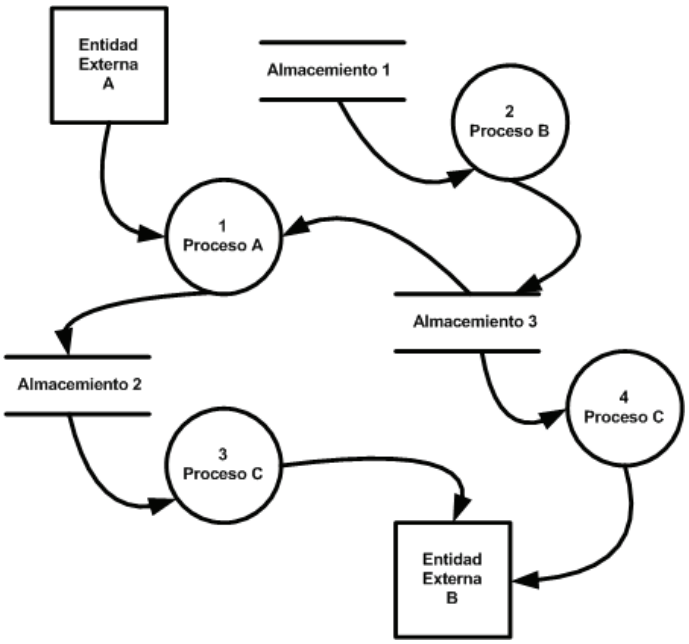


Figura 7.8: Diagrama de flujo de datos de primer nivel

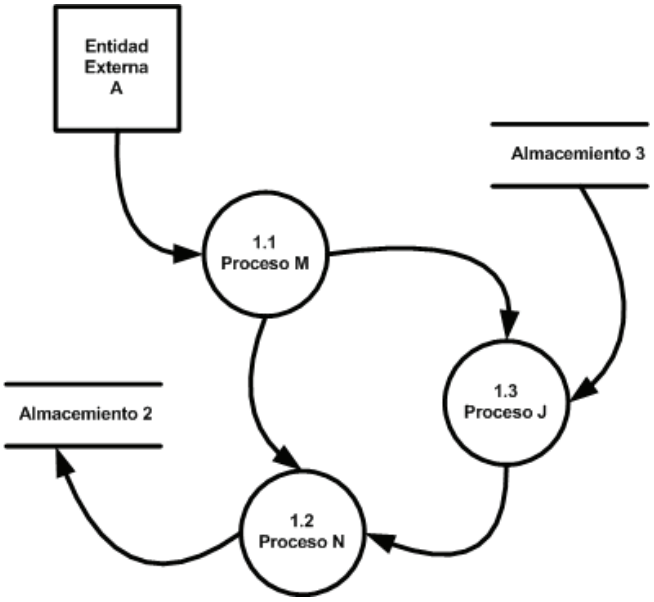


Figura 7.9: Diagrama de flujo de datos de segundo nivel

7.7. LAS ESPECIFICACIONES DE LOS PROCESOS PRIMITIVOS

Los diagramas de flujo de datos permiten identificar las principales funciones o transformaciones que realiza un sistema y las relaciones entre éstas, pero no indican nada acerca de los detalles de cómo se realizan estos procesos.

Para definir los detalles de qué datos de entrada se transforman en qué datos de salida y cómo se realiza esta transformación se necesita una descripción detallada de los procesos. Para esto se utilizará la especificación de proceso.

En los diagramas de flujo de datos de menor nivel, esto es, los más altos en la jerarquía, los procesos se describen mediante un nuevo diagrama de flujo de datos que define, más detalladamente, las funciones que realiza y los flujos que maneja. Este proceso de descomposición debe continuar hasta que se alcance un nivel en el que un proceso pueda ser descripto de forma sencilla y no ambigua. Estos procesos se denominan primitivos.

Una forma de especificar un proceso primitivo es mediante un algoritmo. Esto no quiere decir que el algoritmo propuesto en el análisis se corresponda con la implementación final del proceso, sino que es simplemente una forma de describir la operación. La elección del algoritmo definitivo se hace en la fase de diseño, considerando además de la especificación, criterios de eficiencia y las características del lenguaje de implementación.

7.8. RESUMEN

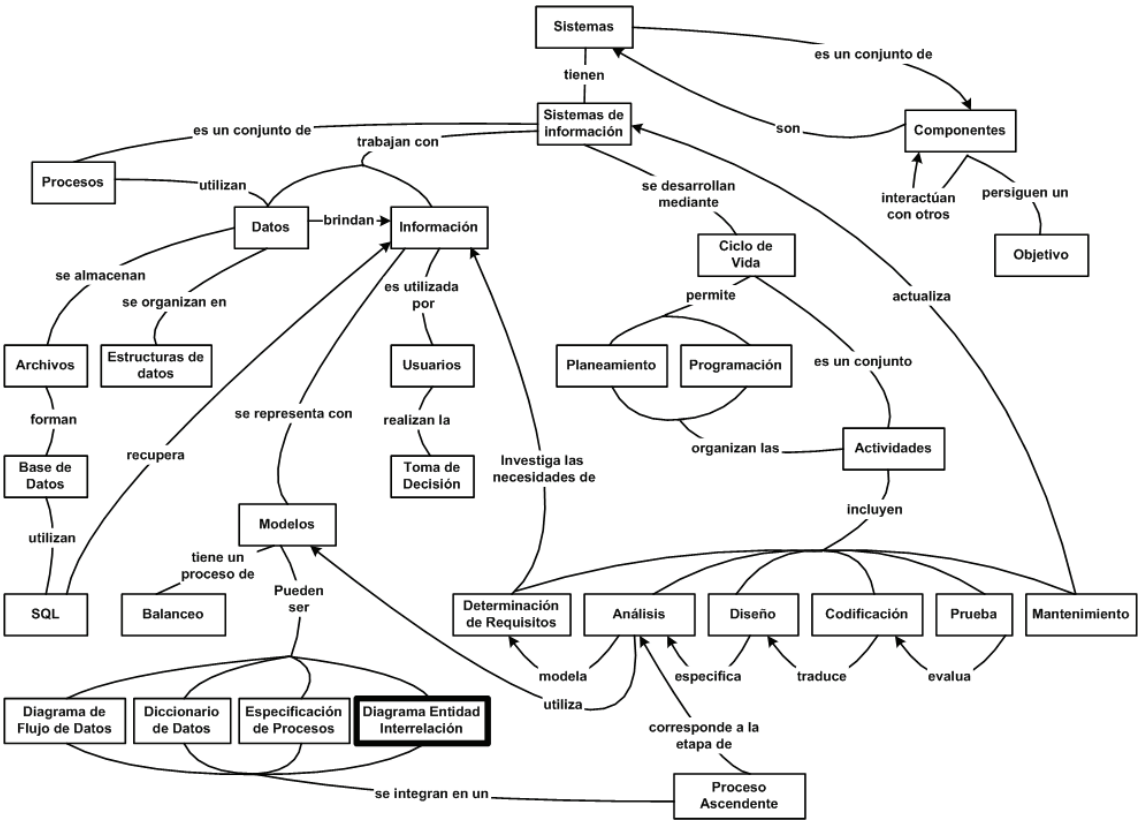
El diagrama de flujo de datos es una herramienta de modelado que describe el comportamiento de un sistema independientemente de su implementación. Está compuesto por procesos, que son transformadores de flujos de datos; almacenamientos, que constituyen depósitos temporales donde se guardan los datos para su posterior utilización; flujos de datos, que representan conductos por donde pasan los datos del sistema y entidades externas que son objetos externos al sistema con los cuales éste se comunica.

El diagrama de flujo de datos es un conjunto de procesos interrelacionados mediante flujo de datos que se utiliza en sistemas de información donde la descripción de los procesos es importante para la etapa de modelado.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 8



El Modelo Entidad Interrelación

El modelo entidad interrelación es una vista conceptual de alto nivel independiente de la futura implementación lógica y física. Está históricamente vinculado al modelo relacional pero puede ser utilizado, también, para representar los datos del sistema y luego ser transformado a cualquier otro modelo lógico.

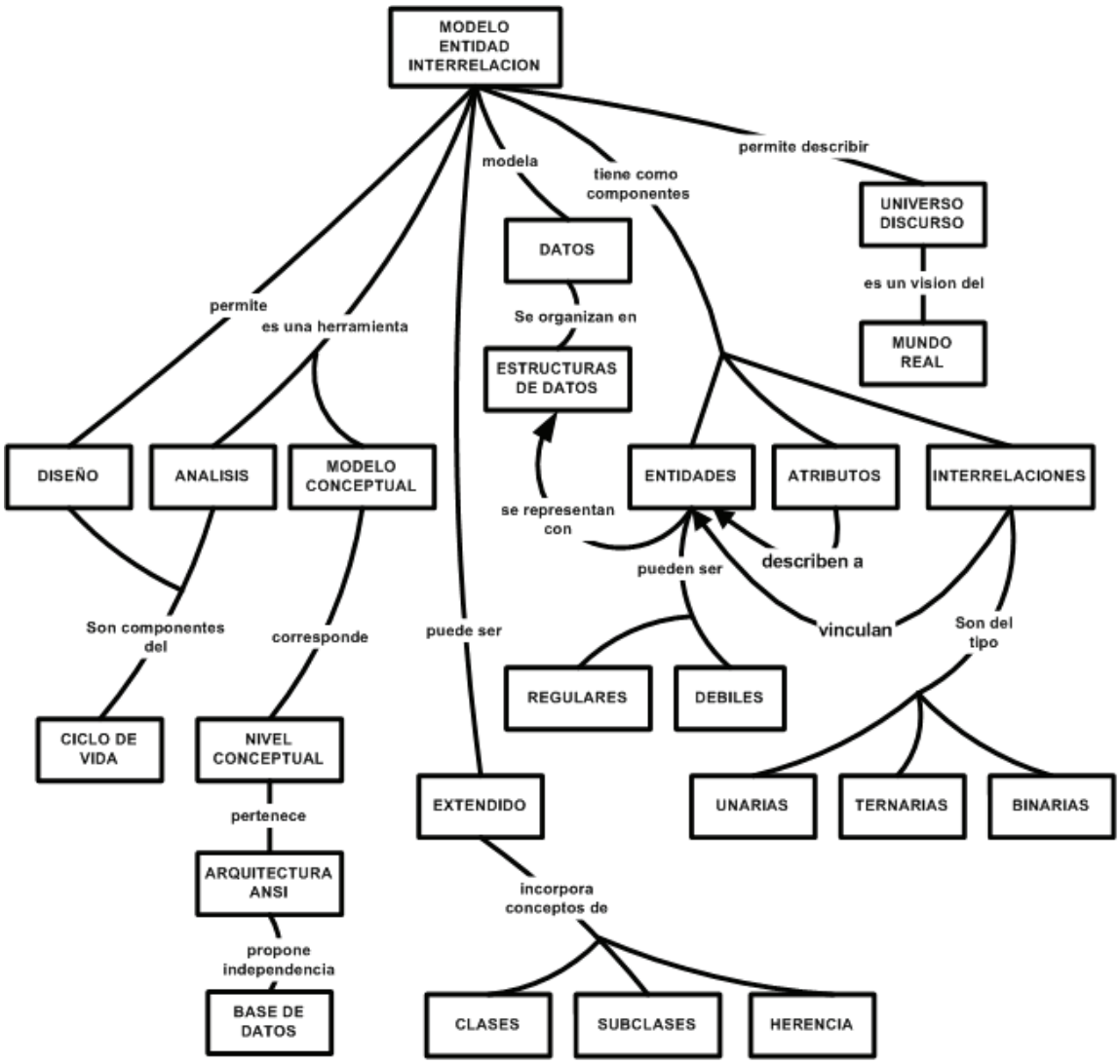


Figura 8.1: Mapa conceptual de “Modelo Entidad Interrelación”

Si sólo interesase modelar las funciones del sistema y la relación entre los datos fuera trivial y no significativa, el diagrama de flujo de datos, sería suficiente. No obstante, normalmente se precisa comprender cómo las estructuras de datos participantes en el sistema se relacionan entre sí. El diagrama entidad interrelación muestra la relación entre los datos del sistema.

El modelo entidad interrelación, como representación conceptual, permite una descripción mediante entidades, atributos e interrelaciones entre esas entidades.

Las características del sistema que fueron investigadas en la etapa de determinación de requerimientos, brindará la información necesaria para realizar los distintos modelos del análisis. Este modelo se construye a partir de los requerimientos del usuario. Puede ser utilizado en el análisis y, mediante su transformación al modelo lógico, normalmente relacional, se constituye, también, en una herramienta de la etapa de diseño. Puede utilizarse, además, para asegurar la consistencia de los requerimientos investigados, haciendo participe al usuario.

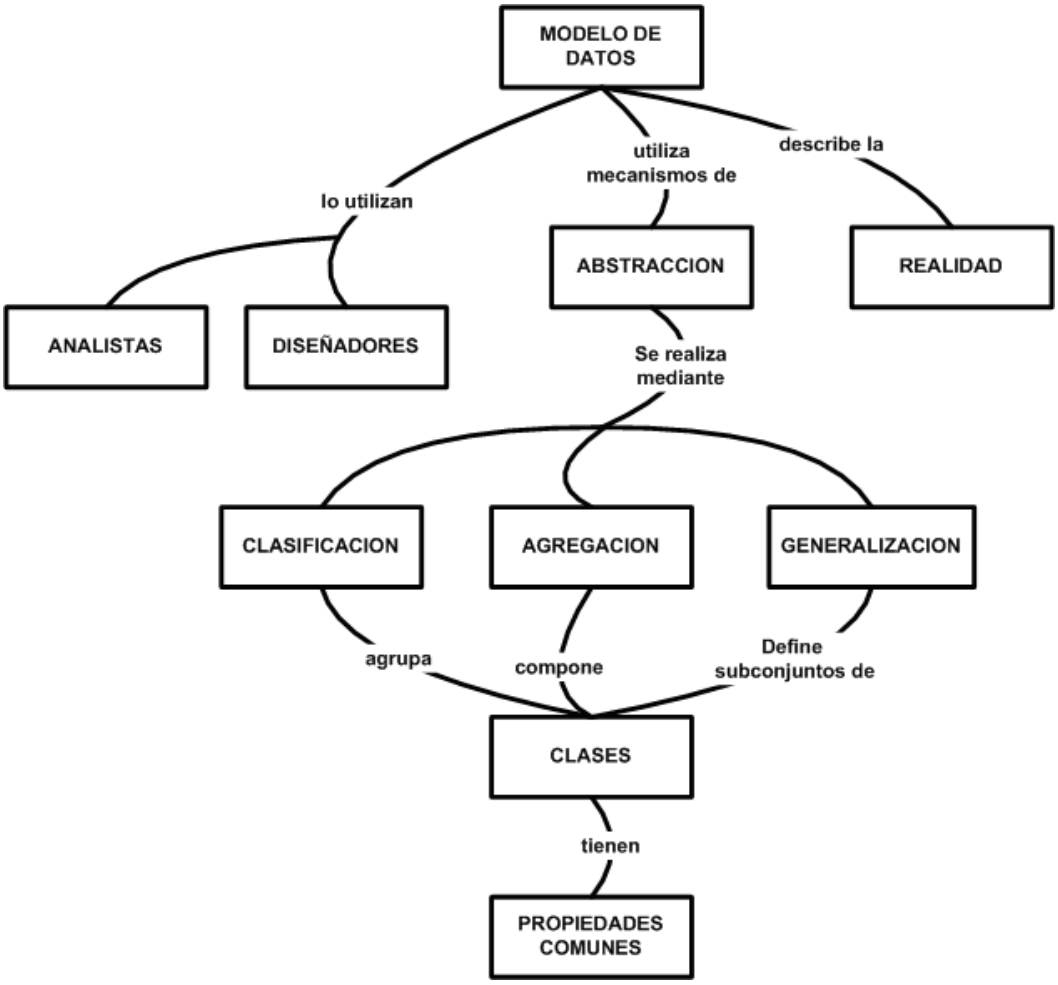


Figura 8.2: Mapa conceptual de “Modelo de Datos”

En capítulos anteriores se estableció una diferencia entre dato e información. Se definía a la información como aquellos datos que fueron procesados por el sistema e interpretados por el usuario para la toma de decisiones. En cambio, los datos son sólo símbolos que describen algo. Esta diferencia se estable porque el usuario precisa información, en cambio, el diseñador precisa tener la libertad suficiente para optar por distintas estructuras de datos que permitan su persistencia, evitando las anomalías de actualización. El modelo entidad interrelación es una herramienta que asiste en esta tarea.

8.1. LOS CONCEPTOS SOBRE MODELO DE DATOS Y ABSTRACCIÓN

Se desarrollarán algunos conceptos propuestos por Batini en su libro Diseño Conceptual de Base de Datos. Los modelos de datos permiten describir la realidad. Los analistas y diseñadores utilizan los modelos para construir esquemas, estos son representaciones incompletas y parciales de esa realidad, pero suficientes para los fines propuestos. La calidad de los esquemas resultantes depende no sólo de la habilidad de los analistas y diseñadores, sino también de las características del modelo de datos seleccionados.

La construcción de los modelos de datos se basa en una pequeña colección de mecanismos de abstracción primitivos. Estos son, la clasificación, la agregación y la generalización. Las abstracciones ayudan al diseñador a entender, clasificar y modelar la realidad.

La abstracción es un proceso mental que se aplica al seleccionar algunas características y propiedades de un conjunto de objetos y excluir otras no pertinentes (figura 8.2). Mediante las abstracciones, el analista y diseñador es capaz de clasificar los objetos del mundo real y modelar las interrelaciones de las distintas clases.

8.1.1. LA ABSTRACCIÓN DE CLASIFICACIÓN

La abstracción de clasificación (figura 8.3) se utiliza para definir un concepto como una clase de objetos de la realidad que se caracteriza por tener propiedades comunes. Se representa gráficamente como un árbol, que tiene como raíz la clase y como hojas los elementos de la clase. La clasificación es el procedimiento utilizado cuando, partiendo de elementos individuales de información, se identifican tipos de campos o atributos. Un mismo objeto real puede, por supuesto, clasificarse de varias maneras.

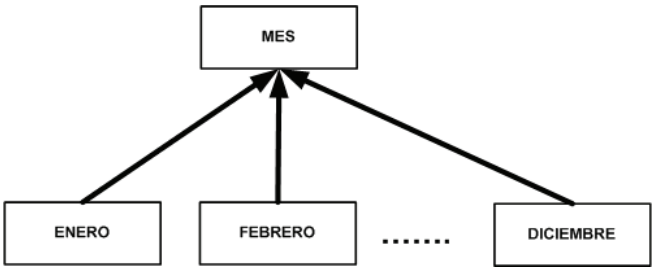


Figura 8.3: Ejemplo de clasificación

8.1.2. LA ABSTRACCIÓN DE AGREGACIÓN

La abstracción de agregación (figura 8.4) define una nueva clase a partir de un conjunto de otras que representan sus partes componentes. Este tipo de abstracción se representa por un árbol en el cual todos los nodos son clases; la raíz representa la clase creada por agregación de las clases representadas por las hojas. Cada rama del árbol indica que una clase hoja es una parte de la clase representada por la raíz.

La agregación es el procedimiento mediante el cual se reúnen tipos de campos relacionados en grupos, como por ejemplo, las estructuras de datos.

La clasificación y la agregación son las dos abstracciones básicas utilizadas para construir estructuras de datos en las base datos y en la mayoría de los lenguajes de programación.

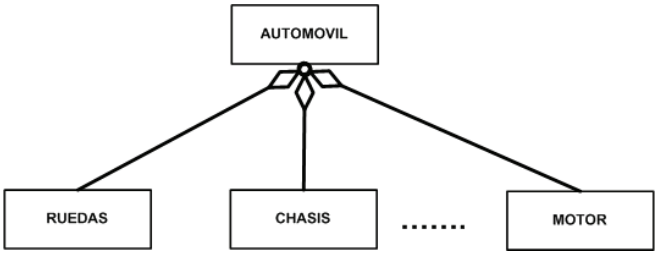


Figura 8.4: Ejemplo de agregación

8.1.3. LA ABSTRACCIÓN DE GENERALIZACIÓN

Una abstracción de generalización (figura 8.5) define una relación de subconjunto entre los elementos de dos o más clases. Cada generalización se representa mediante un árbol, en el que todos los nodos son clases, con la clase genérica como raíz y las clases subconjunto como hojas; cada rama del árbol expresa que una clase hoja es un subconjunto de la clase raíz.

En una generalización, todas las abstracciones definidas para la clase genérica son heredadas por las clases subconjunto.

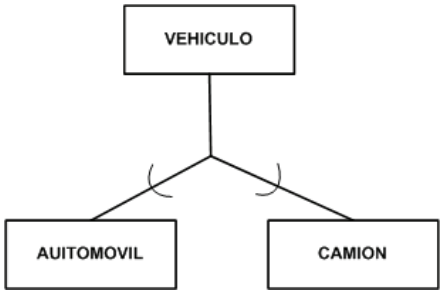


Figura 8.5: Ejemplo de generalización

Las tres abstracciones son independientes entre sí, ninguna puede describirse en función de las otras y proporcionan, cada una de ellas, un mecanismo diferente en el proceso de estructuración de los datos.

8.2. LA AGREGACIÓN BINARIA Y LA CARDINALIDAD

Una agregación binaria (figura 8.6) es una correspondencia que se establece entre dos clases. Se pueden establecer varias agregaciones binarias entre dos clases.

Dada una agregación A entre las clases C_1 y C_2. La cardinalidad mínima de C_1 en A denotada por card-min(C_1, A), es el menor número de correspondencias en las que cada elemento de C_1 puede tomar parte. Asimismo la cardinalidad mínima de C_2 en A, representada como card-min(C_2, A), es el menor número de correspondencias en las que cada elemento de C_2 puede participar.

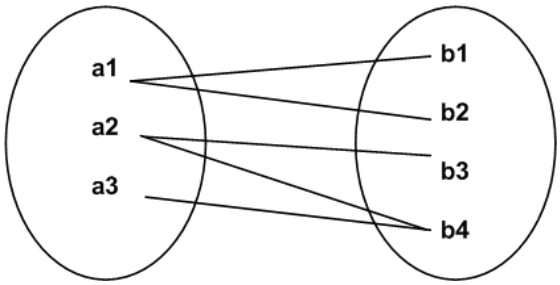


Figura 8.6: Agregación binaria

De la misma manera, la cardinalidad máxima de C_1 en A denotada por card-max(C_1, A), es el mayor número de correspondencias en las que cada elemento de C_1 puede tomar parte. Asimismo la cardinalidad máxima de C_2 en A, re-

presentada como card-min(C_2, A), es el mayor número de correspondencias en las que cada elemento de C_2 puede participar.

Los valores importantes de card-max(C_1,A) son 1 y n; n representa un número natural e indica que cada elemento de C_1 puede pertenecer a una cantidad arbitrariamente grande de correspondencias

8.3. LOS ELEMENTOS BÁSICOS DEL MODELO

Los conceptos básicos provistos por el modelo entidad interrelación son las entidades, las interrelaciones y los atributos. Tanto las entidades como las interrelaciones son clases de objetos.

8.3.1. LAS ENTIDADES

Las entidades son objetos del mundo real con existencia propia, pueden tener existencia física, por ejemplo, las personas o existencia conceptual, por ejemplo una empresa. Las entidades representan clases de objetos de la realidad. Se representan gráficamente por medio de un rectángulo con el nombre de la entidad dentro de él (figura 8.7).

Figura 8.7: Representación de una entidad

8.3.2. LAS INTERRELACIONES

Las interrelaciones representan agregaciones de dos o más entidades. Describen conjuntos de asociaciones o participaciones entre una entidad E_1 y una entidad E_2. La cardinalidad C_1= min : max indica que cada instancia de E_1 puede tener al menos min y como máximo max participantes en R_1. La interrelación inversa se describe mediante R_2 y, además, C_2 involucra los mismos pares de entidades que R_1

Las interrelaciones se representan gráficamente con rombos (figura 8.8). Las interrelaciones n-arias conectan más de dos entidades. La relaciones unarias son interrelaciones que conectan una entidad con sí misma.

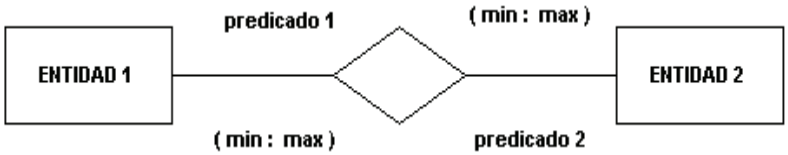


Figura 8.8: Ejemplo de interrelación

Por ejemplo, un alumno puede recibir clases en una sola división, pero esa división puede tener muchos alumnos o no estar asignada a ningún curso, si suponemos que un alumno tiene que tener asignado, al menos, alguna división, su representación en el modelo será como muestra la figura 8.9.

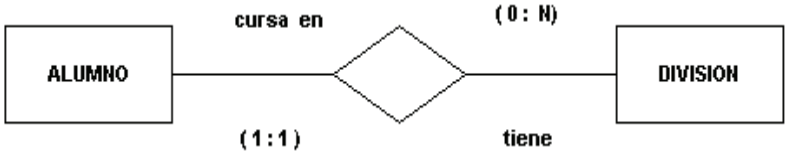


Figura 8.9: Ejemplo de interrelación binaria 1:N

Si se quisiera, por otro lado, modelar que un empleado tiene como supervisor a otro empleado (figura 8.10), su representación sería mediante una interrelación unaria.

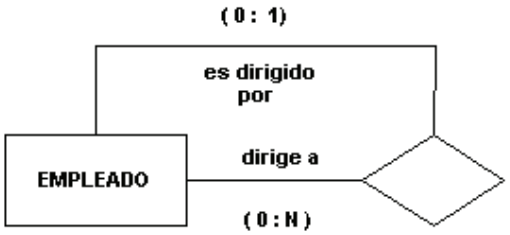


Figura 8.10: Ejemplo de interrelación unaria 1:N

Como ejemplo de interrelación ternaria, se muestra a un alumno que cursa una materia con un profesor, se representa mediante el modelo como lo grafica la figura 8.11.

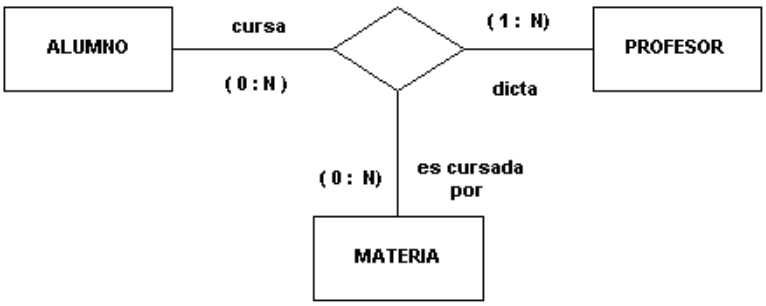


Figura 8.11: Ejemplo de interrelación ternaria

8.3.3. LOS ATRIBUTOS

Los atributos (figura 8.12) representan las propiedades básicas de las entidades o interrelaciones. Toda la información extensiva, tanto para las entidades como para las interrelaciones es aportada por ellos.

De la misma forma que las interrelaciones, los atributos se caracterizan por su cardinalidad máxima y mínima. La cardinalidad mínima indica el número mínimo de valores de atributos asociados con una entidad o interrelación. Sea A un tributo de la entidad E; si card-min(A, E) = 0, el atributo es opcional y puede estar no especificado, es decir, tener un valor nulo para algunas instancias de la entidad. Si, por el contrario, card-min(A, E) = 1 el atributo es obligatorio y al menos un valor del atributo debe especificarse para todos los casos de la entidad. Las mismas definiciones se aplican a los atributos de las interrelaciones.

La cardinalidad máxima indica el número máximo de valores de atributos asociados con cada entidad o interrelación. Sea A un atributo de la entidad E; si card-max(A, E)= 1, el atributo es monovalente, esto significa que en cada instante, el valor será único, si card-max(A, E) > 1, el atributo es polivalente o multivaluado, lo cual significa que ese atributo podrá tener más de un valor asociado en un mismo instante. Las mismas definiciones se aplican a los atributos de las interrelaciones.

Cada atributo se asocia a un dominio particular, es decir, el conjunto de valores legítimos para ese atributo. Un atributo simple se define sobre un dominio simple. Dado un atributo A su dominio puede declarase como dom(A) y definirse como el conjunto de valores válidos de A.

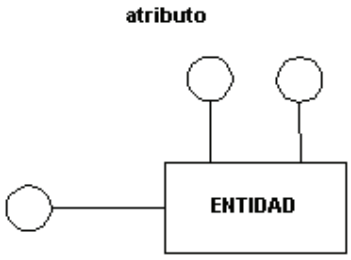


Figura 8.12: Atributos en una entidad

Por ejemplo (figura 8.13), si se considera la entidad ALUMNO, la card-min (apellido paterno, ALUMNO) = 0, esto es, no obligatorio, lo que significa que pueden existir, en el modelo, valores de apellido paterno no definidos para algunos alumnos; la card-min (nombre, ALUMNO) = 1, es obligatorio, por lo tanto, siempre debe existir un valor para ese atributo; la card-max (nombre, ALUMNO) = 1, es monovaluado, esto es, existirá un sólo valor por atributo; por último, la card-max (teléfono, ALUMNO) > 1, es multivaluado, es decir, el alumno puede tener más de un teléfono.

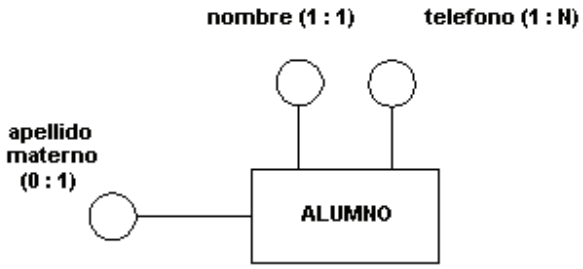


Figura 8.13: Cardinalidad en los atributos

8.4. LA JERARQUÍA DE GENERALIZACIÓN

En el modelo entidad interrelación es posible establecer jerarquías de generalización entre las entidades (figura 8.14). Una entidad E es una generalización de un grupo de entidades E_1, E_2, ... E_n, si cada objeto de la clase E_1, E_2, ... E_n es también un objeto de la clase E.

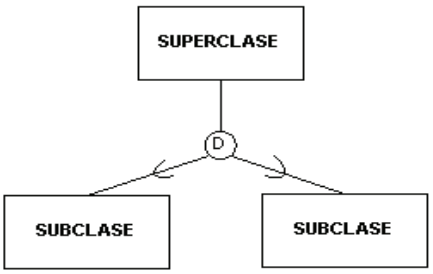


Figura 8.14: Clase y subclases

Cada entidad puede participar en múltiples generalizaciones, posiblemente en el papel de entidad genérica con respecto a una generalización y el papel de entidad subconjunto con respecto a otra generalización.

Las jerarquías de generalización se caracterizan por la propiedad de cobertura. Las generalizaciones pueden ser total (t), parcial (p) y exclusiva (e) o superpuesta (s), el par que se da con mas frecuencia es (t, e) que se considera valor por omisión.

La propiedad fundamental de la abstracción de generalización es que todas las propiedades de la entidad genérica son heredadas por las entidades subconjunto. En términos del modelo entidad interrelación, esto significa que cada a tributo, interrelación o generalización definido para la entidad genérica es heredado automáticamente por todas las entidades subconjunto de la generalización. Esta propiedad es importante, porque permite construir jerarquías de generalización estructuradas.

Por ejemplo (figura 8,15), se puede establecer una jerarquía de clase y subclase con la entidad ESTUDIANTE, éste puede ser de GRADO o POSGRADO

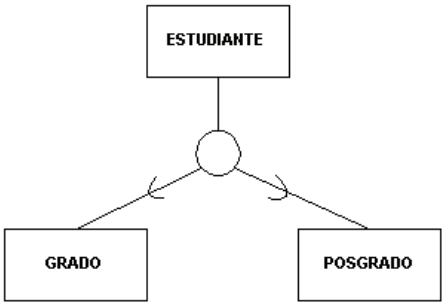


Figura 8.15: Clase y subclases

8.5. LOS ATRIBUTOS COMPUESTOS

Los atributos compuestos (figura 8.16), son grupos de atributos que tienen afinidad en cuanto a su significado o a su uso.

Las cardinalidades mínimas y máximas se aplican a los atributos compuestos con las mismas definiciones dadas para los atributos simples. Pero, asignar cardinalidades mínimas y máximas a los atributos compuestos agrega más capacidades de modelado en comparación con la asignación de cardinalidad a un atributo individual.

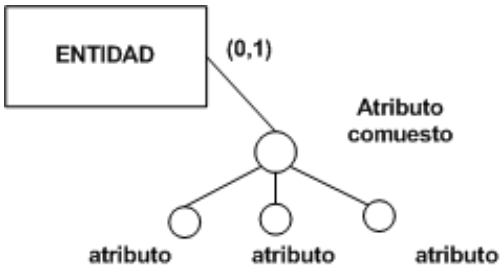


Figura 8.16: Atributo compuesto

Por ejemplo (figura 8.17), se puede modelar la dirección de la entidad ALUMNO como atributo compuesto, sus componentes son calle y número, ambos monovaluados y obligatorios y el piso opcional, en el caso en que no viva en un departamento.

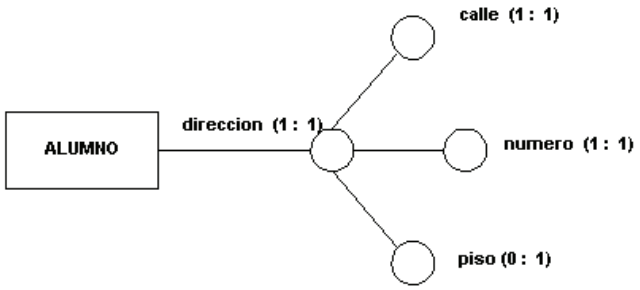


Figura 8.17: Atributo compuesto

8.6. LOS IDENTIFICADORES

La restricción de clave establece que debe existir algún atributo, quizás compuesto, que identifique unívocamente a una entidad de otra. Este atributo se lo denomina identificador.

Especificar que un atributo es clave significa que la propiedad de unicidad se debe cumplir para todas las instancias de la clase entidad. Un identificador de una entidad E, para ser identificador, debe cumplir las siguientes condiciones:

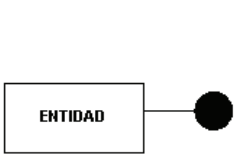
1) Unicidad, esto es, no pueden existir dos instancias de la entidad E con el mismo valor del identificador
2) Minimalidad, es decir, si se omite cualquier atributo A_i del identificador, la condición 1 deja de cumplirse.

Las entidades pueden tener múltiples atributos identificadores alternativos (figura 8.18). Se clasifican en:

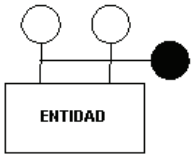
a) Simple, cuando la clave está compuesta por un atributo simple
b) Compuesto, cuando la clave esta compuesta por más de un atributo
c) Interno, si el atributo identificador pertenece a la entidad
d) Externo, si el atributo identificador pertenece a otra entidad.

Las entidades que pueden identificarse internamente se denominan entidades fuertes. Por el contrario, las entidades que tienen sólo identificadores externos se las denomina entidades débiles.

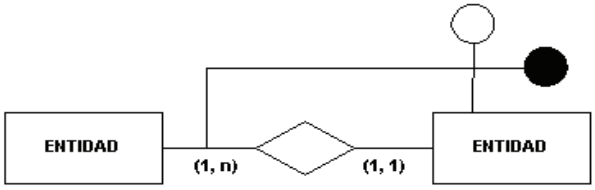
Debido a que la identificación es una propiedad de las entidades, es una de las propiedades heredadas en las generalizaciones, de modo tal que el identificador de la entidad genérica es, asimismo, un identificador de las entidades subconjunto.



Identificador simple e interno



Identificador compuesto e interno



Identificador compuesto, externo y mixto

Figura 8.18: Identificadores

8.7. LAS ENTIDADES COMPLEJAS

Una entidad compleja (figura 8.19) es una abstracción que permite representar la composición de entidades e interrelaciones.

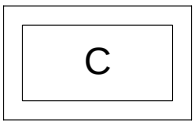


Figura 8.19: Entidad compleja

8.8. LAS ENTIDADES E INTERRELACIONES DERIVADAS

Las entidades o interrelaciones derivadas (figura 8.20), representan objetos derivados que no están almacenados en la base de datos.

Figura 8.20: Entidades e interrelaciones derivadas

8.9. RESUMEN

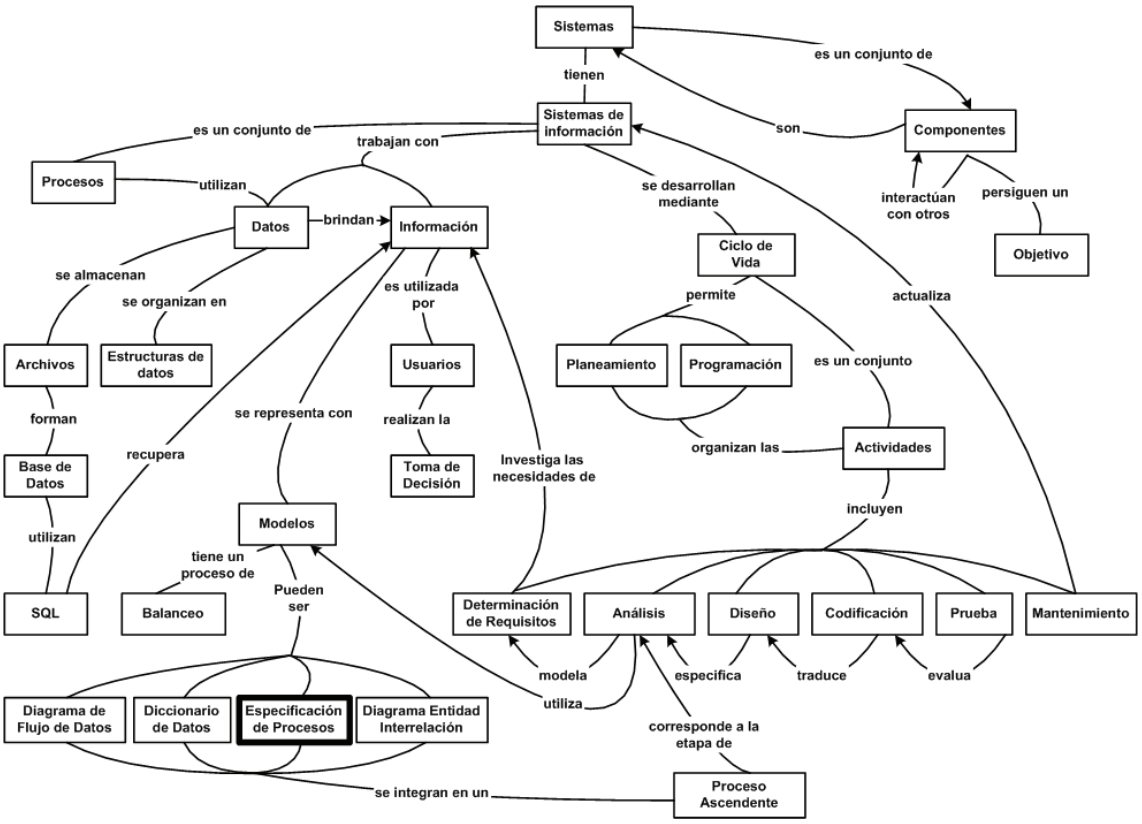
El modelo entidad interrelación se construye a partir de los requerimientos del usuario. Es un modelo conceptual de alto nivel basado en las abstracciones de clasificación, agregación y generalización e independiente de la futura implementación.

Este modelo, como representación conceptual, permite la descripción de las estructuras de datos y sus vínculos mediante entidades, atributos e interrelaciones entre esas entidades.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 9



Especificación de Procesos Estructurados

La especificación de requisitos de los usuarios es el documento que describe las características funcionales de un sistema de información que va a ser desarrollado. Es un documento fundamental ya que, por un lado representa el contrato acordado con el cliente, y por otro sirve de base a todas las etapas posteriores de la ingeniería de software.

El principal objetivo de la especificación de procesos estructurados es definir, de forma clara y no ambigua, las funciones y restricciones del sistema, de forma tal de evitar problemas en las etapas de diseño y codificación.

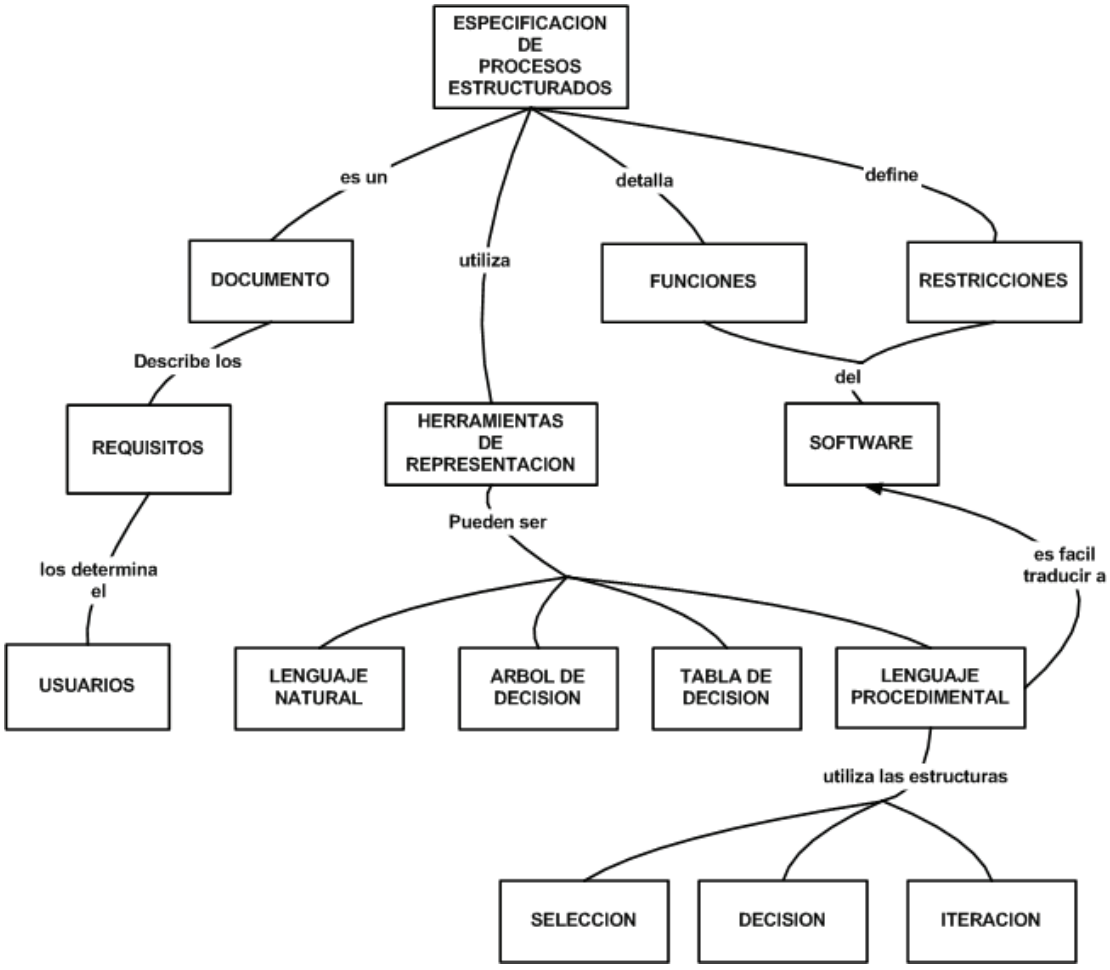


Figura 9.1: Mapa conceptual de “Especificación de Procesos Estructurados”

La especificación de procesos estructurados es el resultado de las tareas de análisis. Para comprender mejor lo que el usuario necesita, se divide el problema en partes y se desarrollan representaciones o modelos que muestren la esencia de los requisitos. Si éste no refleja los requerimientos del cliente entonces, inevitablemente, el diseñador construirá un sistema incorrecto. Por otra parte, si la especificación es incompleta, ambigua o inconsistente, aunque haya sido aceptada por el cliente, no se podrá satisfacer las necesidades en forma adecuada.

Cada una de las dudas, producto de una mala especificación, plantea una serie de alternativas y obliga a elegir entre varias posibilidades durante el diseño y la implementación. Pero estas decisiones no se tomarán con un entendimiento completo del problema, debido a que las realizan diseñadores y programadores que nunca han estado en contacto con el cliente, por lo que el resultado será una implementación que, con seguridad, no cumplirá las necesidades del usuario. Por lo tanto, la forma de especificar un sistema tiene una gran influencia en la calidad de la solución implementada finalmente.

9.1. LOS LENGUAJES DE ESPECIFICACIÓN

La especificación consiste en describir un sistema de forma tal que queden expresadas su funcionalidad, sus restricciones y su rendimiento de la forma más clara y precisa posible. Para ello se debe utilizar algún tipo de lenguaje, existen varias alternativas, cada una de ellas con sus ventajas e inconvenientes.

9.1.1. EL LENGUAJE NATURAL

La solución más intuitiva para la descripción de procesos es utilizar el lenguaje natural, esta es la opción que se ha utilizado tradicionalmente. Entre sus ventajas se puede citar la facilidad de uso y entendimiento, no se debe aprender ningún nuevo lenguaje y cualquier persona puede leer la especificación, para luego comentarla o criticarla. Entre los inconvenientes están la imprecisión y la ambigüedad.

Aunque el análisis de requisitos se haya realizado correctamente, una especificación en lenguaje natural, por las características antes señaladas, puede dar lugar a que la implementación final no cumpla con los requerimientos. Además, debido a su propia facilidad de uso e imprecisión, las especificaciones suelen ocultar características del sistema que sólo se pondrán de manifiesto a la hora de implementarlas, es decir, al traducir la especificación a un lenguaje de programación.

9.1.2. LOS LENGUAJES PROCEDIMENTALES

La especificación del proceso debe expresarse de manera tal que pueda ser verificado tanto por el usuario como por el analista. Precisamente, por esta ra-

zón, se evita el lenguaje narrativo como herramienta de especificación, ya que es ambiguo, sobre todo si describe acciones alternativas y repetitivas. Por su naturaleza, también tiende a causar gran confusión cuando expresa condiciones booleanas compuestas, esto es, combinaciones de los operadores and, or y not.

Los lenguajes de programación tienen una sintaxis carente de ambigüedad, más una semántica bien definida. Con el fin de poder acercar la especificación a la implementación, se podrían utilizar esos mismos lenguajes, o un pseudocódigo basado en ellos, para especificar sistemas. De esta forma, al aproximar la especificación a la implementación se reducen los errores en la codificación.

Otra ventaja es que, al utilizar un lenguaje de programación, es mucho más fácil definir la interfaz de un proceso, ésta está formada por el conjunto de los parámetros de dicha especificación y, además, se podría comprobar si todos los datos necesarios para realizar la transformación están indicados en dicha interfaz.

Los principales inconvenientes son tres. Por un lado, los lenguajes de programación no son lo suficientemente formales como para poder deducir, a partir de él, propiedades de completitud, consistencia o corrección de una especificación. Por otra parte, el uso de lenguajes procedimentales obliga a definir un algoritmo a la hora de especificar un proceso, pero el objetivo de la especificación no es definir el cómo se ha de implementar el sistema sino definir qué se ha de implementar. Otro inconveniente es que el lenguaje utilizado para la especificación sea empleado como lenguaje final de implantación no siendo, quizá, el más apropiado.

Otra alternativa para la descripción de las funciones de un sistema es la especificación del proceso mediante pseudocódigo. Este es un lenguaje similar a los convencionales, con la diferencia de que no tiene una sintaxis tan rígida. Permite la descripción de qué es lo que sucede en cada burbuja primitiva del diagrama de flujo de datos, sin exigir decisiones prematuras. Su propósito es definir lo que debe hacerse para transformar entradas en salidas.

La descripción mediante seudocódigo pueda ser comunicada adecuadamente al amplio público que está involucrado en el desarrollo del sistema. A pesar de que el analista es típicamente quien escribe la especificación del proceso, habitualmente serán diversos tipos de usuarios, tales como los administradores, los auditores, el personal de control de calidad y otros, los que leerán la especificación del proceso.

Las construcciones utilizadas en las especificaciones de procesos mediante seudocódigo son las estructuras de secuencia, iteración y decisión.

9.1.2.1 LA ESTRUCTURA DE SECUENCIA

La estructura de secuencia (figura 9.2), se caracteriza por tener una entrada y una salida dentro de la cual se encuentran una serie de acciones cuya ejecu-

ción es lineal y en el orden en que aparecen. A su vez, todas las acciones tienen una única entrada y una única salida.

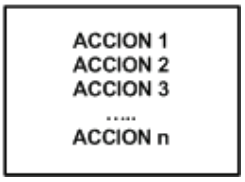


Figura 9.2: Estructura de secuencia

9.1.2.2 LA ESTRUCTURA DE DECISION

La estructura de decisión también tiene una sola entrada y una sola salida, pero dentro de la misma se realiza una acción de entre varias, según una condición preestablecida. Esta condición puede ser simple o compuesta. Las estructuras de decisión pueden ser de dos salidas, en la que una de ellas puede ser la acción nula o de tres o más salidas, que se denominan múltiples.

Se detallarán a continuación las distintas estructuras de decisión.

La estructura "SI ENTONCES SI-NO FIN-SI" (figura 9.3), admite dos salidas y comienza con la palabra reservada SI seguida de la condición que debe probarse. La alternativa que debe ser ejecutada, si la condición resulta verdadera, se encuentra precedida por la palabra ENTONCES. La otra alternativa que debe realizarse, si la condición es falsa, esta precedida por la palabra SI-NO

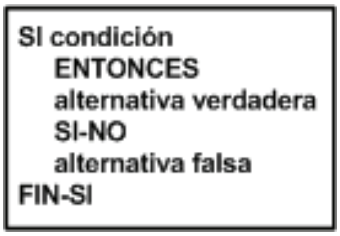


Figura 9.3: Estructura de decisión

Puede suceder (figura 9.4) que, si la condición es falsa, no existan acciones a ejecutar luego del SI-NO, entonces

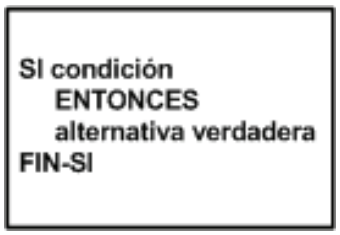


Figura 9.4: Estructura de decisión reducida

Tanto la alternativa verdadera como la falsa, pueden tener a su vez estructuras de decisión.

La estructura de decisión generalizada o múltiple (figura 9.5), tiene el siguiente formato general:

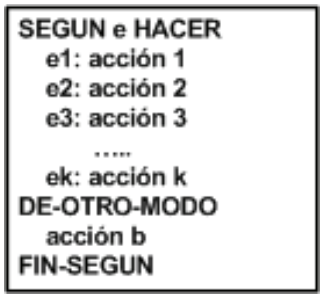


Figura 9.5: Estructura de decisión

En esta estructura, existe una expresión “e” cuyos resultados pueden ser los valores e1, e2,..., ek. En función de estos valores se ejecutará una y sólo una de las acciones indicadas. Si el valor de “e” es distinto de e1, e2,..., ek, se ejecutará la acción b.

9.1.2.3 LA ESTRUCTURA DE REPETICIÓN

Otra de las construcciones utilizadas en la estructura de repetición. En esta estructura existe una entrada y una salida dentro la cual se repite una acción que generalmente es una estructura de secuencia, un número determinado o indeterminado de veces, dependiendo en este caso del cumplimiento de una condición.

Las estructuras de repetición pueden ser:

La estructura PARA (figura 9.6), es una estructura que ejecuta un conjunto de acciones un número “n” de veces conocido anticipadamente.

```
PARA v DESDE vi HASTA vf CON-PASO p HACER
        acción 1
        acción 2
        acción 3
           ....
        acción k
FIN-PARA
```

Figura 9.6: Estructura de repetición “PARA”

Donde v es una variable numérica, llamada variable de condición, vi y vf son variables de tipo numérico, constantes numéricas o expresiones aritméticas. vi es el valor inicial, vf es el valor final y p es el paso. p puede ser positivo o negativo, pero no nulo.

La estructura MIENTRAS (figura 9.7) es una estructura que repite una acción mientras se cumpla la condición que controla el bucle. La característica principal de esta estructura es que la condición es evaluada siempre antes de cada repetición.

El número de repeticiones de esta construcción oscila entre 0 e infinito, dependiendo de la evaluación de la condición cuyos argumentos, al menos una vez, deberán modificarse dentro del bucle, pues de no ser así el número de repeticiones será infinito constituyéndose en un bucle sin salida.

```
MIENTRAS condición HACER
        acción 1
        acción 2
        acción 3
           ....
        acción k
FIN-MIENTRAS
```

Figura 9.7: Estructura de decisión “MIENTRAS”

La estructura REPETIR HASTA (figura 9.8), es una estructura que repite una acción hasta que se cumpla la condición que controla el bucle, la cual se evalúa después de cada ejecución del mismo. El número de repeticiones oscila entre 1 e infinito, dependiendo de la evaluación de la condición, cuyos argumentos deberán modificarse dentro del bucle, pues de no ser así, también, el número de repeticiones será infinito estando nuevamente en presencia de un bucle sin salida.

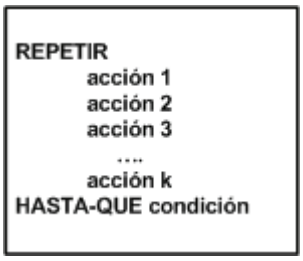


Figura 9.8: Estructura de decisión “REPETIR”

9.3. EL ÁRBOL DE DECISIÓN

El árbol de decisión (figura 9.9), es un diagrama que muestra en forma secuencial condiciones y acciones, presenta qué condiciones se consideran en primer lugar, cuál en segundo y así sucesivamente hasta que al final se despliega una acción determinada.

Esta herramienta de especificación se utiliza principalmente para organizar la información recopilada en la etapa de requerimientos y poder entender las combinaciones de condiciones cuando éstas no son muy complejas y no todas son posibles.

La construcción del árbol comienza desde la raíz, donde se inicia también la secuencia de decisión, la rama a seguir dependerá de las condiciones y de la decisión que deba tomarse. Las hojas, al final del árbol, representan las acciones a seguir.

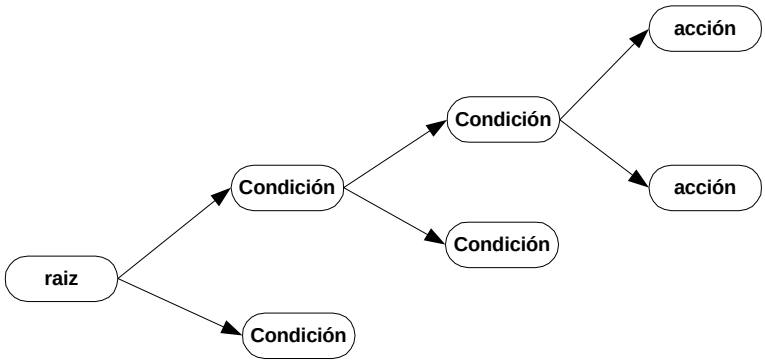


Figura 9.9: Árbol de decisión

9.4. LA TABLA DE DECISIÓN

La tabla de decisión (figura 9.10) se utiliza cuando se deben especificar procesos complejos. Está compuesta por cuatro cuadrantes. En el primer cuadrante,

está la combinación de condiciones, aquí se enlistan todas las posibles combinaciones de condiciones, que por ser éstas booleanas tendrá 2^n columnas.

El segundo cuadrante, denominado identificación de condiciones, señala las condiciones que se consideran relevantes, éstas deben ser booleanas, es decir, condiciones que solamente pueden ser verdaderas o falsas.

En el tercer cuadrante, se ubica la identificación de acciones, en él se detallan todas las posibles acciones a realizar.

Por último, En el cuarto cuadrante se ubican las acciones elegidas. Aquí se realiza una marca en la intersección de la combinación de las condiciones y la acción a realizar después de concluir el análisis.

Figura 9.10: Tabla de decisión

Para construir la tabla (figura 9.11), se parte de la identificación de todas las condiciones relevantes, la respuesta a estas condiciones debe ser verdadera o falsa. La cantidad de condiciones limita la construcción práctica de la tabla de decisión; el número de combinaciones de condiciones crece en forma exponencial (2 condiciones, 4 combinaciones, 3 condiciones, 8 combinaciones, 4 condiciones, 16 combinaciones, etc.)

Luego se identifican las acciones a seguir. La cantidad de acciones no constituye un limitante práctico en la construcción de la tabla. Por último, se marcan, para cada combinación de condiciones, la o las acciones a realizar.

Luego de terminada la construcción de la tabla, se analizan las redundancias y contradicciones que pudieran existir. Hay redundancia cuando aparecen las mismas condiciones para dos o más combinaciones y ésta difieren en solo una condición. Existe contradicción cuando un conjunto de condiciones no puede ocurrir nunca. Esto puede suceder cuando se trabaja con rango de valores, por ejemplo fechas o valores numéricos.

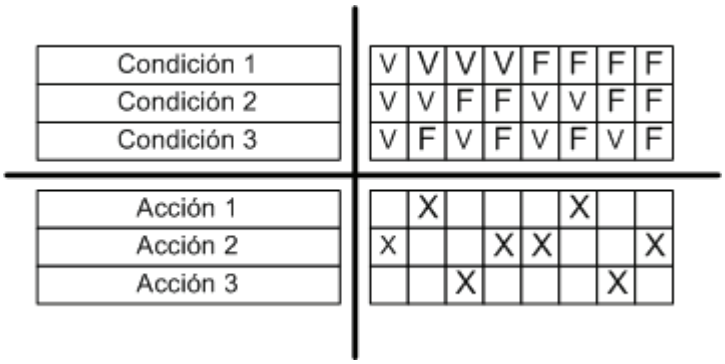


Figura 9.11: Ejemplo de tabla de decisión con tres condiciones

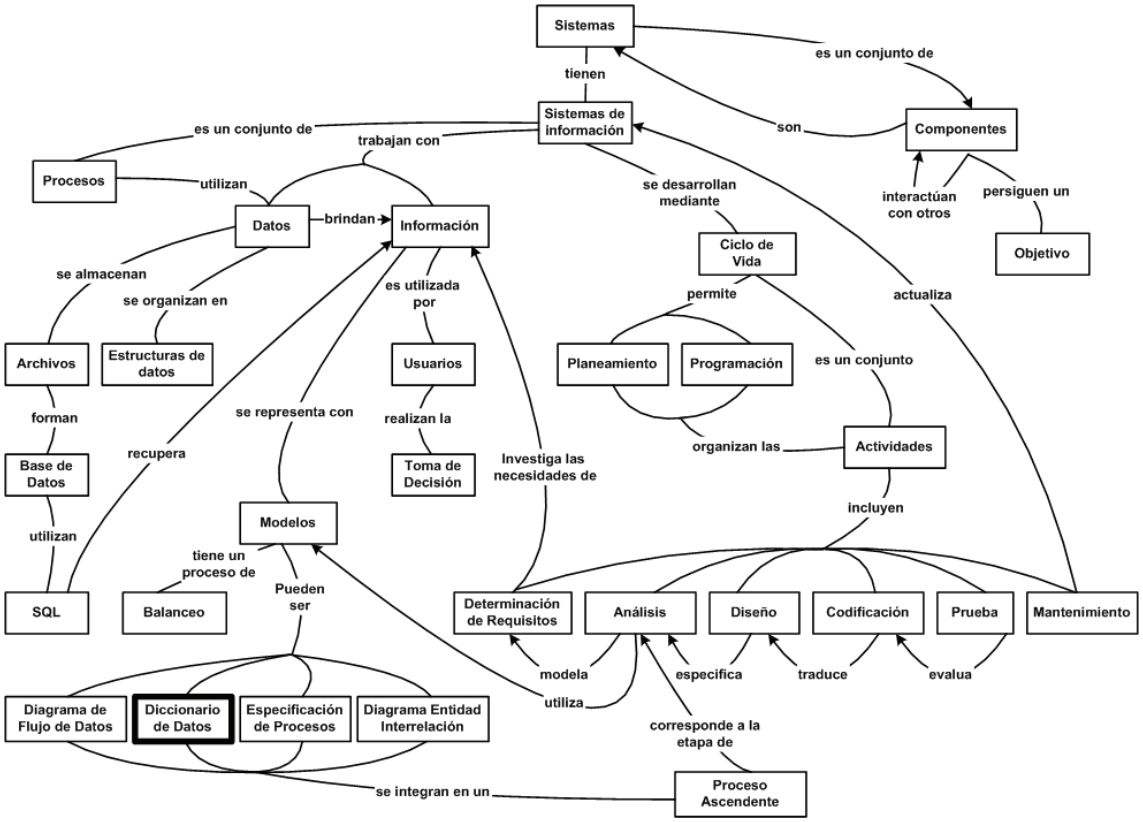
9.5. RESUMEN

El objetivo principal de la especificación de procesos es definir de forma clara y no ambigua las funciones y restricciones del sistema, de forma que se eviten problemas en las etapas de diseño y codificación. La especificación es el resultado del proceso de análisis. Una especificación es, por definición, una descripción de lo que se quiere realizar, no de cómo se va a realizar o implementar. Se pueden utilizar, para su descripción, un conjunto de herramientas tales como el lenguaje natural, al lenguaje procedimental, el árbol de decisión y la tabla de decisión.

Lectura requerida

Neil C. G.; **Análisis de Sistemas. Un enfoque conceptual**. 2da. Edición. Buenos Aires, UAI, 2005.

Capítulo 10



Diccionario de Datos

Los diferentes modelos del análisis estructurado permiten tener una perspectiva del sistema desde diferentes puntos de vista. Cada uno de estos modelos enfoca una visión determinada del sistema en estudio. Toda la información que complementa a los modelos gráficos se encuentra organizada y detallada en el diccionario de datos.

El diccionario de datos, por lo tanto, es un listado organizado que contiene los detalles del sistema, con definiciones precisas y rigurosas para que, tanto el

usuario como el analista, tengan un entendimiento común de todas las entradas, salidas, componentes de los almacenamientos y procesos intermedios que existen en el sistema.

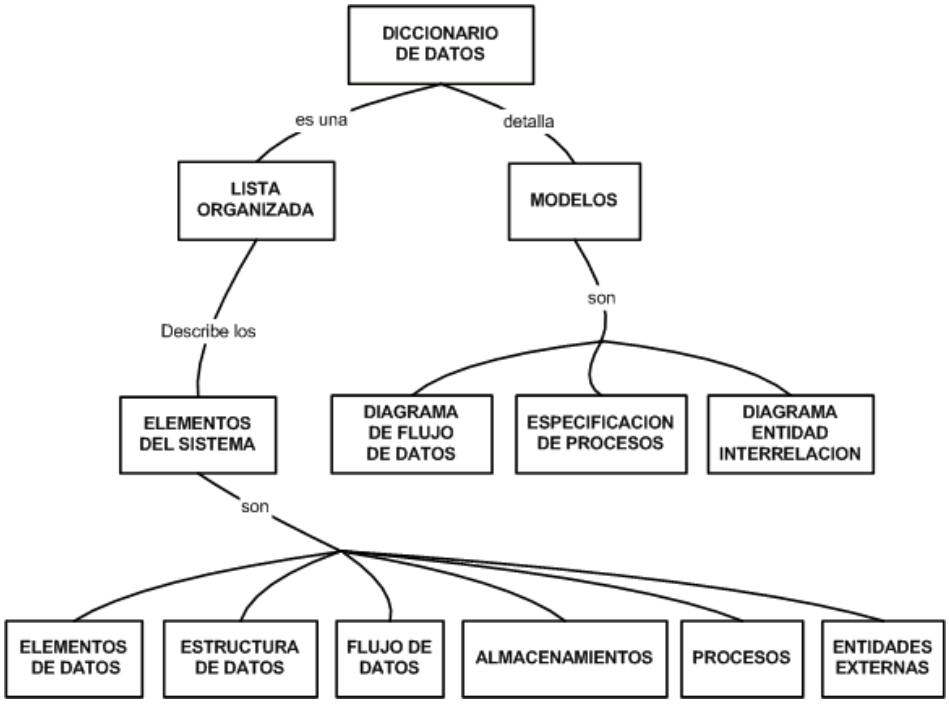


Figura 10.1: Mapa conceptual de “Diccionario de Datos”

El diccionario de datos define el significado de los flujos y almacenamientos que se muestran en los diagramas de flujo de datos; detalla la composición de la agrupación de estructura de datos que se mueven a lo largo de los flujos, es decir, estructuras complejas que pueden descomponerse en unidades más elementales. Describe, además, la composición de los almacenamientos y especifica los valores y unidades relevantes de los datos elementales que componen los flujos de datos y los almacenamientos de datos. Detalla, también, las relaciones entre entidades que se enfatizan en el diagrama de entidad-interrelación

10. 1. LA NOTACIÓN DEL DICCIONARIO DE DATOS

El diccionario de datos posee una notación simple que permite describir cada uno de los componentes del sistema en estudio.

10.1.1. EL DATO ELEMENTAL

El dato elemental es la mínima unidad indivisible, participa como componente en las estructuras de datos del sistema. Cualquier descomposición del dato

elemental carecerá de sentido dentro del ámbito de estudio. Son sinónimos de datos elementales: elemento de dato, campo, atributo, etc. El dato elemental es el componente con mayor nivel de detalle, de él se especifica el nombre, su descripción, el alias, su longitud y tipo y el dominio de valores admisibles. El nivel de detalle dependerá de las necesidades del analista y muchas veces de la herramienta utilizada para la documentación.

Se deben asignar, a los datos elementales, nombres que sean significativos en el contexto del desarrollo del sistema. Por ejemplo, son nombres válidos: apellido, fecha-factura, no es conveniente utilizar, en la descripción de los datos elementales, nombres enigmáticos o carentes de significado.

Por ejemplo: XXY, fec-noc-ter.

La descripción del elemento de datos indica, de manera breve, lo que éste dato elemental representa para el sistema. Permite tener una descripción del significado que se le atribuye en el sistema al elemento de dato. El símbolo que se utiliza para realizar comentarios textuales en la definición de los componentes es "* *".

Por ejemplo: Apellido = *apellido del empleado*", DNI = *documento nacional de identidad*.

El alias se utiliza cuando existe, en la organización, más de una forma de denominar a un dato elemental.

Por ejemplo: fecha de nacimiento = *alias de fech_nac*

El longitud y el tipo detalla, independientemente de la implementación en un lenguaje de programación determinado, la cantidad de espacio asignado a él y el tipo de dato que representa. Este puede ser, entre otros, alfanumérico, numérico; fecha, boolean, etc.

Por ejemplo: Nombre, alfanumérico (30), Edad, numérico (2).

Existen otras formas de representar tanto la longitud como el tipo de datos, esto es, mediante un comentario.

Por ejemplo: Nombre = *alfanumérico de 30 espacios*, Edad = *numérico de 2 espacios*

El dominio detalla el conjunto de valores permitidos para cada dato elemental. El dominio puede ser de valores discretos o continuos. Cuando es de valores discretos se especifica detallando por extensión, mediante un par ordenado, los posibles valores y su significado. Por ejemplo: estado-civil = {(s, soltero); (c, casado), ...}. Cuando los valores permitidos son continuos se especifica el rango mediante el valor inicial y valor final que puede tomar.

Por ejemplo: fecha-nac = {vi:17/06/98; vf: 19/02/99}

10.1.2 LA ESTRUCTURA DE DATOS

Los datos elementales se agrupan en estructuras para describir componentes del sistema. Una estructura de datos está compuesta por elementos y/o estructuras de datos. Estas estructuras se construyen sobre un conjunto de relaciones entre los componentes. La definición de una estructura de datos comienza con el símbolo "=", este tiene el significado de "está compuesto por".

La relación secuencial define los componentes que siempre estarán incluidos en la estructura. Pueden ser datos elementales o estructuras de datos. A esta relación se denomina también concatenación. El símbolo que se utiliza para su descripción es "+".

Por ejemplo:
Dirección = calle + número + código_postal + localidad
Alumno = código + nombre + dirección

La relación de selección define distintas alternativas para datos elementales o estructura de datos incluida dentro de una estructura. De todas las opciones, solamente se elige una. La relación de selección se puede considerar, para los datos elementales, como un dominio de valores discretos. El símbolo que se utiliza para su descripción es "[ ]". Para separar opciones alternativas en la construcción de selección, se utiliza el símbolo "|"

Por ejemplo:
Documento = [dni | cedula de identidad | pasaporte]
sexo = [Femenino | Masculino]

La relación de repetición define la iteración de un dato elemental o una estructura de datos cero o más veces dentro de una estructura. El símbolo que se utiliza para su descripción es "v_i{ }v_f"; siendo v_i el valor inicial y v_f el valor final .

Por ejemplo:
Solicitud = nombre del cliente + domicilio de envío + 1{artículo}10.
Significa que puede requerirse entre 1 y 10 artículos en la solicitud.

La relación opcional indica que el dato elemental o la estructura de datos puede estar o no presente dentro de una estructura. Es un caso especial de repetición con $v_i = 0$ y $v_f = 1$. El símbolo que se utiliza es "( )".

Por ejemplo:
domicilio del cliente = domicilio de envío + (domicilio de facturación)
O, lo que es igual:
domicilio del cliente = domicilio de envío + 0{domicilio de facturación}1

El identificador indica un campo único en los almacenamientos, que puede ser compuesto y tiene la particularidad de no poder repetirse y tampoco tener valores nulos. Corresponde, en el modelo relacional, a la clave primaria. El símbolo que se utiliza para su descripción es "@"

Por ejemplo:
Socio = @código + nombre + apellido + dirección

10.1.3. EL FLUJO DE DATOS

Los flujos de datos son conductos por donde pasan los elementos de datos o las estructuras de datos que comunican a los componentes del diagrama de flujo de datos. En este diagrama, los flujos se representan mediante líneas con flechas que indican su dirección. Se describe, en el diccionario de datos, mediante el nombre, contenido, origen y destino

El nombre se utiliza para referenciar un flujo de datos. Se deben asignar nombres de flujos que sean significativos en el contexto del desarrollo del sistema. Por ejemplo: Datos del socio. La descripción indica, de manera breve, lo que ese flujo representa para el sistema. Por ejemplo: Datos de los socios del club El contenido, describe los componentes del flujo, pueden ser elementos de datos o estructuras de datos.

Por ejemplo:
Datos del socio = nombre + apellido + ... + dirección

La fuente indica el origen del flujo de datos, puede ser una entidad externa, un proceso, o un almacenamiento. El destino indica hacia donde se dirige el flujo de datos, puede, al igual que la fuente, ser una entidad externa, un proceso, o un almacenamiento.

10.1.4. EL ALMACENAMIENTO

Los almacenamientos son flujos de datos en reposo. Se representan de la misma forma que los flujos de datos. Por lo tanto tienen nombre, descripción, contenido y además se detallan los flujo de entrada y salida.

El flujo de entrada indica cuáles son los flujos que alimentan al almacenamiento. Esto implica una alteración del contenido. Puede ser una inserción, borrado o modificación de los valores componentes de la estructura.

El flujo de salida indica cuáles son los flujos que se extraen del almacenamiento. Esto implica, solamente, la lectura de los valores componentes de la estructura sin alterar el contenido de los mismos.

10.1.5. LOS PROCESOS

Los procesos son los transformadores de flujos de datos del sistema. Los procesos primitivos del diagrama de flujo de datos, serán descriptos, además, en forma más detallada mediante alguna de las herramientas de especificación. En general, se define el nombre, la descripción y los flujos de entrada y salida. Se deben asignar nombres a los procesos que sean significativos en el contexto del desarrollo del sistema. En general se recomienda que sean verbos más objeto sobre el cual éste actúe. Por ejemplo: Inscribir socio

La descripción indica, de manera breve, lo que este proceso representa para el sistema. Por ejemplo: proceso de inscripción de socios al establecimiento deportivo
El flujo de entrada indica cuáles son los flujos que alimentan al proceso. El flujo de salida indica cuáles son los flujos que se salen del proceso.

10.1.6. LAS ENTIDADES EXTERNAS

Las entidades externas representan a los objetos externos al sistema en estudio, de tal manera que determinan sus fronteras. Pueden ser personas, organizaciones u otros sistemas. Se detalla mediante el nombre, la descripción y los flujos de datos asociados.
Se deben asignar nombres a las entidades externas que sean significativos en el contexto del desarrollo del sistema. Por ejemplo: DGI
La descripción indica, de manera breve, lo que la entidad externa representa para el sistema. Por ejemplo: Dirección General Impositiva
Los flujos de datos asociados indican cuáles son los flujos de entrada y salida asociados a esa entidad externa

10.2. RESUMEN

El diccionario de datos es una lista organizada de todos los componentes del sistema, con definiciones precisas y rigurosas para que, tanto el usuario como el analista, tengan un entendimiento común de todas las entradas, salidas, componentes de los almacenamientos y procesos intermedios. Complementa los demás modelos del sistema.