# UNIDAD V

# *ANALISIS DE REQUERIMIENTOS DE LA PROGRAMACION*

Contenido:

# 5.1 INTRODUCCION

Un desarrollo de software tiene éxito si se han comprendido perfectamente los requisitos del software. Si un programa se analiza y especifica pobremente, decepcionará al usuario y desprestigiará a quien lo ha desarrollado. No importa lo bien diseñado o codificado que esté un programa, si no se ha analizado correctamente, defraudará al cliente y frustrará al desarrollador.

Cuando desarrollamos software siempre pensamos en encauzar nuestro esfuerzo a construir el producto correcto, sin embargo frecuentemente nos damos cuenta que el producto que construimos no es el correcto cuando ya está terminado, cuando ya se han invertido muchas horas y jornadas de trabajo y se ha invertido bastante dinero.

*El correcto entendimiento* que tengamos de los requisitos y funciones que debe proporcionar el software es la clave para que podamos tomar el camino que nos lleve a construir el producto correcto.

Ese entendimiento "correcto" que necesitamos lo podemos obtener como resultado de aplicar un Análisis de Requisitos del Software.

El Análisis de Requisitos del Software es una fase de la Ingeniería de Software que enlaza la Definición del Software a nivel Sistema y el Diseño de Software en un proceso de descubrimiento, refinación, modelado y especificación.

El análisis de requisitos proporciona:

- º Especificación de la función y rendimiento.
- º Descripción de la interface con otros elementos.
- º Establece las restricciones de diseño.
- º Una representación de los datos y funciones.
- º Un medio para valorar la calidad del software terminado.

El análisis de requisitos se sintetiza en 5 tareas:

1. Reconocimiento del problema
2. Evaluación y síntesis
3. Modelización
4. Especificación
5. Revisión

1) Reconocimiento del Problema

Inicialmente se debe estudiar la *Definición del Sistema* y el *Plan del Proyecto* para comprender el sistema y revisar las razones que se usaron para hacer las estimaciones en la planeación.

Se debe además establecer contacto con el equipo del cliente para reconocer el problema tal como lo percibe el cliente.

2) Evaluación y Síntesis

El analista debe evaluar el flujo y la estructura de la información, definir todas las funciones del software, establecer las características de la interfaz y las restricciones de diseño. Esto servirá finalmente para sintetizar una solución global.

El analista debe centrarse en el "qué" ( qué se hace, qué se tiene, qué se puede hacer ) y no en el "cómo".

3) Modelización.

Durante la evaluación y síntesis de la solución, se pueden crear modelos del sistema en un esfuerzo por entender mejor el flujo de datos y de control, la operación, funciones y contenido de la información.

Esta modelización puede hacerse a través de:

- Diagramas de Flujo de Datos
- Prototipo
- Manual de Usuario Preliminar.

4) Especificación

El análisis debe, al final, proporcionar una representación que pueda ser revisada y aprobada por el cliente. Por ello para describir las características, funciones y atributos del software se escribe una *Especificación Formal de Requisitos*.

5) Revisión

Los documentos que especifican los requisitos del software sirven como base para una revisión, lo cual producirá quizás, modificaciones en la definición del proyecto y en la reevaluación del plan para determinar si las primeras estimaciones siguen siendo válidas.

Siempre es mejor descubrir los comentarios del tipo: " La idea es correcta, pero esta no es la forma en que pensé que se haría", lo más pronto posible.

## 5.2 PRINCIPIOS DEL ANALISIS

Los siguientes principios resumen la escencia del Análisis de Requisitos:

1. Debe representarse el *dominio de la información*
2. Debe definirse las *funciones* que debe realizar el software
3. Debe representarse el *comportamiento* del software
4. Deben dividirse los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas.
5. El análisis debe ir desde lo escencial hasta el detalle de implementación.

**Dominio de la Información**

El software se construye para procesar datos, para transformarlos es decir, para aceptar datos de entrada, manipularlos y producir información de salida. Pero el software también procesa acontecimientos, que controlan al sistema y no es mas que un dato binario, tal como un sensor que envía al software una señal de alarma.

Por tanto, los datos ( números, caracteres, imágenes, sonidos, etc. ) y el control ( acontecimientos ) son parte del dominio de la información.

Para conocer y determinar el dominio de la información se debe examinar desde tres enfoques:

1. Contenido de la Información
2. Flujo de la Información
3. Estructura de la Información

*Contenido de la Información*

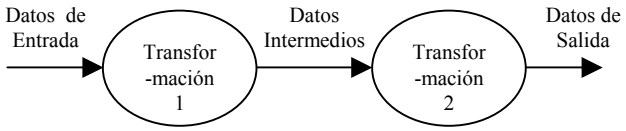
Representa los elementos de datos individuales que componen otros elementos mayores de información. Por ejemplo, el contenido de un elemento de información llamado Nómina podría ser:

*Nómina*

- Nombre del empleado
- Sueldo Neto
- Sueldo Bruto
- Deducciones

*Flujo de la Información*

Es la manera en que se representa la forma en que los datos cambian conforme se mueven a través del sistema.




Las transformaciones que se aplican a los datos son funciones que debe realizar un programa.

*Estructura de la Información*

Representa la organización interna de los distintos elementos de datos, ejemplo:

*Sueldo Neto*

- Numérico
- 6 dígitos enteros
- 2 dígitos decimales

**Modelado**

Los modelos se crean para entender mejor la entidad que se va a construir. Los modelos pueden emplear una notación gráfica que muestra información, procesamiento y comportamiento del sistema. El segundo y tercer principio de análisis requieren la construcción de los modelos funcional y de comportamiento.

*Modelo Funcional*

El modelo funcional representa las transformaciones ( entrada-procesamiento-salida ) que los datos sufren dentro del sistema. El modelo funcional empieza con un modelo sencillo a nivel contexto ( ej. El nombre del software a construir ). Después de varias iteraciones se consiguen más y más detalles funcionales, hasta representar toda la funcionalidad del sistema.

*Modelo de Comportamiento*

El software puede responder a acontecimientos del mundo exterior que hacen que el sistema cambie de un estado a otro, el estimulo-respuesta es la base del modelo de comportamiento. Un estado es un modo de comportamiento tal como calculando, imprimiendo, esperando, haciendo cola, etc. que cambia solo cuando ocurre un acontecimiento: 1) un reloj que indica que se cumple un intervalo de tiempo, 2) un movimiento de ratón, 3) un sistema externo, etc.

El modelo de comportamiento es una representación de los estados del software y los acontecimientos que causan que cambie de estado.

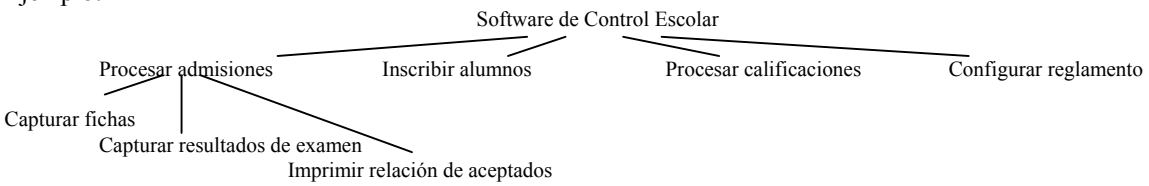
**Partición**

El cuarto y quinto principios del análisis sugieren que se puedan particionar el dominio de la información, funcional y de comportamiento. Muchas veces los problemas son demasiado grandes y complejos para entenderlos globalmente. Es más conveniente dividirlo en partes mas sencillas de manejar, analizar y entender.

La partición de la información o la función puede representarse jerárquicamente:

1. Descomponiendo el problema si nos movemos horizontalmente en la jerarquía
2. Exponiendo más detalles conforme nos movemos verticalmente

Ejemplo:




## 5.3 CONSTRUCCION DE PROTOTIPOS

Un prototipo es un sistema de trabajo que se desarrolla con rapidez para probar ideas y el entendimiento sobre el nuevo sistema, en otras palabras no solo es un diseño en papel, sino un software que corre y produce información impresa o en pantalla.

Las siguientes son razones para desarrollar prototipos de sistemas:

1. El cliente sabe que tiene un problema, pero no sabe que información necesita.
2. Requerimientos demasiado vagos como para formular un diseño.
3. Requerimientos bien definidos perso se tiene poco conocimiento para construir un sistema que cubra esos requisitos.

*Características*

1. Deben ser creado con rapidez ( unos cuantos dias ).
2. Económicos de construir.
3. No son tan eficientes como el producto final.
4. No contienen los toques finales y la estética del producto final.
5. Carecen de controles y validaciones de los datos de entrada.

*Métodos para el desarrollo de prototipos*

1. Los prototipos se pueden desarrollar utilizando lenguajes de programación y métodos convencionales.

2. Se pueden usar componentes de software reusables para ensamblar el prototipo de forma rápida, tal como un programa con rutinas de manejo de pantalla o de archivos.

3. Uso de técnicas de 4a Generación entre las cuales se encuentran los "generadores de aplicaciones" que son programas que producen otros programas. Estas herramientas permiten al analista definir la estructura de las pantallas de despliegue visual, los datos de entrada y formatos de reportes.

*Cuando construir un prototipo*

No todos los tipos de software son adecuados para la construcción de prototipos, ya que el área de aplicación, la complejidad, las características del proyecto influyen en esta decisión.

En general, cualquier aplicación que crea una representación visual dinámica, que interactúe mucho con el usuario o que requiera algoritmos desarrollados en forma evolutiva, es candidata para construir un prototipo.

## 5.4 METODOS DE ANALISIS DE REQUISITOS

Existen otros métodos alternativos a la construcción de prototipos que se utilizan también para la modelización del sistema, estos métodos usan notaciones propias para representar el flujo, contenido y estructura de la información, entre estos métodos se encuentran los siguientes:

1. Análisis Estructurado
2. Análisis Orientado a Objetos
3. Análisis Orientado a la Estructura de Datos

**Análisis Estructurado**

El método de Análisis Estructurado es un método que ha prosperado y cada vez es más ampliamente utilizado en la comunidad de Ingeniería de Software. El método se basa en la construcción de modelos utilizando una notación gráfica para modelar el flujo y contenido de la información.

Los modelos construidos deben lograr tres objetivos:
1. Describir lo que el cliente requiere
2. Establecer una base para crear el diseño del software
3. Definir los requisitos a validar cuando el software este terminado

Para lograr estos objetivos el análisis estructurado emplea los siguientes componentes:

1. El diagrama entidad relación ( para modelar los datos )
2. El diagrama de flujo de datos ( para modelar la función )
3. El diagrama de transición de estados ( para modelar el comportamiento )
4. El diccionario de datos ( para especificar los detalles de cada objeto de datos ).

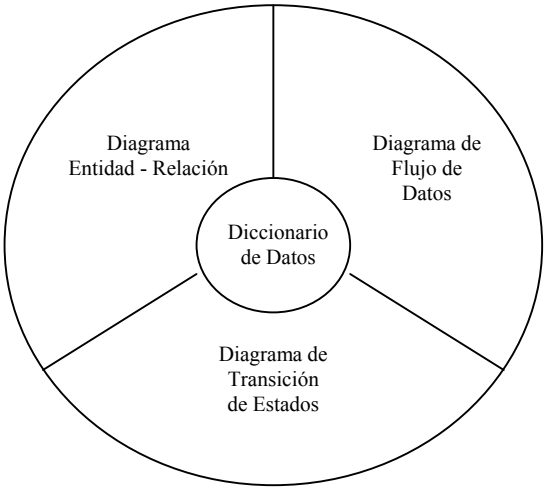



Fig.- Estructura del Modelo de Análisis Estructurado

## 5.4.1 Modelado de Datos

El modelo de datos ayuda al ingeniero de software a representar lo siguiente:

- Los objetos ( o entidades ) de datos que el sistema debe entender
- Los atributos que describen a cada objeto
- Las relaciones entre los objetos

El modelado de los datos emplea el Diagrama Entidad Relación ( DER ) como base. Como el DER se centra solo en los datos esto hace que proporcione el entendimiento del dominio de la información del problema. El DER servirá a los diseñadores del software como base para producir el diseño de la base de datos.

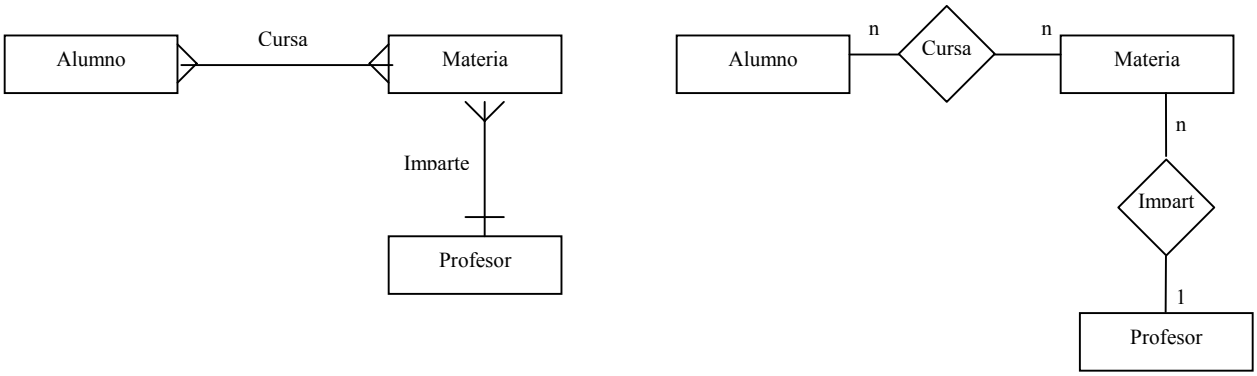
Se han propuesto diferentes notaciones gráficas para el DER Los objetos o entidades son representados por rectángulos y las relaciones por líneas o rombos que conectan directamente a los objetos. Por ejemplo:


Sin embargo indicar que el objeto X se relaciona con el objeto Y no es suficiente. Se debe comprender la cantidad de ocurrencias en que los objetos X – Y se relacionan, esto se hace aplicando un concepto llamado CARDINALIDAD.

Dos objetos se pueden relacionar con las siguientes cardinalidades:

- Uno a uno
- Uno a muchos
- Muchos a muchos

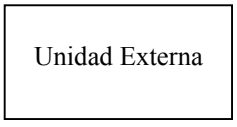
Los DER emplean símbolos que permiten representar la cardinalidad de las relaciones entre los objetos.




## 5.4.2 Modelado Funcional y Flujo de Información

El diagrama de flujo de datos ( DFD ) es una técnica gráfica que representa el flujo de la información y las transformaciones que se aplican a los datos al moverse desde la entrada a la salida.

La notación básica para construir DFD's es la siguiente:



Un rectangulo representa a un elemento del sistema ( por ejemplo un hardware, una persona, un programa ) o un sistema que produce o recibe información que es transformada por el software.

Proceso

Un circulo representa un proceso o transformación que se aplica a los datos y los cambia de alguna forma.

Elemento de Datos.

La flecha representa a un elemento o una colección de elementos de datos.

Almacén de

Representa información almacenada y que utiliza el software.

Los DFD's pueden representar un sistema a cualquier nivel de detalle funcional. Un DFD de nivel cero es también denominado *diagrama de contexto,* representa al elemento de software o al sistema como una sola burbuja o proceso de transformación con datos de entrada y salida representados con flechas.

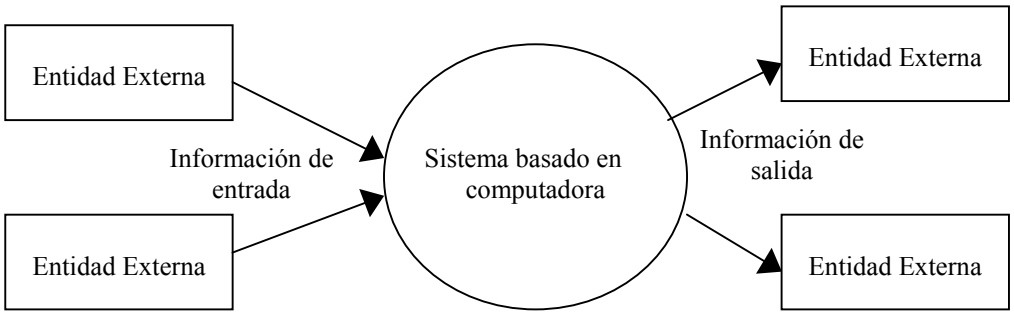



FIG- Modelo de Flujo de Información a Nivel 0 o de Contexto

A partir del DFD de nivel 0 si se quieren representar mas detalles se crearán entonces diagramas de nivel 1, 2 etc. según la profundidad de los detalles.

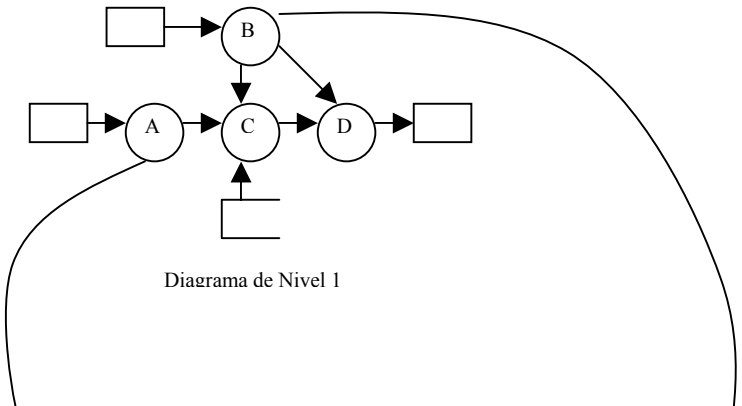



Diagrama de Nivel 1

El DFD es una herramienta gráfica muy valiosa para el análisis de requisitos del software. Sin embargo muchas veces el analista confunde los propósitos del diagrama, al elaborarlo e interpretar su función como un diagrama de flujo ( los diagramas utilizados para representar un algoritmo )

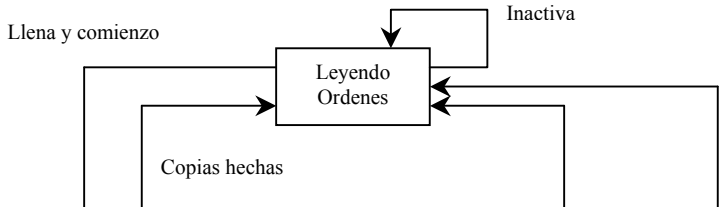
Un DFD representa el flujo de la información sin representar explícitamente la lógica de procesamiento, es decir, el DFD no representa condiciones, bucles, tiempo de ocurrencia de los procesos, etc. Los DFD no representan algoritmos, más bien cada burbuja de un DFD representa una función del sistema y esa función puede posteriormente detallarse en una especificación que derive en un algoritmo y a su vez en un programa.

### 5.4.3 Modelado del Comportamiento

El diagrama de transición de estados DTE representa el comportamiento de un sistema que muestra los estados y los sucesos que hacen que el sistema cambie de estado.

Un estado es un modo observable de comportamiento, por ejemplo monitoreando, comprobando, calculando, etc. Cada estado representa un modo de comportamiento y el DTE indica cómo se mueve el sistema de un estado a otro.

Ejemplo: Un DTE que representa el comportamiento para un software de una máquina fotocopiadora sería el siguiente:



Cada rectángulo representa un estado del sistema. Las flechas representan los sucesos y señalan de que estado a qué estado cambia el sistema cuando ocurre dicho suceso. Las flechas se etiquetan con el nombre del suceso.

### 5.4.2 Diccionario de Datos

La notación básica para crear los DER's, DFD's y DTE's no son suficiente para describir los requisitos del software, por ello el método de Análisis Estructurado se complementa con el *Diccionario de Datos*, en el cual se define el *contenido de la información* o en otras palabras se describen los datos del sistema.

Un Diccionario de Datos es una lista con todos los detalles y descripciones de los elementos incluidos en los Diagramas de Flujo de Datos que describen al sistema.

El diccionario de datos debe contener:

1. Descripción de los almacenamientos de datos

2. Descripción de los procesos
3. Descripción de las estructuras de datos
4. Descripción de los elementos de datos
5. Descripción de los flujos de datos

Para describir con claridad un elemento dato y sus relaciones con otros datos se utiliza la siguiente notación:

| Notación | Significado |
|---|---|
| = | Está compuesto de |
| + | y |
| [ \| ] | o |
| $\{\}^n$ | n repeticiones de |
| ( ) | datos opcionales |
| * * | comentario |

Ejemplo: No-Control = Año-Ingreso + No-Tecnologico + No-Ficha-Admision

Los siguientes ejemplo de formatos pueden ser utilizados para integrar el diccionario de datos:

*Almacenamiento de Datos*

Nombre: Alumnos Inscritos
Descripción: Alumnos que oficialmente forman parte de la matrícula escolar.

Descripción de Datos: - Matrícula - Promedio a la fecha
- Nombre completo - Carrera
- Domicilio
- Semestre que cursa

Volumen: 850 alumnos inscritos actualmente
Crecimiento 180 registros cada semestre

*Procesos*

Nombre: Calcular Promedio General
Descripción: Calcula el promedio general del alumno en base a las calificaciones finales obtenidas en la carrera.

Entradas: Kardex de calificaciones

Salidas:

Resum

Nombre: Acta de Calificaciones

Descripción: Documento oficial para que el profesor notifique a la Dirección las calificaciones finales de los alumnos en la materia impartida.

Contenido: Acta_Calific = { Matricula + Nombre + Calificación }$^n$

Donde 30 <= n <= 45 aprox.

Volumen: Mínimo 1, máximo 3 por materia.

Aprox. 200 actas semestralmente por todas las materias

*Estructura de Datos*

*Elemento Dato*

| | |
|---|---|
| Nombre: | Tipo de Examen |
| Descripción: | Oportunidad de examen en la que se evalúa al alumno para determinar si aprueba la materia. |
| Tipo: | Alfabético |
| Longitud: | 1 |
| Rango de Valores: | Tipo Examen = [ O \| E \| T ] |

Lista de Valores Específicos ( si los hay ):

| | |
|---|---|
| O | Ordinario |
| E | Extraordinario |
| T | Título de Suficiencia |

## 5.5 LA ESPECIFICACION DE REQUISITOS DEL SOFTWARE

La Especificación de Requisitos del Software se produce como culminación de la tarea de análisis. En este documento quedan plasmados los resultados de la tarea de análisis, tales como:

1. Una descripción funcional detallada.
2. Una indicación de los requisitos de rendimiento.
3. Restricciones para el diseño.
4. Criterios de validación.

En muchos casos la Especificación de Requisitos del Software puede ir acompañada de un prototipo ejecutable ( que en algunos casos puede remplazar la Especificación ), un prototipo en papel o un Manual de Usuario Preliminar.

El formato de la Especificación de Requisitos del Software puede tener la estructura siguiente:

Panorama del producto y resumen
Ambientes de desarrollo/operación/mantenimiento
Interfaces externas y flujo de datos
Despliegues al usuario / formato de informes
Resumen de comandos del usuario
Diagrama de flujo de datos de alto nivel
Fuentes y destinos lógicos de datos
Almacenamiento lógico de datos
Diccionario lógico de datos
Especificaciones funcionales
Requisitos de operación
Condiciones de excepción / manejo de excepciones
Subconjuntos iniciales y prioridades de codificación
Modificaciones y mejoras previstas
Criterios de aceptación
Pruebas funcionales y de operación
Estándares de documentación
Guías de diseño
Fuentes de información
Glosario de términos

## 5.6 REVISION DE LA ESPECIFICACION

El desarrollador de software y el cliente realizan una revisión de la Especificación de Requisitos del Software ( y/o del prototipo ). Debido a que la Especificación constituye el fundamento de la fase de desarrollo, se debe tener un cuidado extremo en la realización de la revisión.

Terminada la revisión tanto el cliente como el técnico deben "firmar" la especificación, este será un "contrato" para el desarrollo del software.

El cliente debe tener en cuenta que cada cambio posterior será una ampliación del software y podrá en consecuencia incrementar el costo y/o retrasar la agenda.

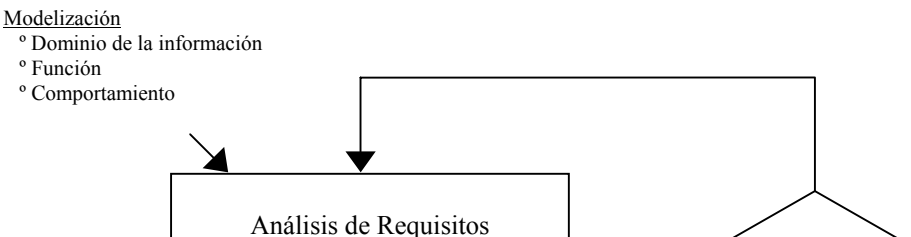


Fig- Fase de Análisis de Requisitos

## 5.7 INGENIERIA DE SOFTWARE ASISTIDA POR COMPUTADORA (CASE)

Llevar a cabo el análisis estructurado en forma manual usando papel, plantillas de símbolos y lápiz puede resultar difícil sobre todo en la modelización de grandes sistemas.

Por esta razón las herramientas CASE son un apoyo imprescindible. Existen un sin numero de programas CASE con múltiples propósitos: hay herramientas CASE para la construcción de prototipos, para el análisis y diseño usando una metodología específica, etc.

Ejemplos de herramientas CASE comerciales:

Herramienta: AxiomSys ( Herramienta CASE para el analisis de sistemas )
Fabricante: STG Inc. ( Inglaterra )

Metodologia: Analisis Estructurado
Plataforma: Windows 3.1, Windows 95 y Unix

Herramienta: Asistente de Desarrollo
Fabricante: Ristanoviz Case
Metodología: Oritentado a Objetos
Plataforma: Win 3.1 y Win 95

La herramienta CASE AxiomSys de Análisis Estructurado permiten que con pulsaciones de ratón y barra de herramientas de dibujos crear rápidamente un DFD. Además permiten ir especificando las descripciones de cada elemento que se incorpora al diagrama, con lo que al final se genera automáticamente el Diccionario de Datos. La herramienta lleva a cabo validaciones automáticas de consistencia entre los DFD de mayor nivel y sus DFD hijos o de detalle.

-oOo-