37
2j.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA DE DIAGNOSTICO PARA EL CONTROL ESTADISTICO DE PROCESOS

T E S I S

Que para obtener el titulo de

INGENIERO EN COMPUTACION

p r e s e n t a

JAIME ENDO SUZUKI

Director de Tesis: Dr. Alejandro Terán Castellanos

México, D. F. 1997

A mi Padre :

Que gracias a su ejemplo de trabajo y disciplina, me ha servido para jamás rendirme, y redoblar esfuerzos ante una adversidad.

A mi Madre†:

Por su amor y su confianza. Que me han servido de impulso para decidir y trabajar en todas mis metas.

Mama, gracias por todo, te dedico este trabajo.

A Tomás :

Que en todo momento a sido más que un hermano, un gran amigo.

A Mika :

Por enseñarme la grandeza de este mundo.

A Mario :

Por enseñarme lo que no se debe de hacer en la vida.

A LA :

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO.

Por darme la oportunidad de formar parte de ella, y recibir una formación profesional, deportiva y espiritual.

A :

Todos los profesores de la Facultad de Ingeniería, que todos formaron parte de mi desarrollo. En especial a :

Ing. Adolfo Millán N., por la confianza que tuvo en mi persona.

Dr. Alejandro Terán C., por permitirme aprender el significado de la sencillez y la calidad en el trabajo.

CONTENIDO GENERAL.

CONTENIDO DE FIGURAS.

CONTENIDO DE TABLAS.

INTRODUCCION

INTRODUCCIÓN.

La elaboración de ésta tesis tiene la finalidad de desarrollar un sistema de cómputo para la detección de faltas de control (**DFC**) considerando un entorno gráfico. Para ello se consideran los siguientes aspectos:

- Ingeniería de programación orientada a objetos.
- Control estadístico de procesos.

En las últimas décadas la ingeniería de programación ha sufrido una gran evolución, como consecuencia del auge de los sistemas de cómputo. Esto dada la aceptación en todas las áreas de la actividad del ser humano, en los cuales se han desarrollado sistemas de cómputo específicos para cada actividad. En los últimos años se han desarrollado una gran cantidad de sistemas de cómputo y progresivamente más complejos, tal es el caso del entorno gráfico, procesamiento distribuido, aplicaciones integradas, etc. Creando la necesidad de desarrollar metodologías capaces de soportar dicha complejidad.

La ingeniería de programación orientada a objetos es un concepto reciente dentro de las tecnologías de desarrollo de sistemas de cómputo. Dicho concepto tiene la capacidad de:

- Soportar con facilidad los sistemas complejos.
- Facilitar la reutilización de código.
- Disminuir el tiempo de desarrollo.
- Asegurar la eficiencia de sistema.
- Facilitar el mantenimiento.

La tecnología de objetos es considerado como un paso importante dentro de la industrialización del software, logrando un proceso de manufactura sistemático y con ello aumento en la productividad y sistemas de gran calidad.

A pesar de la ventajas de la ingeniería orientado a objetos y las múltiples aplicaciones que se han desarrollado (programación, Sistemas operativos, Bases de datos), aún no se ha aplicado dicha tecnología debidamente.

En la actualidad para mantener un nivel competitivo, toda organización industrial o de servicios, necesita el uso de la calidad, para lograr que sus productos o servicios que se

generan sean virtualmente uniformes. Para ello se recurre como metodología fundamental, el control estadístico de procesos, así como su herramienta básica, los diagramas de control.

El texto se desarrolla en cuatro capítulos:

El **CAPITULO I** describe los objetivos y conceptos del control estadístico de procesos. Los diagramas de control, sus componentes, su interpretación y el procedimiento para su desarrollo. Reglas de corrida de AT&T como herramienta para la detección de inestabilidad en un proceso..

El **CAPITULO II** describe los conceptos básicos dentro de la orientación a objetos. Describe la metodología de desarrollo ***OMT***, desarrollado entre otros por Rumbaugh. Describe y desarrolla la fase de análisis para el sistema DFC.

El **CAPITULO III** describe y desarrolla la fase de diseño del sistema DFC bajo la metodología ***OMT***. Describe el procedimiento para obtener un buen modelo.

El **CAPITULO IV** describe la fase de implantación, considerando una programación orientado a objetos y un lenguaje orientado a objetos. Desarrolla la implantación del análisis y diseño del sistema **DFC** en lenguaje de programación C++.

CAPITULO I

CONTROL ESTADISTICO DE PROCESOS

1.1. INTRODUCCIÓN

El uso de la calidad como un arma competitiva impone, sobre las organizaciones industriales o de servicios, la necesidad de controlar la variabilidad de los servicios o productos que ofrecen. Por ejemplo, para una compañía manufacturera, no es suficiente que sus productos cumplan con las especificaciones, sino que debe ofrecerlos al mercado con la menor variación posible entre ellos. Para lograr que los productos o servicios que se generan sean virtualmente uniformes, es indispensable el estudio cuidadoso de las fuentes de variación de su proceso productivo [Taylor2, Wheeler].

El objetivo de este capítulo es presentar una metodología fundamental para el control y el mejoramiento de la calidad: el control estadístico de procesos, así como su herramienta básica, los diagramas de control. Asimismo, en este capítulo se considera el hecho de que, a lo largo del tiempo, la aplicación de diagramas de control ha incorporado el uso de conjuntos de reglas, conocidas como reglas para corridas, con el fin de asegurar un adecuado análisis de las fuentes de variación de un proceso productivo.

El capítulo I se desarrolla en tres secciones fundamentales :

La **SECCIÓN 1.2.** presenta los objetivos y los conceptos básicos del control estadístico de procesos.

La **SECCIÓN 1.3.** describe las generalidades de un diagrama de control. Para el caso específico de los diagramas de control que maneja el sistema **DFC**, se analizan aspectos diversos como: sus componentes, su interpretación, el esquema de muestreo para su implantación, así como el procedimiento para su desarrollo.

La **SECCIÓN 1.4.** presenta al conjunto de reglas para corridas más populares en la industria, conocidas como las Reglas para corridas de AT&T, cuya finalidad es la detección de inestabilidad en un proceso cuando dicha inestabilidad presenta características que la hacen no detectable con el uso de diagramas de control tradicionales.

1.2. EL CONTROL ESTADÍSTICO DE PROCESOS

Por *control estadístico de procesos* (**CEP**) se denota a un conjunto de técnicas y metodologías que utilizan herramientas gráficas y estadísticas para el análisis y el control de la variabilidad de un proceso [Kiemele]. Shewhart, el pionero del CEP, reconoció la inevitabilidad de la variación en un proceso productivo e identificó dos diferentes tipos de variación [Shewhart]:

1.- Controlada o común : Generada por un patrón estable y consistente a lo largo del tiempo.

2.- No controlada : Generada por un patrón que cambia con el tiempo.

Cada tipo de variación reconoce diferentes fuentes: la controlada es atribuible a causas meramente aleatorias o "comunes", mientras que la no controlada es originada por causas "asignables" o "especiales" [Deming, Shewhart]. Para ilustrar lo anterior, considere el proceso de manufactura de una flecha maquinada para la cual se monitorea, a lo largo del tiempo, una característica o variable que determina su calidad (por ejemplo, su diámetro). Si bien no cabe esperar que dos flechas producidas tengan exactamente el mismo diámetro (debido a factores o causas comunes, como la imperfección del material, de la maquinaria, etc.), la variabilidad entre uno y otro de los diámetros observados puede mostrar un patrón de variación consistente en el tiempo. Por tanto, el diámetro de las flechas puede considerarse una variable aleatoria sujeta a una distribución probabilística que no cambia con el tiempo. Ocasionalmente pueden presentarse factores o causas especiales de variación (por ejemplo, desajustes en la maquinaria utilizada, un cambio en el proveedor de la materia prima, etc.) que alteran sensiblemente el patrón de variación previamente identificado.

Un *proceso estable* es un proceso sujeto sólo a variación común y se dice que está bajo control estadístico. En contraposición, un *proceso inestable* es aquél sujeto a variación de ambos tipos (tanto común como no controlada), y se dice que está fuera de control estadístico.

1.3 DIAGRAMAS DE CONTROL PARA VARIABLES

Shewhart (Shewhart) concibió una herramienta gráfica, el *diagrama de control* (*control chart*), que muestra, de manera ordenada según el tiempo en que se observan, un conjunto de datos. La finalidad de un diagrama de control es proporcionar evidencia acerca del estado de control estadístico de un proceso y señalar la ocurrencia de una causa especial de variación, lo que permite realizar las medidas correctivas pertinentes.

La figura 1.1. muestra, esquemáticamente, un diagrama de control. El diagrama incluye dos *límites de control* (el límite superior de control, **LSC** y el límite inferior de control, **LIC**) y una *línea central* (**LC**), que son parámetros que se determinan estadísticamente. Un conjunto de datos observados se considera generado por un proceso estable si todos sus puntos, graficados en un diagrama de control, se encuentran dentro de los límites de control. Por el contrario, se detecta la ocurrencia de una causa especial de variación cuando un punto de los datos observados cae fuera de los límites de control. Las conclusiones a las que permite llegar un diagrama de control se muestran en la tabla 1.1.

FIG. 1.1. COMPONENTES DE UN DIAGRAMA DE CONTROL TRADICIONAL

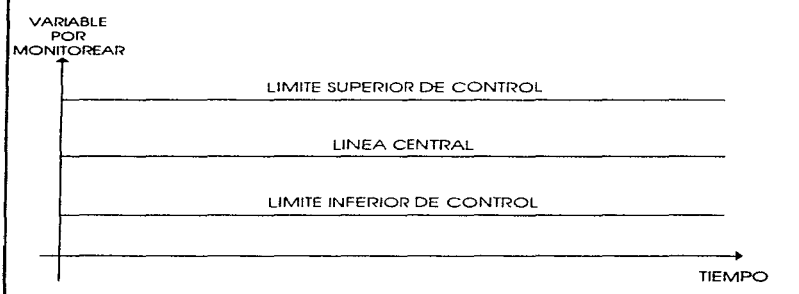


TABLA 1.1. CONCLUSIONES DERIVADAS DE UN DIAGRAMA DE CONTROL

TODO EL CONJUNTO DE OBSERVACIONES DE LA VARIABLE POR MONITOREAR DENTRO DE LOS LIMITES DE CONTROL: EVIDENCIA DE QUE EL PROCESO ES *ESTABLE*, ES DECIR, ESTA BAJO CONTROL. HAY QUE "DEJAR EL PROCESO EN PAZ".

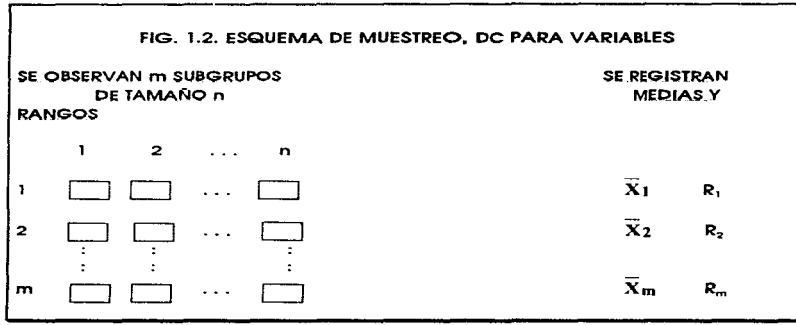
AL MENOS UNA DE LAS OBSERVACIONES DE LA VARIABLE POR MONITOREAR FUERA DE ALGUNO DE LOS LIMITES DE CONTROL: EVIDENCIA DE QUE EL PROCESO ES *INESTABLE*, ES DECIR, ESTA FUERA DE CONTROL. ES NECESARIO "AJUSTAR" EL PROCESO.

El monitoreo de un proceso se realiza a través de un esquema de muestreo en el cual se observan, en diferentes tiempos, **m** muestras o *subgrupos*. La práctica más común consiste en que:

1.- Los subgrupos se observan a intervalos fijos de tiempo (por ejemplo, cada hora)
2.- Todos los subgrupos tienen el mismo tamaño, **n**.

El sistema desarrollado en esta investigación se concentra en *diagramas de control para variables*, que permiten el monitoreo de una característica de calidad medible. El patrón de variación de las mediciones realizadas sobre un proceso se expresa en términos tanto de su dispersión (variabilidad), como de su localización. Por lo tanto, el monitoreo de un proceso a través de diagramas de control para variables requiere la preparación y el análisis simultáneo de un diagrama para la dispersión (variabilidad de una unidad de producto a otra) y otro para la dispersión (comúnmente, el promedio) del proceso. La pareja de diagramas comúnmente utilizadas son el *diagrama* $\mathbf{R}$ y el *diagrama* $\bar{\mathbf{X}}$. El primer diagrama monitorea la dispersión del proceso, a través de los rangos observados en cada uno de los subgrupos. El segundo diagrama monitorea la localización del proceso, mediante los promedios de los subgrupos. La figura 1.2 muestra, de manera esquemática, la manera en la que se realiza el esquema de muestreo para el monitoreo simultáneo de dispersión y localización.

FIG. 1.2. ESQUEMA DE MUESTREO, DC PARA VARIABLES



La práctica común para la adopción de diagramas de control consiste en:

1.- Suponer que las mediciones sigue una distribución normal.

2.- La adopción de la convención norteamericana conocida como *límites 3-sigma.*

Bajo esta práctica, los parámetros de un diagrama de control para el rango, **R**, se determinan de la siguiente manera (Grant) :

$$LC = \bar{R} \qquad (1a)$$

$$LSC = D_4\bar{R} \qquad (1b)$$

$$LIC = D_3\bar{R} \qquad (1c)$$

Por su parte, los parámetros de un diagrama de control para $\bar{\bar{X}}$ se determinan como:

$$LC = \bar{\bar{X}} \qquad (2a)$$

$$LSC = \bar{\bar{X}} + 3\hat{\sigma}_{\bar{X}} = \bar{\bar{X}} + \frac{3\bar{R}}{d_2\sqrt{n}} \qquad (2b)$$

$$LIC = \bar{\bar{X}} - 3\hat{\sigma}_{\bar{X}} = \bar{\bar{X}} - \frac{3\bar{R}}{d_2\sqrt{n}} \qquad (2c)$$

En donde $\bar{R}$ y $\bar{\bar{X}}$ representan el promedio aritmético de los rangos y las medias de los m subgrupos, respectivamente. Las constantes D_4, D_3, y d_2 se encuentran tabuladas en los textos clásicos de CEP (Grant) y sus valores se muestran en la tabla 1.2.

TABLA 1. 2. FACTORES UTILIZADOS PARA CALCULOS

TAMAÑO SUBGRUPO, n	D_3	D_4	A_2	d_2
2	0	3.27	1.88	1.128
3	0	2.57	1.02	1.693
4	0	2.28	.73	2.059
5	0	2.11	.58	2.326
6	0	2.00	.48	2.534
7	.08	1.92	.42	2.704
8	.14	1.86	.37	2.847
9	.18	1.82	.34	2.970
10	.22	1.78	.31	3.078

Tomada de: [Grant]

El procedimiento para desarrollar y aplicar los diagramas de control para variables ($\bar{R}$ y $\bar{\bar{X}}$) se muestra en la tabla 1.3 .

Un proceso se considera inestable si al menos uno de los **m** rangos o una de las **m** medias es mayor que el respectivo **LSC** o menor que el correspondiente **LIC**. El uso simultáneo de diagramas para la dispersión y para la localización permite, entonces, detectar la inestabilidad de un proceso bajo diferentes variantes: cambio sólo en su dispersión, sólo en su localización, o en ambas.

TABLA 1.3. DESARROLLO Y APLICACIÓN DE DIAGRAMAS R Y $\bar{X}$

1.- DETERMINE EL TAMAÑO DE LOS SUBGRUPOS (USUALMENTE, **n** = 2 A 5). TRATE DE QUE LAS MEDICIONES SE REALICEN BAJO CONDICIONES IDENTICAS.

2.- DETERMINE EL TIEMPO ENTRE SUBGRUPOS, DE ACUERDO CON: POSIBILIDAD DE DETECTAR CAMBIOS EN EL PROCESO, COSTO DEL MUESTREO.

3.- REALICE Y REGISTRE LAS MEDICIONES DE CADA SUBGRUPO.

4.- PARA CADA SUBGRUPO, **j**, CALCULE:

$$\bar{X}_j = \sum_{i=1}^{n} \frac{X_i}{n} \qquad R_j = X_{máx} - X_{mín}$$

5.- OBSERVE 20-30 SUBGRUPOS (NO RECOMENDABLE MENOS DE 10).

6.- A PARTIR DE TODOS LOS **m** SUBGRUPOS, DETERMINE:

$$\bar{\bar{X}} = \sum_{j=1}^{m} \frac{\bar{X}_j}{m} \qquad \bar{R} = \sum_{j=1}^{m} \frac{R_j}{m}$$

7.- ELABORE LOS DIAGRAMAS DE CONTROL **R** Y $\bar{X}$.

8.- SI EL DIAGRAMA **R** ESTA BAJO CONTROL, ANALICE DIAGRAMA $\bar{X}$.

9.- DEL ANALISIS DE AMBOS DIAGRAMAS:

- SI EL PROCESO ESTA BAJO CONTROL, UTILICE LOS LIMITES DE CONTROL PARA NUEVAS OBSERVACIONES.
- SI EL PROCESO NO ESTA BAJO CONTROL: (1) DETECTE Y CORRIJA LOS PROBLEMAS, (2) ELIMINE LOS DATOS ASOCIADOS A PUNTOS FUERA DE CONTROL Y (3) RECALCULE LOS LIMITES DE CONTROL.

1.4 REGLAS PARA CORRIDAS DE AT&T

El esquema presentado en la sección anterior, al que denominaremos *esquema tradicional*, sólo permite detectar cambios drásticos en la dispersión o la localización de un proceso. Se han desarrollado diferentes conjuntos de reglas para detectar otros tipos de inestabilidad en un proceso (por ejemplo, cambios leves, tendencias, patrones sistemáticos, etc.), conocidas como *reglas para corridas* (*runs rules*). La presentación, uso y análisis de algunos conjuntos de reglas se pueden encontrar en (Hoyer, Nelson, Western).

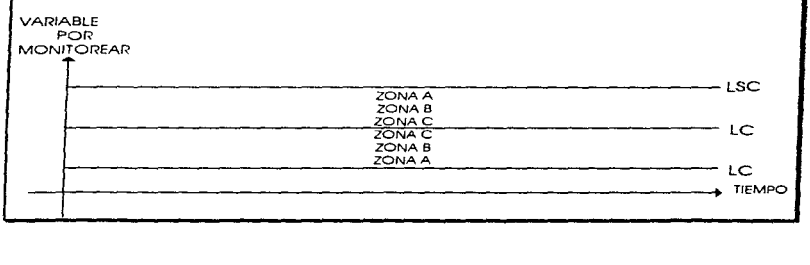
Las características comunes de todas las reglas para corridas son:

1.- Dividen la región entre un límite de control y la línea central de un diagrama en diferentes zonas (comúnmente en tres zonas con la misma área, como se muestra en la figura 3).

2.- La activación de una regla (señal de inestabilidad en un proceso) se basa en el análisis de una *corrida* (conjunto de **r** puntos sucesivos del diagrama), de tal manera que si **k** puntos se ubican en una cierta zona, se considera que el **k**-ésimo punto activa a la regla.

La figura 1.3 presenta las diferentes zonas de aplicación de una regla, en relación con los diferentes componentes de un diagrama de control.

FIG. 1.3. ZONAS DE APLICACIÓN DE UNA REGLA DE CORRIDA



El conjunto de reglas para corridas más usuales son las *reglas para corridas de AT&T*, que se han convertido en un estándar en la práctica del CEP. Su finalidad es detectar cambios de diferente magnitud en un proceso. Las reglas que lo conforman son [Western]:

Regla 1: se activa cuando un punto cae fuera de los límites de control. Esta regla es equivalente al esquema tradicional propuesto por Shewhart.

Regla 2: se activa cuando dos de tres puntos sucesivos caen en la zona **A** (de un mismo lado respecto a **LC**) o más lejos. Permite la detección de cambios moderados en un proceso.

Regla 3: se activa cuando cuatro de cinco puntos sucesivos caen en la zona **B** (de un mismo lado respecto a **LC**) o más lejos. Esta regla permite detectar cambios leves en un proceso.

Regla 4: se activa cuando ocho puntos sucesivos caen en la zona **C** (del mismo lado respecto a **LC**) o más lejos. Permite detectar cambios pequeños en un proceso.

La tabla 1.4 muestra las conclusiones que pueden derivarse de la aplicación de reglas para corridas para un proceso en el cual se monitorean dispersión y localización.

TABLA 1.4. CONCLUSIONES DERIVADAS DEL USO DE REGLAS PARA CORRIDAS

NINGUNA OBSERVACIÓN DE LA VARIABLE POR MONITOREAR ACTIVA ALGUNA REGLA PARA CORRIDA: EVIDENCIA DE QUE EL PROCESO ES ***ESTABLE***, ES DECIR, ESTA BAJO CONTROL. HAY QUE "DEJAR EL PROCESO EN PAZ".

AL MENOS UNA DE LAS OBSERVACIONES DE LA VARIABLE POR MONITOREAR ACTIVA POR LO MENOS UNA DE LAS REGLAS PARA CORRIDAS: EVIDENCIA DE QUE EL PROCESO ES ***INESTABLE***, ES DECIR, ESTA FUERA DE CONTROL. ES NECESARIO "AJUSTAR" EL PROCESO.

Es conveniente señalar que si bien existe una amplia gama de software en el mercado que incluye, entre sus capacidades, el control estadístico de procesos (una lista reciente de compañías que ofrecen este tipo de software puede encontrarse en [Struebing]), estos suelen ser caros (el rango de costos van desde poco menos de cien hasta varios miles de dólares) o no incluir reglas para corridas.

Por tal razón, la investigación que dio lugar a este documento tuvo como finalidad el desarrollo de un sistema de cómputo que fuera accesible para la industria pequeña y mediana mexicana, que incluyera las reglas para corridas de AT&T y que, posteriormente, fuera fácilmente expandible para incorporar otros conjuntos de reglas.

1.5. RESUMEN DE CAPITULO.

En este capítulo se presenta una metodología fundamental para el control y mejoramiento de la calidad : el control estadístico de procesos, así como su herramienta básica, los diagramas de control. Se describen los objetivos y los conceptos básicos del control estadístico de procesos, así como, los componentes, la interpretación, el esquema de muestreo para la implantación, y el procedimiento para el desarrollo de los diagramas de control.

Presenta además, el conjunto de reglas para corrida de AT&T, cuya finalidad es la detección de inestabilidad de un proceso no detectable con los diagramas de control tradicionales.

Con el desarrollo del capítulo I, se tiene una idea clara, de la finalidad del sistema de detección de falta de control **(DFC)**, y se retoma para realizar el análisis y diseño en los capítulos II y III.

CAPITULO II

ANÁLISIS

2.1. INTRODUCCIÓN.

La orientación a objetos es un concepto o tecnología que ha surgido en los últimos años y que ha cobrado un gran auge. A partir de la tecnología orientado a objetos se logra (Taylor) :

- Un desarrollo más rápido basado en la reutilización de subprogramas y aplicaciones completas, así como en la reutilización de prototipos.
- Una mejor calidad del sistema, proporcionado por la herencia[1] , teniendo la facilidad de crear nuevos subprogramas evitando la adición de nuevos problemas. Con ello los subprogramas ya creados no pierden su rendimiento.
- Un mantenimiento más sencillo proporcionado por el encapsulado[2] con lo cual se logra aislar los problemas de manera tal que facilita determinar alguna falla.
- Una considerable reducción en los costos basado en las características antes descritas.
- Mayor flexibilidad, basada en la escalabilidad (construir sobre lo existente sin modificarlo) y la adaptabilidad (capacidad de cumplir con requerimientos).

Para el desarrollo orientado a objetos existen diversas metodologías para el análisis y diseño (Anexo 1) cada uno de ellos con características propias.

Para el desarrollo del sistema de detección de falta de control **(DFC)**, se utilizará la metodología **OMT** (*Object Modeling Technique*) propuesta entre otros, por **Rumbaugh**. Esta es una metodología estática en donde, se enfatiza más en los datos que en las funciones. Este enfoque proporciona una base más sólida para el desarrollo de sistemas en la que los objetos y sus relaciones son susceptibles a modificaciones. La metodología **OMT** utiliza diagramas simples y descriptivos.

La metodología **OMT** consiste en construir un modelo de la aplicación basado en la creación de los objetos que intervienen en

[1] Herencia : Es un concepto de la tecnología orientado a objetos, basado en una relación jerárquica de los subprogramas. Esto es, un subprograma de nivel inferior contiene todas las características de un subprograma superior.

[2] Encapsulado : Es el proceso de ocultamiento para el resto del sistema, de todos los detalles de un subprograma.

el sistema y posteriormente, durante el diseño, se le agregan los detalles de la implantación. La metodología consta de las siguientes etapas :

1.- **ANÁLISIS** : La primera etapa consta de un enunciado, expresando el problema a resolver. Durante el análisis se construye un modelo de la situación del mundo real, presentando las propiedades importantes. Los objetos[3] en el modelo deben contener los conceptos del dominio de la aplicación y no los conceptos de la implementación. Un buen modelo es comprendido por expertos de la aplicación. El modelo de análisis no posee decisiones de implantación.

2.- **DISEÑO DEL SISTEMA** : Durante el diseño del sistema se realizan decisiones de alto nivel acerca de la arquitectura. En el diseño de sistema se deciden las características de desarrollo para su optimización, se eligen las estrategias para atacar el problema.

3.- **DISEÑO DEL OBJETO** : Durante el diseño del objeto se construye un modelo basado en el análisis pero agregando los detalles de la implantación. En el diseño del objeto se incorporan las estrategias establecidas durante el diseño del sistema. El punto principal del diseño del objeto es la estructuración de los datos y los algoritmos necesarios para la implantación de cada clase[4].

4.- **IMPLANTACIÓN** : Las clases y las relaciones desarrolladas durante el diseño del objeto son traducidos a un lenguaje de programación. Para llevar a cabo la implantación, es necesario determinar el lenguaje de programación que se utilizará en dicha etapa. La programación es la parte mecánica del ciclo de desarrollo, ya que todas las decisiones se tomaron durante el diseño.

En el capitulo II se describe y se desarrolla la primera etapa de la metodología seleccionada (ANÁLISIS), para el desarrollo del sistema **DFC,** considerando los conceptos y necesidades descritas en el capitulo I.

---

[3] Objeto : Abstracción de datos para el manejo del problema (sección 2.3.1.)

[4] Clase : Grupo de objetos con caracteristicas similares.

El objetivo del análisis es la construcción de un modelo preciso y comprensible del mundo real. Para ello es necesario considerar los requerimientos del sistema y su relación con el mundo real. En esta etapa se considera lo **QUE** va a realizar el sistema sin importar **COMO** lo va a realizar, ni la manera en que se implante el sistema. Como resultado del análisis se obtienen modelos precisos que permiten la generación de soluciones a problemas dados.

La fase de análisis no sigue una secuencia lógica, de hecho no es un proceso mecánico. Los modelos a desarrollar dependen en gran medida de la perspectiva del diseñador, esto es, pueden existir varios modelos diferentes para un mismo problema.

En la fase de análisis se consideran varias etapas para su desarrollo y son las siguientes :

1.- Enunciado del problema.
2.- Modelo de objeto.
3.- Modelo dinámico.
4.- Modelo funcional.

La figura 2.1. muestra las etapas y fases que conforman la metodología **OMT**.

El capitulo II se desarrolla en cuatro secciones fundamentales :

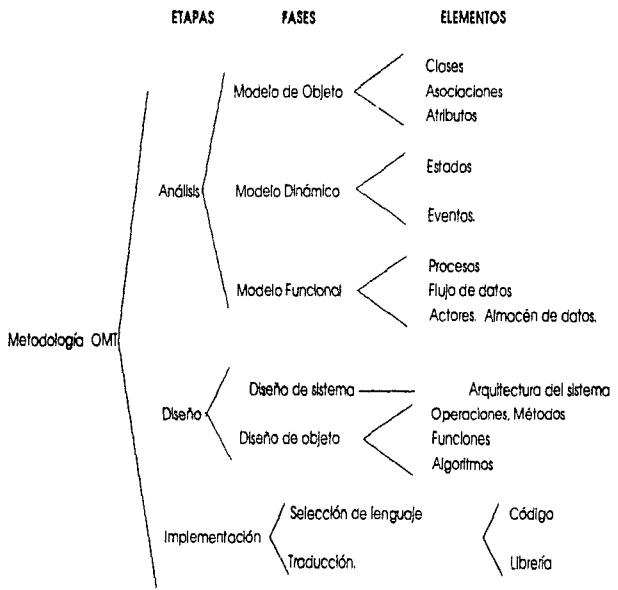
La **SECCIÓN 2.2** desarrolla el enunciado del problema, como primer paso para el desarrollo del sistema.

La **SECCIÓN 2.3.** describe y desarrolla el modelado de objeto.

La **SECCIÓN 2.4.** describe y desarrolla el modelado dinámico.

La **SECCIÓN 2.5.** describe y desarrolla el modelado funcional.

FIGURA 2.1. DIAGRAMA ESQUEMATICO DE LAS ETAPAS DE LA METODOLOGIA OMT.



2.2. ENUNCIADO DEL PROBLEMA.

El enunciado del problema es el primer paso de la fase de análisis, y consta de un enunciado que describe el problema y la necesidades del sistema, puede ser ambiguo, incompleto e incluso inconsistente, ya que en los modelos posteriores se define con mayor claridad el problema y sus soluciones.

Para el sistema **DFC**, el enunciado del problema se describe de la siguiente forma :

Elaboración de un paquete de diagnóstico de procesos productivos que permite verificar su estabilidad y, en su defecto, efectuar la detección de falta de control estadístico. Esto mediante una interfaz gráfica amigable para el usuario.

El sistema debe permitir :

AL USUARIO :

- **Introducir los datos en subgrupos. (En lote o interactivo)**
- **Introducir las características de los datos (Número de subgrupos y tamaño de subgrupos)**

Una vez realizado el procesamiento de la información :

EL SISTEMA :

- **Desplegar las gráficas de control.**
- **Señalar subgrupos fuera de control.**
- **Obtener conclusiones preliminares a partir de los diagramas de control.**
- **Indicar el estado del proceso : estable (bajo control) o inestable (falta de control).**

Con el enunciado del problema, se comienza a identificar las características necesarias del sistema.

2.3. MODELADO DEL OBJETO.

El modelaao del objeto es el primer paso para el análisis de los requerimientos del problema y se basa en el enunciado del problema y del conocimiento del entorno en el cuál se aplica el sistema.

El modelado del objeto se enfoca a la estructura estática del sistema, los objetos y las clases, es por ello que el modelo de objeto antecede al modelo dinámico y al modelo funcional, ya que una estructura estática es más fácil de analizar, comprender y comunicar.

Para el desarrollo adecuado de un modelo de objeto correcto se recomiendan los siguientes pasos : (Rumbaugh)

1.- Identificación de las clases.
2.- Preparación de un diccionario de datos.
3.- Identificación de las asociaciones entre objetos.
4.- Identificación de los atributos.
5.- Organización y simplificación de las clases utilizando herencia.
6.- Refinamiento del modelo.
7.- Formato de interfaz.

2.3.1. IDENTIFICACION DE LAS CLASES.

El primer paso para la construcción del modelo del objeto es la identificación de las clases.

Una **clase** se forma por *un grupo de objetos con propiedades o atributos similares, conductas u operaciones y semántica común con otros objetos.*(Rumbaugh). Un **objeto** es : *Un concepto o una abstracción de datos para el manejo del problema.* Por ejemplo : para el sistema **DFC**, datos, tablas, diagrama serían algunos objetos. Un objeto dentro de la programación se describe como: *Un empaque de código que contiene un estado y una colección de datos y operaciones.* Los objetos se forman por tres componentes principales que son : (Fig. 2.2.) (Booch).

- **ESTADO** : El estado de un objeto reúne todas las propiedades estáticas del objeto, por ejemplo : El tipo de valor correspondiente a cada una de las propiedades. Las

propiedades son características distintivas que contribuyen a identificar o especificar un objeto.

- **COMPORTAMIENTO** : El comportamiento define la forma en que actúan los objetos. Se define en términos de sus cambios de estado y de mensajes generados. Descrito de otra forma, el comportamiento es definido por sus acciones. Dentro de la programación orientada a objetos, las operaciones se definen como métodos y forman parte de la declaración del objeto.
- **IDENTIDAD** : La identidad es el medio por la cual los datos se agrupan en entidades distinguibles y cuantificables. Dichas entidades forman un objeto. Por medio de la identidad se logra que la referencia a un objeto sea uniforme e independiente, permitiendo una mezcla de objetos creados. Dicho de otra forma la identidad es la propiedad de un objeto que lo identifica y lo distingue de otros objetos.

Las clases y objetos se identifican como sustantivos en la descripción del problema.

Para el sistema **DFC** se identifican las siguientes clases :

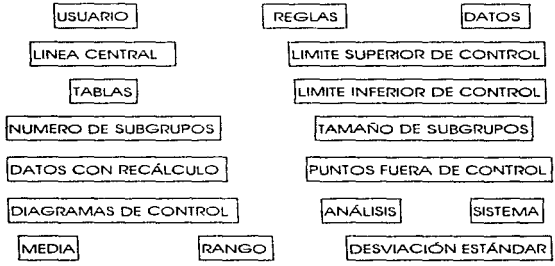
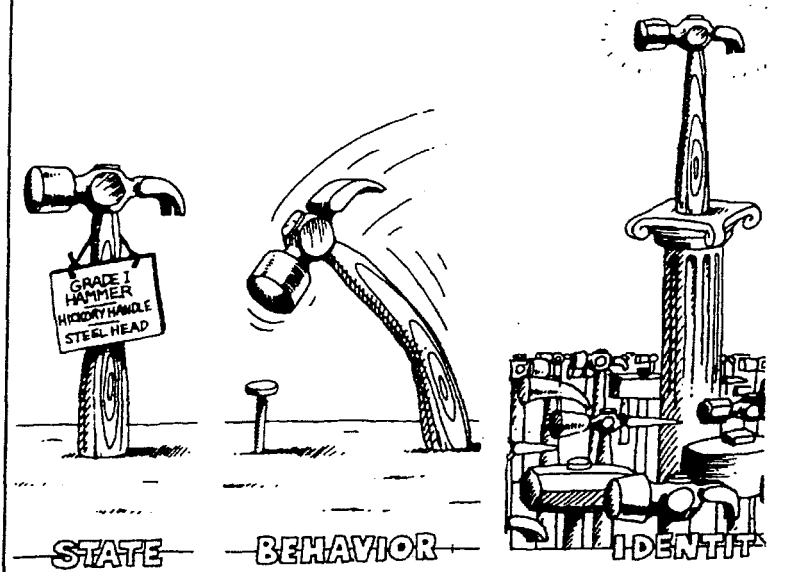


Figura 2.2. COMPONENTES DEL OBJETO (Booch)



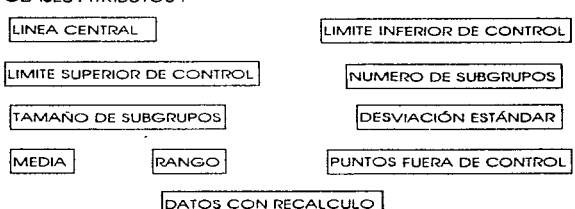
Después de considerar las clases relacionadas, se prosigue con la selección de las clases adecuadas, de acuerdo con los siguientes criterios :(Rumbaugh).

- Clases Redundantes : Si existen dos clases que contienen la misma información, se selecciona aquélla que tenga el nombre más descriptivo para el entorno del problema.
- Clases irrelevantes : Si existe alguna clase que su participación es irrelevante para el problema, deberá ser eliminada.
- Clases vagas : Aquellas clases que su definición o participación no sea específica, deberá ser eliminada.
- Atributos : Aquellas clases que se tomaron en consideración, pero que no poseen una individualidad definida, deberán formar parte de otras clases.
- Operaciones : Si un nombre describe una operación aplicable a un objeto, y no puede ser manipulada por sí misma, no forma parte de un objeto.
- Elementos de implantación : Aquellas clases que hacen referencia a la implementación deberán ser eliminadas.

Con base en los criterios descritos anteriormente, se eliminarían las siguientes clases identificadas :

CLASES REDUNDANTES :

CLASES ATRIBUTOS :



CLASES VAGAS :

TABLAS SISTEMA

OPERACIONES :

ANALISIS

Con la eliminación de clases realizada, se identifican tres objetos clave para el desarrollo del sistema **DFC**. Dichas clases son las siguientes:

CLASES PARA EL DESARROLLO DEL SISTEMA DFC

2.3.2. PREPARARACION DEL DICCIONARIO DE DATOS.

Después de definir las clases, se construye un diccionario de datos que contiene la información de cada entidad. Esta descripción deberá ser lo más completa posible. Puede contener el medio en que se desenvuelve, restricciones, alcance, asociaciones, atributos y operaciones.

Para las clases obtenidas en el desarrollo del sistema **DFC**, se describe el siguiente diccionario de datos :

- **DATOS** : Contiene todos los datos obtenidos y organizados en subgrupos. Tiene métodos para recibir los datos en forma interactiva o en lote y para calcular los puntos de las gráficas de control.
- **DIAGRAMACONTROL** : Contiene todos lo elementos de la gráfica, como son el límite central, límite superior de control, límite inferior de control, y los puntos que conforman la gráfica. Así como los métodos para la obtención de dichos elementos.

- **REGLAS** : Contiene principalmente las reglas de corrida para detectar puntos fuera de control. Contiene como atributos las variables necesarias para dicho proceso, como son : AreaSup, AreaInf, PuntosCons, PuntosArea, PuntosFuera y los métodos necesarios para obtener dichos atributos.

2.3.3. IDENTIFICACION DE LOS ATRIBUTOS.

El siguiente paso es la identificación de los atributos de cada clase.[5]

Los atributos son los datos o valores que poseen los objetos del mundo real. Cada atributo de una clase posee un valor para cada instancia de esa clase, estos valores pueden ser iguales o distintos. Cada nombre de atributo es único dentro de una clase. Un atributo es una valor, más no es un objeto.

Los atributos forman parte de la estructura básica del problema y tendrán que ser identificados tomando en cuenta los requerimientos del problema y su relación con el mundo real. Los nombres seleccionados para los atributos deberán ser claros y específicos.

Para obtener los atributos correctos es conveniente considerar el siguiente conjunto de criterios : (Rumbaugh).

- Objetos : Si un atributo es de gran importancia como entidad independiente, el atributo se considera como objeto y se manejará como tal.
- Asociación : Si el valor de un atributo varía dependiendo de la aplicación en particular, este valor o atributo formará parte de una asociación.
- Identificadores : En el proceso de implantación se utilizan identificadores para la referencia específica a un objeto. Estos identificadores no forman parte de los atributos durante la fase de análisis.

---

[5] Bajo el criterio desarrollado por Rumbaugh, la identificación de atributos se desarrolla después de la identificación de las asociaciones. Pero se invirtió el orden por considerar que las asociaciones son más fáciles de identificar una vez descrita las clase.

- Atributos de asociación : Si un atributo depende de la presencia de alguna asociación, este atributo formará parte de los atributos de asociación.
- Valores internos : Los valores internos deben ser eliminados durante la fase de análisis. Los valores internos son aquellos que describen el comportamiento interno del objeto y no pueden ser vistos desde el exterior.
- Atributos específicos : Aquellos atributos que generan o necesitan de una gran cantidad de operaciones deberán ser omitidas.
- Atributos incoherentes : Aquellos atributos que son completamente incoherentes con respecto a la clase, deberán ser eliminados o integradas a otra clase.

Para las clases seleccionadas para el sistema **DFC** se identifican los siguientes atributos :

DATOS :

- Datos. (Contiene los datos por analizar, agrupados en subgrupos).
- NoSubGpos. (Cantidad de subgrupos de los datos por analizar)
- TamSubGpo. (Cantidad de elementos que contiene cada subgrupo).
- X (Arreglo con los valores de la media de cada subgrupo).
- R (Arreglo con los valores de rango de cada subgrupo).

DIAGRAMACONTROL :

- LC (Valor del límite central del diagrama de control)
- LIC (Valor del límite interior de control del diagrama de control).
- LSC (Valor del límite superior de control del diagrama de control).
- PuntosGra : (Valores para desplegar en la gráfica de control).

REGLA :

- AreaSup (Especifica el área superior al límite central de control, en donde se aplica la regla para corridas correspondiente).
- AreaInf (Especifica el área inferior al límite central de control, en donde se aplica la regla para corridas correspondiente).
- Puntoscons (Cantidad de elementos por considerar para aplicar una determinada regla para corridas).
- PuntosArea (Cantidad de elementos por considerar para detectar el rompimiento de una regla y considerarla como un punto fuera de control estadístico)
- PuntosFuera (Elementos o subgrupos que rompen con la regla para corridas y que se considera como fuera de control estadístico).

2.3.4. IDENTIFICACION DE LAS ASOCIACIONES.

Una asociación es toda relación existente entre las clases, e incluso, cuando una clase hace referencia a otra, se genera una asociación. Las asociaciones son bidireccionales, a pesar que los nombres sean en una sola dirección. Es importante considerar el nombre que se le asigna a cada asociación, ésta deberá ser clara, simple y explícita. Las asociaciones se describen como verbos en la descripción del problema. Por ejemplo : Genera, Proporciona, etc.

Existen cuatro conceptos relacionados con las asociaciones, que son de son de suma importancia considerar : (Rumbaugh).

1.- **MULTIPLICIDAD** : La multiplicidad especifica la cantidad de instancias de cierta clase que pueden estar relacionadas con una instancia de otra clase asociada. La multiplicidad podría ser : una, muchas e incluso infinitas, pero nunca podrán tomar un valor negativo. Para indicar la multiplicidad, se considera una simbología especial al final de la trayectoria o línea de asociación. Para el caso de la metodología **OMT**(Rumbaugh) un círculo sólido representa una asociación de cero o más, un círculo hueco representa una asociación de cero o uno, la

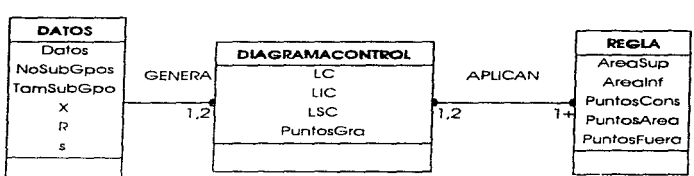
trayectoria sin símbolo representa una asociación de uno a uno. En general, la multiplicidad numérica se indica al final de la trayectoria.

2.- **CLASE ASOCIADA** : De manera similar a como las clases poseen atributos, las asociaciones también poseen sus atributos. Los atributos de las asociaciones son útiles para aumentar la flexibilidad si la multiplicidad de la asociación llega a modificarse. Los atributos de asociación de una clase asociada no pueden formar parte de los atributos de clase.

3.- **AGREGACIÓN** : La agregación es un tipo de asociación especial. La agregación es una relación " **Parte de** ", que representa los objetos componentes que forman parte de un objeto mayor. La agregación se representa mediante un rombo en el objeto mayor de la trayectoria de asociación.

4.- **GENERALIZACIÓN** : La generalización es la relación entre una clase y sus clases derivadas. Una clase derivada es aquella que genera los atributos y operaciones de su clase padre (*herencia*). A las clases derivadas se les denomina subclase, y a las clases padre se les denomina superclase. La generalización se define como una relación " **Tipo de** ", por ello, todos los atributos y operaciones definidas en una superclase pueden ser implementadas en sus subclases. La generalización es de suma importancia, ya que a partir de ésta se generan los códigos reutilizables. La generalización se representa mediante un triángulo dentro de la trayectoria de superclase a subclases.

Para el sistema **DFC** se identifican las siguientes asociaciones de las clases seleccionadas :



Al diagrama anterior, que agrupa a las clases y sus asociaciones se le denomina diagrama de objetos, y es el diagrama principal del modelado del objeto. La representación de la clase en el diagrama de objetos contiene tres secciones, cada uno de los cuales contiene : nombre de la clase, nombre (s) del atributo y nombre (s) del método.

Como se observa en el diagrama del objeto del sistema **DFC**, éste no contiene operaciones, ya que estos se identificarán en el transcurso del modelado dinámico y modelado funcional.

2.3.5. ORGANIZACION Y SIMPLIFICACION UTILIZANDO HERENCIA.

El siguiente paso es organizar las clases utilizando herencia (Generalización). Esta organización puede ser agrupando las clases con atributos u operaciones similares generando una superclase o refinando las clases existentes, en clases más específicas generando subclases.

Debido a sus características, el sistema **DFC** no requiere una simplificación utilizando herencia, pero este proceso es importante para el desarrollo de cualquier otro sistema.

2.3.6. REFINAR EL MODELO.

Es difícil que en un solo recorrido se obtenga un adecuado modelo del objeto. Para ello, es necesario revisar los objetos, sus atributos y asociaciones. Que todos ellos sean claros concisos, claros y sin redundancia. Muchos aspectos del modelo de objeto se van refinando conforme se avanza en el modelo dinámico y el modelo funcional.

2.3.7. FORMATO DE INTERFAZ.[6]

Toda aplicación puede dividirse en dos partes :

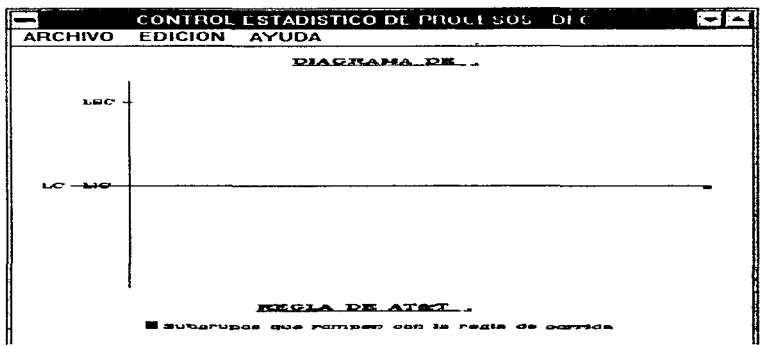
1.- Procedimientos lógicos.

2.- Formato para interactuar con el mundo exterior. (Interfaz).

Para el desarrollo del formato de interface no es necesario ser muy específico, pero es de gran utilidad para no ignorar los elementos que intervienen en el control lógico.

Las fig. 2.3. muestran el formato de interfaz inicial para el sistema **DFC**.

FIGURA 2.3. FORMATO DE INTERFAZ INICIAL.



[6] En la metodología **OMT**, el formato de interface forma parte del modelo dinámico, pero dada sus características estáticas se incluyo en el modelo de objeto.

2.4. MODELADO DINÁMICO.

Después de examinar y comprender la estructura estática del sistema (Modelo del objeto), en donde se considera la estructura del objeto y sus relaciones entre ellos en un tiempo determinado, se procede a examinar los objetos y sus relaciones a través del tiempo.

El modelo dinámico describe los aspectos del sistema que dependen del tiempo, sin considerar las operaciones que se realizan o cómo son implementados. El modelo dinámico es insignificante para sistemas con datos estáticos por ejemplo una base de datos.

Para desarrollar un modelo dinámico eficiente se consideran los siguientes pasos : (Rumbaugh).

- Preparación del escenario.
- Identificación de los eventos entre objetos.
- Construcción del diagrama de estados.

2.4.1.PREPARACION DEL ESCENARIO.

La preparación del escenario es el primer acercamiento a la creación del modelo dinámico.

Un escenario es una secuencia de eventos que ocurre durante la ejecución de un sistema. La profundidad de un escenario puede variar, pudiendo incluir todos los eventos de un sistema o sólo aquellos eventos que generan determinados objetos del sistema.

Para la creación del escenario, primero se consideran los datos generales, posteriormente se consideran los casos especiales, como los valores máximos y mínimos, y finalmente se procede con los casos de error, tales como los valores inválidos.

Para el sistema **DFC**, el primer acercamiento de un escenario es el siguiente :

- El usuario selecciona la forma de introducir datos. (Interactivo o en lote).
- El usuario introduce los datos (Datos, NoSubGpos, TamSubGpo)
- El sistema verifica consistencia de los datos.
- El usuario introduce el tipo de gráfica de control. En caso de no existir, se dá tipo de gráfica por omisión.
- El sistema calcula los atributos necesarios para desplegar la gráfica de control. X, R, LC, LIC, LSC, PuntosGra
- El sistema calcula los subgrupos fuera de control. (SusGpoFra).
- El sistema despliega la gráfica de control señalando puntos fuera de control.
- El sistema despliega la conclusión preliminar y la indicación del nivel de control.
- El sistema recalcula y repite procedimiento.

Los casos especiales y los casos de error se irán identificando conforme se desarrolle el modelo dinámico y el modelo funcional.

2.4.2. IDENTIFICACION DE LOS EVENTOS ENTRE LOS OBJETOS.

A los atributos y asociaciones que forman parte del modelo del objeto, se le conocen como los estados del sistema. Los estados del sistema pueden variar ante el estímulo de otro objeto, dependiendo de sus características. A los estímulos que realiza un objeto se le denominan **eventos**. Los eventos son *sucesos que no tienen duración a través del tiempo y se encargan del transporte de información de un objeto a otro.*

Para identificar los eventos entre objetos, se examina el escenario para identificar los eventos externos, tales como entradas, decisiones, interrupciones, transiciones del usuario o del algún componente externo. Los cálculos de cómputo no se consideran eventos a menos que interactúe con algún agente externo.

Después de identificar los eventos, se procede a agrupar aquellos eventos cuyos parámetros o valores no afectan al control lógico.

Considerando el escenario desarrollado para el sistema **DFC**, se obtiene el siguiente diagrama de eventos :

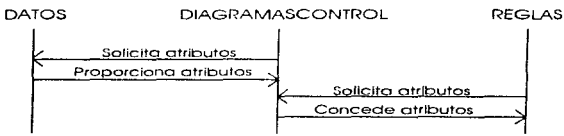


Diagrama de eventos.

Una vez elaborado el diagrama de eventos, se procede a construir el diagrama de flujo de eventos. En este diagrama se resumen los eventos entre las clases, sin considerar su secuencia. El diagrama de flujo de eventos es la contraparte dinámica del diagrama de objetos.

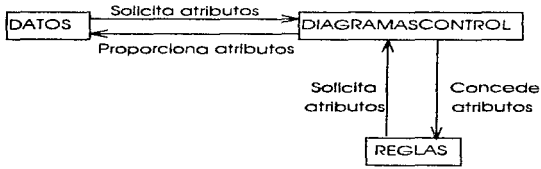


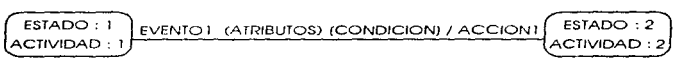
Diagrama de flujo de eventos.

2.4.3. CONSTRUCCION DEL DIAGRAMA DE ESTADOS.

El diagrama de estados es el elemento fundamental del modelo dinámico. En general, está formado por : **eventos** y **estados**. Representa la secuencia de eventos y estados que se llevan a cabo por un objeto en particular, esto es, cada objeto posee un diagrama de estados específico.

Los eventos son los *estímulos del objeto que se desarrollaron en el diagrama de eventos* (4.2.2.). Los estados son *la abstracción de los atributos y enlaces de un objeto*. Los estados tienen cierta duración a través del tiempo, que depende de la intervención anterior y posterior de los eventos. Los estados y los eventos se encuentran estrechamente relacionados.

Para desarrollar el diagrama de estados, los estados se representan por medio de rectángulos circulares y los eventos se representan a través de arcos o trayectorias entre los estados. Dentro de los rectángulos de los estados se especifica el nombre del estado y la actividad que se realiza. En las trayectorias de los eventos se describen el nombre del evento, atributos, condiciones y la acción que realiza dicho evento.



Los diagramas de estado pueden ser cíclicos o lineales. Si existe una secuencia repetitiva de eventos se considera un ciclo. En el caso de los diagramas lineales, la clase u objeto al cual representa tiene un tiempo determinado de vida. Su estado inicial se representa a través de un círculo sólido, y su estado final se representa con un círculo.

Al igual que en las asociaciones de los diagramas de objetos, en los estados y los eventos se pueden aplicar los conceptos de agregación y generalización (2.3.6.).

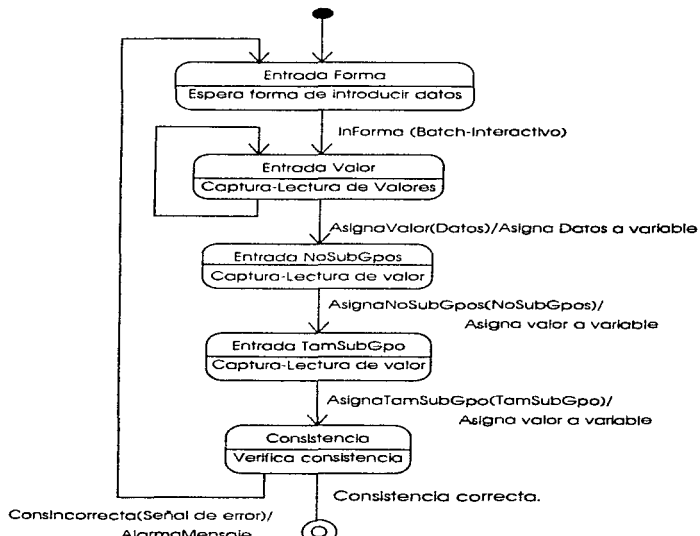
Para la construcción de los diagramas de estado se consideran los siguientes puntos : (Rumbaugh).

- Minimizar el uso de ciclos en lo posible. Los ciclos causan error, ya que el estado inicial y el estado final son iguales.
- Localizar los estados que se encuentren repetidos o que compartan características similares, para crear superestados a través de la generalización.
- Considerar los casos especiales y los casos de error.
- Con los estados y los eventos obtenidos, crear un modelo dinámico más simple, utilizando agregación.

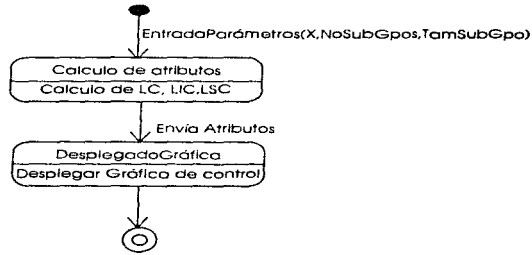
Si existen clases en las que sus estados o eventos no influyen en el control lógico de la aplicación, dichas clases no necesitan del diagrama de estados.

A continuación se desarrollan los diagramas de estados para las clases del sistema **DFC**.

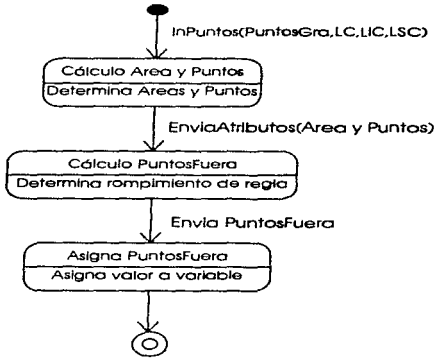
DATOS



DIAGRAMACONTROL



REGLAS



Al obtener los diagramas de estados, se procede a verificar la consistencia del mismo, considerando los diagramas de estados de todos lo objetos que interactúan en la aplicación. Para ello es necesario considerar lo siguiente :

- Cada evento posee un estado anterior y uno posterior.
- Cada estado posee un evento anterior y uno posterior.
- Los eventos entre las clases deben ser consistentes.

2.5. MODELADO FUNCIONAL.

El modelo funcional describe las operaciones dentro de un sistema, la forma en que los valores de entrada se modifican o se realizan operaciones en ellas para generar los valores de salida. En este modelo se especifica la operación, pero no se especifica cuando se realizan dichas operaciones.

El modelo funcional consta de múltiples diagramas de flujo de datos, en los cuales se muestra el flujo de los datos desde la entrada, las operaciones aplicadas, hasta la salida de datos. Dichos diagramas de flujo son de gran utilidad para mostrar la dependencia funcional entre los valores.

Para la construcción de un modelo funcional eficiente es recomendable seguir los siguientes pasos : (Rumbaugh).

- Identificación los valores de entrada y salida.
- Construcción el diagrama de flujo de datos, mostrando las dependencias funcionales entre ellos.
- Descripción de las funciones.
- Identificación de las restricciones.

2.5.1. IDENTIFICACIÓN DE LOS VALORES DE ENTRADA Y SALIDA.

Los valores de entrada y salida son los parámetros de los eventos que relacionan el sistema con el mundo exterior. La identificación de estos valores es el primer paso para la construcción del modelo funcional. Para ello es importante el enunciado del problema con el fin de tomar en cuenta todos aquellos valores que hayan sido ignorados u olvidados.

De acuerdo con el enunciado del problema, las necesidades del sistema **DFC** y las clases creadas, se tienen los siguientes valores de entrada y salida :

- Valores de entrada y salida del sistema **DFC** :
  Entrada :
  Datos.
  NoSubGpos. (Cantidad de subgrupos de los datos a analizar.)
  TamSubGpo. (Cantidad de elementos de cada subgrupo).
  Salida :
  PuntosFuera.

- Valores de entrada y salida del objetos **DATOS** :
  Entrada :
  Datos.
  NoSubGpos.
  TamSubGpo.
  Salida :
  X
  R
  NoSubGpos.
  TamSubGpo.

- Valores de entrada y salida del objeto **DIAGRAMACONTROL** :
  Entrada :
  X o R
  NoSubGpos
  TamSubGpo.
  Salida :
  Gráfica de control.

- Valores de entrada y salida del objeto **REGLA** :
  Entrada :
  LC
  LSC
  LIC.
  PuntosGra

Salida :

PuntosFuera.

2.5.2.CONTRUCCION DEL DIAGRAMA DEL FLUJO DE DATOS.

El Diagrama del flujo de datos muestra la relación funcional de los valores operados en un sistema, incluyendo valores de entrada, valores de salida y valores internos. Muestra los valores del objeto origen, los procesos que modifican dichos valores, y el flujo de dichos valores hasta el objeto destino. No especifican las decisiones o la secuencia de operación.

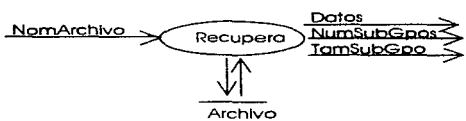
La construcción del diagrama del flujo de datos se realiza a través de diversas capas. La capa superior se forma con procesos sencillos. Para los procesos subsecuentes se consideran las entradas y salidas de un proceso para especificar, mediante otros diagramas de flujo de datos, la forma en que se manejan dichos valores.

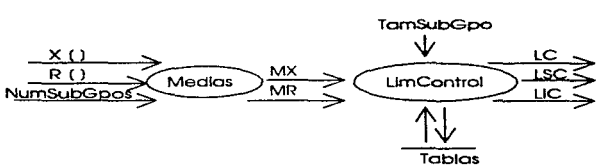
El diagrama del flujo de datos contiene los siguientes elementos :

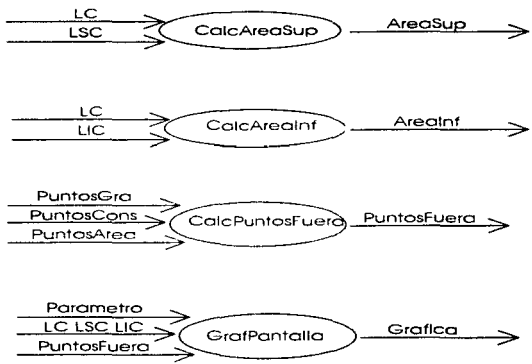
- **Procesos** : Todo aquello que modifica los valores. En el diagrama del flujo de datos se representan mediante una elipse que contiene el nombre de la operación correspondiente. La descripción detallada de las operaciones se realiza posteriormente y puede ser en lenguaje natural, ecuación matemática o por algún otro medio. Los procesos son implantados como métodos en la clases en el modelo de objeto.
- **Flujo de datos** : Representación de la conexiones entre los procesos o los objetos. Su función es la de transportar datos de un proceso a otro. Los valores no se modifican en su trayecto. Un flujo de datos se representa mediante arcos entre los procesos correspondientes.
- **Actores** : Es un objeto activo que maneja el flujo de datos generando o recibiendo valores. Los actores generan las entradas o reciben las salidas del flujo de datos. Los actores se representan mediante rectángulos, representando un objeto. *La implantación de los actores se representa mediante un objeto,* que en sí es un manejador externo.

- **Almacén de datos** : Es un objeto pasivo dentro del diagrama de flujo de datos que almacena datos para una posterior utilización. El almacén de datos se caracteriza por la posibilidad de acceder los datos en un orden diferente a como fueron almacenados. Se representa mediante un par de líneas paralelas, y entre éllas se denota el nombre del almacén. *La implantación de un almacén de datos en un lenguaje orientado a objetos se realiza mediante la creación de un objeto archivo.*

Para el sistema **DFC**, algunos de los diagramas de flujo de datos serían los siguientes :







2.5.3. DESCRIPCION DE LAS FUNCIONES.

En el proceso de refinamiento del diagrama del flujo de datos, se incluye una descripción de las funciones. Dicha descripción puede hacerse a través de:

- Funciones matemáticas.
- Tabla de valores de entrada y salida para un número de elementos pequeño.
- Ecuaciones que expresen los elementos de salida en términos de elementos de entrada.
- Tabla de decisiones.
- Código de programación.
- Lenguaje natural.

Durante la etapa de diseño es necesario especificar las operaciones de acceso público y privadas, con la finalidad de asegurar el encapsulado.

Para los procesos descritos del sistema **DFC** se desarrolla la siguiente descripción :

- Recupera : Función que se encarga de recuperar los datos almacenados en disco para su posterior evaluación.
- Captura : Rutina que se encarga de solicitar los datos al usuario y a la vez, verificar la consistencia de los mismos.
- CalX : Elabora un arreglo de dimensión m. (numero de subgrupos), a partir del cálculo de las medias de cada subgrupo. Esto genera la columna de las medias de X de la figura 1.3. Para cada elemento del arreglo se usa la siguiente fórmula :

$$\overline{X} = \frac{\sum_{i=1}^{TamSubGpo} X}{TamSubGpo}$$

- CalR : Elabora un arreglo de magnitud m. (numero de subgrupos), conformado por los rangos de cada subgrupo. Con ello se genera la columna de los rangos de la figura 1.3. Para cada elemento del arreglo, se utiliza la siguiente fórmula :

$$R = Valormax - Valormin$$

- Medias : Calcula la media de los arreglos, que puede ser el de las medias, de los rangos y .desviaciones estándar Con ello produce las siguientes salidas: MX, MR y MS.
  - MX : Media de las medias de cada subgrupo.
  - MR : Media de los rangos de cada subgrupo.
  - MS : Media de las desviaciones estándar de cada subgrupo.

  Considerando que $\overline{Y}$ toma los valores de X, R y s se toma la siguiente fórmula:

$$\overline{Y} = \frac{\sum_{i=1}^{NumSubGpos} Y}{NumSubGpos}$$

- LimControl : Calcula los límites de control (LC, LSC y LIC) de un cierto diagrama de control. Para ello se consideran las fórmulas siguientes :

Para diagramas de X :

$$LC = \overline{\overline{X}}$$

$$LSC = \overline{\overline{X}} + \frac{3\overline{R}}{d2\sqrt{TamSubGpo}}$$

$$LSC = \overline{\overline{X}} - \frac{3\overline{R}}{d2\sqrt{TamSubGpo}}$$

Para diagramas de R :

$$LC = \overline{R}$$
$$LIC = D3 * \overline{R}$$
$$LSC = D4 * \overline{R}$$

- CalcAreaSup : Calcula el área superior respecto a la línea central de un diagrama de control. Esta área permite la aplicación de una determinada regla para corridas. El cálculo se basa en la siguiente fórmula :

$$AreaSup = \frac{LSC - LC}{3}$$

- CalcAreaInf : Calcula el área inferior respecto a la línea central de un diagrama de control. Esta área permite la aplicación de una determinada regla para corridas. El cálculo se basa en la siguiente fórmula :

$$AreaInf = \frac{LC - LIC}{3}$$

- GrafPantalla : Despliega gráfica de control con todos sus componentes y señala los subgrupos fuera de control.

2.5.4. IDENTIFICACION DE LAS RESTRICCIONES ENTRE OBJETOS.

Las restricciones son las dependencias funcionales entre objetos que no están relacionadas por una dependencia de entrada-salida. Las restricciones pueden ser entre dos objetos al mismo tiempo. entre instancias del mismo objeto en tiempo diferido. O entre instancias de diferentes objetos en tiempo diferido.

Las precondiciones son restricciones para los valores de entrada que deben satisfacer una condición. Las postcondiciones son restricciones que deberán de cumplir los valores de salida. Durante el análisis es necesario considerar las restricciones dentro del modelo dinámico y modelo funcional para completar la aplicación.

Para el sistema **DFC**, se pueden identificar de primera instancia las siguientes restricciones :

- La cantidad de datos recibidos en la clase DATOS, debe ser coherente con el número de subgrupos (NoSubGpos) y el tamaño del subgrupo (TamSubGpo).
- Los valores de X y R de la clase DATOS, no pueden ser valores negativos.
- El límite inferior de control (LIC) de la clase DIAGRAMACONTROL aplicado a un diagrama de R no podrá tomar un valor negativo. En caso de que resulte negativo su calculo, tomará el valor de cero.

2.6. RESUMEN DE CAPITULO.

En este capítulo se describe y se implanta la primera etapa del proceso de desarrollo del sistema **DFC** (Detección de faltas de control, conforme la metodología **OMT** (Object Modeling Technique) de **Rumbaugh**.

Para lograr dicha etapa (Análisis) se elabora el :

- Enunciado del problema.
- Modelado de objeto.
- Modelado dinámico.
- Modelado funcional.

Se presentan una serie de diagramas del sistema **DFC.** Esenciales para el desarrollo del sistema y muestra una descripción general del problema a resolver.

- Diagrama de objeto.
- Diagrama de estados.
- Diagrama de flujo de datos.

Los diagramas anteriormente señalados se retoman para su refinamiento, en la etapa de diseño del objeto que se describe en el capítulo III.

El siguiente paso para el desarrollo del sistema **DFC**, bajo la metodología **OMT** es el diseño, y se describe y desarrollo en el capitulo III.

CAPITULO III

DISEÑO

3.1. INTRODUCCIÓN.

En la metodología **OMT**(Rumbaugh) se identifican dos etapas fundamentales para el desarrollo de sistemas :

1.- Análisis

2.- Diseño.

La etapa de análisis fue descrita y desarrollada para el sistema DFC, en el capítulo anterior.

La etapa de análisis tiene como finalidad describir **QUE** es lo que va a realizar el sistema. La finalidad de la etapa de diseño es describir **COMO** se va a solucionar el problema.

La etapa de diseño se divide a su vez en dos subetapas :

1.- Diseño del sistema. Se realizan decisiones de alto nivel acerca de la arquitectura y de las características de desarrollo para su optimización, así mismo se eligen las estrategias para atacar el problema.

2.- Diseño del objeto. Se construye un modelo basado en los resultados de la etapa del análisis, lo cual se identifican los detalles requeridos para la implantación, incorporando las estrategias establecidas durante el diseño del sistema. El punto principal del diseño del objeto es el desarrollo de la estructura de datos y de los algoritmos necesarios para la implantación de cada una de las clases.

En el transcurso del capítulo III se describen y se desarrollan las subetapas del diseño del sistema y del diseño del objeto para el desarrollo del sistema **DFC**.

El capitulo III se desarrolla en tres secciones fundamentales :

La **SECCIÓN 3.2** describe el diseño del sistema, identificando los pasos requeridos para el desarrollo eficiente de la etapa y su concretización para el desarrollo del diseño del sistema para el sistema **DFC**.

La **SECCIÓN 3.3** describe y desarrolla el diseño del objeto para el sistema **DFC**. Se concretizan los pasos recomendados por **Rumbaugh** para un eficiente desarrollo del sistema y su aplicación para la implantación eficiente del sistema **DFC**.

La **SECCIÓN 3.4** describe las arquitecturas comunes de los sistemas. Así como la importancia que cobra cada modelo dependiendo del tipo de sistema.

3.2. DISEÑO DEL SISTEMA

El diseño del sistema es una estrategia de alto nivel para la solución de un problema y la construcción de su solución. El diseño del sistema incluye decisiones acerca de la organización de subsistemas dentro de los sistemas y el lugar que ocupa cada subsistema dentro de los componentes de software y hardware.

A la organización del sistema se le denomina arquitectura del sistema. La arquitectura del sistema muestra la organización de todo el sistema con componentes llamados subsistemas.

El diseño del sistema se realiza mediante los cuatro pasos siguientes :

- Descomposición el sistema el subsistemas.
- Identificación de concurrencias principales del problema.
- Selección de equipo para la instalación de subsistemas.
- Especificación de las condiciones y prioridades.

3.2.1. DESCOMPOSICIÓN DEL SISTEMA EN SUBSISTEMAS.

El primer paso para el diseño del sistema consiste en la división del sistema en pequeños componentes. Cada componente denominado subsistema posee funciones específicas, a través de los cuales proporcionan servicios al resto del sistema, a través de un medio de comunicación. Este medio de comunicación se identifica con las entradas y salidas de parámetros o variables de cada subgrupo.

Con el desarrollo del sistema a través de subsistemas se logra la implantación independiente sin afectar el propósito del sistema.

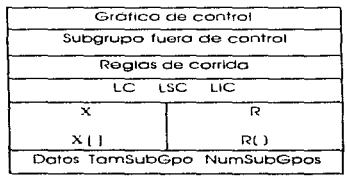
La descomposición del sistema en subsistemas puede ser organizado por capas horizontales o bloques verticales,

En un sistema con capas horizontales los subsistemas son secuenciales, uno arriba de otro. Los objetos de cada capa pueden ser independientes aunque existan relación entre los objetos de distintas capas. La relación entre capas es unidireccional. Es decir, una capa tiene conocimiento de su capa inferior pero no tiene conocimiento de su capa superior. Por ejemplo para el sistema **DFC** los límites de control y los subgrupos fuera de control serían dos capas horizontales. La capa de

subgrupos fuera de control tiene conocimiento de los límites de control pero no sucede a la inversa.

En un sistema con bloques verticales, el sistema se divide en varios subsistemas independientes, cada subsistema provee algún tipo de servicio.

El siguiente diagrama muestra el sistema **DFC** dividido en subsistemas.



Como se puede observar el sistema está constituido por capas horizontales (Gráfica de control, subgrupo fuera de control, Reglas de corrida, Límites de control, Datos), y bloques verticales (X ( ) y R ( )).

3.2.2. IDENTIFICACIÓN DE CONCURRENCIAS .

Uno de los propósitos del diseño de sistema es identificar los objetos concurrentes y los objetos independientes.

Dos objetos son concurrentes si ambos pueden recibir eventos al mismo tiempo sin la interacción entre ellos. El modelo dinámico es la guía para identificar los eventos concurrentes.

Es preferible crear subsistemas independientes ya que reduce significativamente el costo y tiempo ocasionado por la comunicación entre subsistemas.

Dadas las características del sistema **DFC** y por la forma en que se desarrolla no existen subsistemas o eventos concurrentes. Todos los subsistemas son independientes.

3.2.3. SELECCIÓN DEL MEDIO PARA LA INSTALACIÓN DE LOS SUBSISTEMAS.

Los subsistemas pueden ser instalados en :

1.- Software.

2.- Hardware.

Durante el diseño del sistema se decide en donde se instalarán los subsistemas, considerando : Costo, compatibilidad, eficiencia del equipo y flexibilidad para futuros cambios de diseño.

Los subsistemas se instalan en hardware por las siguientes razones :

- Existen equipos que proporcionan la funcionalidad requerida.
- Se requiere un alto rendimiento que un procesador de propósito general no puede proporcionar y existen procesadores que proporcionan tal eficiencia.

Los subsistemas se instalan en software por las siguientes razones :

- Es suficiente el rendimiento proporcionado por un procesador de propósito general.
- Existen planes futuros para la modificación o mantenimiento del sistema.. El software es más flexible para futuros cambios.

Para el sistema **DFC** no es necesario desarrollar algún tipo de elección, ya que por las especificaciones iniciales del sistema, y considerando el enunciado del problema, el sistema se instalará e implementará para equipo PC compatible con interfaz gráfica 3.X o superior. Esto considerando la cantidad de equipos instalados en el presente y la disponibilidad del mismo para la aplicación del sistema **DFC**.

3.2.4. ESPECIFICACIÓN DE LAS CONDICIONES Y PRIORIDADES.

En el diseño del sistema es necesario determinar las condiciones que rigen el funcionamiento del sistema. Las tres condiciones principales son : inicio, fin y error.

La condición de inicio especifica los parámetros y valores asignados a los datos y variables para inicializar el sistema.

La condición de fin indica la realización de la finalidad del sistema.

La condición de error se determina para lograr un sistema menos vulnerable a condiciones adversas, tales como, parámetros fuera de rango, operaciones no definidas y errores por parte del usuario.

Durante el diseño del sistema es necesario considerar las prioridades en los siguientes aspectos para lograr el sistema deseado :

- Cantidad de información a procesar.
- Rapidez de ejecución.
- Costo.

La determinación de prioridades es importante, dado que, alguno de los aspectos son inversamente proporcionales, esto es, a mayor cantidad de información a procesar, mayor costo y menor velocidad de respuesta. Las prioridades son filosofías de diseño que sirve como guía para el diseño de sistema, y varían dependiendo del juicio e interpretación del diseñador .

Para el sistema **DFC** se tienen las siguientes condiciones :

- Condición de inicio : Asignar valores coherentes a los atributos : Datos, NoSubGpos, TamSubGpo de la clase DATOS. El total de datos será de (NoSubGpos * TamSubGpo). Divididos en subgrupos. El número máximo de datos será de 8 elementos por cada subgrupo y 50 subgrupos.
- Condición de fin : Posterior al determinar recálculo y desplegar el diagrama de control correspondiente.
- Condición de error : El usuario no introduce la cantidad de datos establecidos por el mismo. El usuario introduce los valores de TamSubGpo y NoSubGpos igual a cero. Los valores de X y R de la clase DATOS toman valores negativos. El límite inferior de control de la clase DIAGRAMACONTROL toma un valor negativo.

3.3. DISEÑO DEL OBJETO.

En el diseño del objeto se definen con detalle las clases, asociaciones y algoritmos desarrollados durante el análisis. Dicho detalle enfocada a la implantación, esto es, considerando :

- Tiempo de ejecución.
- Memoria.
- Equipo disponible.
- Medio utilizado para implantación.

Las etapas que se siguen para el desarrollo del diseño de objeto son : (Rumbaugh)

- Integrar los tres modelos para obtener las operaciones entre las clases.
- Diseñar algoritmos para la implantación de operaciones.
- Implantar el control para las interacciones externas.
- Ajustar la herencia.
- Empaque físico.
- Documentar las decisiones de diseño.

El diseño de objeto es un proceso interactivo entre los tres modelos obtenidos en el análisis y las decisiones tomadas en e diseño del sistema. Durante el diseño del objeto es importante especificar en lo posible las operaciones y atributos para crear con ello nuevas operaciones, atributos y clases necesarias para una implantación eficiente del diseño.

3.3.1. INTEGRACIÓN DE LOS TRES MODELOS.

Los tres modelos (Objeto, Dinámico y Funcional) se encuentran estrechamente relacionados, tanto por sus componentes como por sus funciones.

El modelo del objeto obtenido durante la fase de análisis no contiene operaciones, las operaciones se obtienen a partir de las acciones y actividades del modelo dinámico y de los procesos del modelo funcional. Para completar el modelo del objeto, se consideran los modelos dinámico y funcional tomando en cuenta los siguientes puntos :

DEL MODELO DINÁMICO :

- El modelo dinámico especifica los cambios permitidos en los objetos del modelo de objeto.
- Los *estados* equivalen a los atributos y asociaciones que posee un objeto en un tiempo determinado.
- Los *estados* se definen por las interacciones entre objetos y eventos.
- Los *eventos* pueden ser representados como las operaciones en el modelo de objeto.
- Los *eventos* pueden ser definidos a través de diferentes clases de objetos.
- Todo *evento* recibido por un objeto es asociado con una operación.
- Generalmente los *eventos* ocurren por pares de eventos, un objeto que manda un evento y otro que lo recibe y responde son alguna respuesta.
- Una *transición* es un cambio de estado de un objeto que se representa como una operación dentro de un objeto..
- El nombre de las operaciones corresponde al nombre de los eventos.

Retomando el diagrama de estados de la sección 2.4.3. se tienen los siguientes estados y eventos :

- **Clase DATOS** :
  - Estados .
    - EntradaForma
    - EntradaValor
    - EntradaNoSubGpos
    - EntradaTamSubGpo
    - Consistencia
  - Eventos .
    - InForma.
    - AsignaValor.
    - AsignaNoSubGpos.
    - AsignaTamSubGpo.
    - ConsIncorrecta
    - ConsCorrecta.

- **Clase DIAGRAMACONTROL** :
  - Estados .
    - Calculo de atributos
    - DesplegadoGrafica.
  - Eventos.
    - EntradaParámetros.
    - Envía Atributos.

- **Clase REGLA** :
  - Estados.
    - Cálculo Area y Puntos.
    - Cálculo PuntosFuera.
    - Asigna PuntosFuera.
  - Eventos.
    - InPuntos.
    - EnviaAtributos
    - Envía PuntosFuera.

DEL MODELO FUNCIONAL :

- El modelo dinámico muestra la secuencia en que las operaciones se desarrollan.
- Los *procesos* dentro del modelo funcional son operaciones en objetos. En algunas ocasiones un proceso corresponde a varias operaciones y en otras ocasiones una operación corresponde a varios procesos.
- Los *actores* son objetos explícitos en el modelo de objeto.
- El *almacén de datos* es un objeto pasivo que responde únicamente a llamadas.
- Los *flujos de datos* son valores en el modelo de objeto.

Retomando los diagramas de flujo de datos de la sección 2.5.2. se tienen los siguientes procesos :

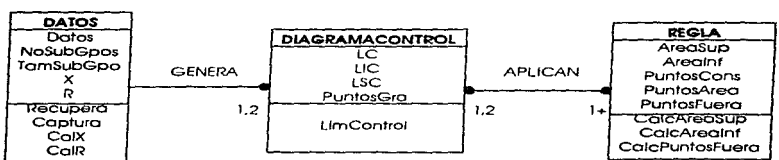
- **Clase DATOS** :
  - Recupera.
  - CalX
  - CalR
- **Clase DIAGRAMACONTROL** :
  - Medias.
  - LimControl.
  - GrafPantalla.

- **Clase REGLA** :
  - CalcAreaSup
  - CalcAreaInf
  - CalcPuntosFuera

Combinando las operaciones obtenidas de los diagramas dinámico y funcional se tienen las siguientes operaciones para las clases del sistema **DFC** :

- **Clase DATOS** :
  1.- Recupera.
  2.- Captura.
  3.- CalX
  4.- CalR.
- **Clase DIAGRAMACONTROL** :
  1.- Medias.
  2.- LimControl.
  3.- PuntosGra.
  4.- GrafPantalla.
- **Clase REGLA** :
  1.- CalcAreaSup.
  2.- CalcAreaInf.
  3.- CalcPuntosFuera.

Una vez definidas las operaciones, es posible completar el diagrama de objeto de la sección 2.3.4. y quedaría como sigue :



Durante el diseño de objeto se tendrá que convertir la estructura gráfica del diagrama a una secuencia de pasos de un algoritmo.

3.3.2. DISEÑO DE ALGORITMOS.

Todas las operaciones especificadas en el modelo funcional deben ser traducidas en algoritmos. Durante la fase de análisis se describen las operaciones que se realizan y con el algoritmo se describe la manera *COMO* se realizan dichas operaciones. Los algoritmos pueden ser llamadas o simples operaciones. Incluso estos pueden ser recursivos.

Para el diseño de algoritmos se consideran los siguientes aspectos :

- Elegir algoritmos que minimicen costos e implementación.
- Seleccionar la estructura de datos apropiados para el algoritmo.
- Definir nuevas clases y operaciones en caso necesario.
- Asignar responsabilidades a las operaciones de cada clase.

En la elección de algoritmos se consideran aspectos como :

- Sencillez de cómputo. En este aspecto se consideran las funciones y estructura de datos que minimicen el tiempo del procesador.
- Facilidad de implantación y comprensión. Es preferible dar prioridad a la sencillez, siempre y cuando no se afecte la eficiencia del sistema.
- Flexibilidad. Es importante considerar que todo el sistema se encuentra expuesto a futuras modificaciones.

A manera de ilustración, para el sistema **DFC**, el diseño de algoritmos para la operación Recupera sería el siguiente :

Recupera: Abre un archivo de disco con los datos a graficar.

Entrada :
- NomArchivo : Nombre del archivo donde se encuentran los datos a graficar.

Salida :
- Datos : Los datos a analizar.
- NumSubGpos : Numero total de subgrupos que conforman los datos.
- TamSubGpo : Numero de elementos que conforman el subgrupo.

ALGORITMO :

Abre archivo (NomArchivo)
Asigna valor a NumSubGpos. (Verifica que no sea mayor a rango)
Asigna valor a TamSubGpo. (Verifica que no sea mayor a rango).
Asigna valores al arreglo bidimencional con los datos a analizar. (Verifica la cantidad de valores especificados).
Cierra archivo. (NomArchivo)

El diseño de algoritmos del resto de las operaciones se encuentran en el Anexo 2.

3.3.3. IMPLANTACION DEL CONTROL.

Durante el diseño es necesario refinar las estrategias para implantar los modelos de eventos y estados del modelo dinámico.

A continuación se muestran los cuatro pasos para la conversión del diagrama de estados en código de programación :

1.- Identificar el control principal. Comenzando con el estado inicial y la secuencia de eventos en condiciones normales, esto es, sin considerar condiciones de error.
2.- Identificar el control alterno. Esto se traduce en instrucciones de condición dentro del código.
3.- Identificar controles atrasados. Los controles atrasados son aquellos estados que fueron establecidos con anterioridad. Este punto se traduce en ciclos dentro del código.
4.- Los estados y transiciones restantes corresponden a condiciones especiales. Estos pueden ser implementados de diversas formas, como las condiciones de error, manejo del estado de las banderas, etc.

Toda transición de un estado corresponde a un nuevo valor de alguna variable. Dependiendo de dicho valor se procede a algún estado determinado.

Considerando, los diagramas de estados del modelo dinámico (2.4.3.), y las condiciones y prioridades desarrolladas en el diseño de sistema (3.2.4.), se obtiene el algoritmo para el sistema **DFC** descrito en el Anexo 3.

3.3.4. AJUSTE DE HERENCIA.

Durante el diseño del objeto es posible reordenar las clases creadas durante el análisis para incrementar la herencia. Durante esta etapa se consideran todas las similitudes que tengan los objetos o las operaciones para crear un objeto padre que reúna a varios objetos, con comportamiento y estructura común.

En caso de que existan operaciones similares, pero alguno de ellos con menor número de argumentos, es recomendable tomar los argumentos similares para formar una clase padre, y agregar los demás argumentos en las clases hijos. Todo esto aplicando herencia.

Al aplicar la herencia es necesario que todas las operaciones contengan la misma cantidad y tipo de argumentos y resultados. Para el caso del sistema **DFC** no es necesario realizar ningún ajuste de herencia ya que por sus características carece de ellas.

3.3.5. EMPAQUE FISICO.

El empaque físico hace referencia a la estructura que toman los modelos obtenidos durante el análisis, al momento de transformarlos a código de programación. En la estructura se consideran los aspectos o virtudes que posee el lenguaje de programación seleccionado.

El empaque físico consiste de los siguientes aspectos :

1.- Ocultación interna de información.

2.- Construcción de módulos.

3.3.5.1. OCULTACIÓN INTERNA DE INFORMACIÓN.

Una meta del diseño es lograr transformar las clases en cajas negras, esto es, los aspectos propios de la clase se consideran privadas. Los métodos y atributos *privados* son aquellos que sólo pueden ser accesados por la propia clase.

Durante el análisis no se consideró el ocultamiento interno de información. Durante el diseño se definen claramente los elementos que interactúan con el exterior. Así como los elementos que sólo conciernen a la clase que pertenecen. Estos últimos elementos son privados y sobre ellos se aplica el ocultamiento.

Durante el análisis pueden existir métodos que puedan operar sobre algunos elementos de otra clase, pero esto puede causar un desarrollo frágil, ya que algún cambio en algún método puede ocasionar comportamientos inesperados, es por ello, que durante el diseño se limita el alcance de los métodos.

Es preferible que cada método tenga un conocimiento limitado del modelado en general. A continuación se describen algunos aspectos a considerar para lograr que una operación posea un alcance limitado del modelado en general :

- Asignar a cada clase la responsabilidad de sus operaciones y asignarle la información correspondiente a la clase.
- El acceso a un atributo de un objeto perteneciente a otra clase será a través de una operación de esta misma clase.
- Definir la interacción con el exterior en un alto nivel de abstracción.
- Identificar y separar aquellas clases que interactúan entre el sistema y el mundo exterior.
- Eliminar la aplicación de métodos como resultado de otros métodos. Es preferible obtener métodos como resultado de dos operaciones.

Para llevar a cabo el ocultamiento interno de información del sistema **DFC** se consideran los siguientes aspectos :

Para la clase **DATOS** :

Los atributos :

- Datos
- NoSubGpos
- TamSubGpo

se consideran como atributos privados de la clase. Y los atributos :

- X
- R
- Tamaño
- Cantidad

se consideran como atributos públicos. Ya que dichos atributos son solicitados por otras clases. Así mismo los atributos Tamaño y Cantidad toman los valores de TamSubGpo y NoSubGpos pero se duplica dicho atributos para asegurar la integridad de la clase.

Para la clase **DIAGRAMACONTROL** :

El atributo :

- PuntosGra

Se considera como atributo privado de la clase y los atributos :

- LC
- LSC
- LIC
- Elementos

se consideran como atributos públicos. El atributo elementos es el mismo que atributo PuntosGra, pero PuntosGra se reserva como privado para asegurar la integridad de la clase.

Para la clase **REGLA** :

Los atributos :

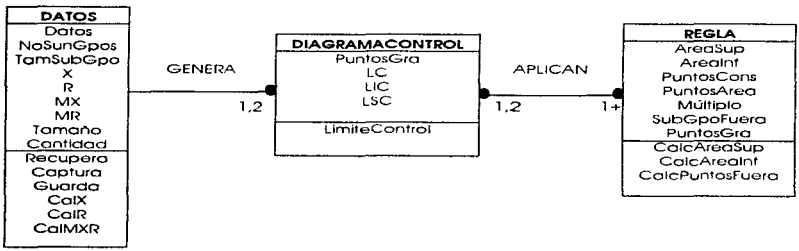
- AreaSup
- AreaInf
- PuntosCons
- PuntosArea
- Múltiplos
- PuntosFuera.

Son atributos públicos.

Para todas los métodos de todas las clases se consideran como métodos públicos.

Considerando el diseño de algoritmos, la implantación de control, y el ocultamiento interno de información, el diagrama de

objeto final para el sistema **DFC** quedaría como sigue : (Los atributos subrayados son atributos privados).



3.3.5.2. CONSTRUCCIÓN DE MÓDULOS.

Durante la etapa de análisis y de diseño del sistema se obtuvieron los modelos de objetos en módulos. Los módulos obtenidos necesitan ser organizados para la implantación final. En dicha organización se pueden crear nuevas clases que no existían durante la fase de análisis.

Un módulo puede definirse a través de otros módulos conectados a través de asociaciones.

Los módulos contienen objetos con un mismo propósito, tienen aplicaciones similares.

3.3.6. DOCUMENTACION DE LAS DECISIONES DEL DISEÑO.

Las decisiones realizadas a lo largo de la etapa de diseño, tienen que ser documentadas. De lo contrario, llegan a ser confusas. Sobre todo cuando en el desarrollo se involucra un grupo de personas, la documentación es esencial para el mantenimiento del sistema.

La documentación del diseño describe de manera detallada el modelo del objeto, tanto en forma gráfica como en

forma textual. Es recomendable incluir los comentarios adicionales para mostrar la implantación de las decisiones.

Durante el diseño del objeto se especifica a detalle el modelo funcional obtenido durante la etapa de análisis. A pesar de que se utiliza la misma notación de la utilizada durante el análisis, es importante especificar todas las operaciones, resultados, los valores de entrada-salida así como los efectos que se generan.

Para el modelo dinámico es necesaria una eficiente estructuración del código de programación.

Es recomendable tener por separado la documentación obtenida durante la etapa de análisis y la documentación obtenida durante la etapa de diseño del objeto. Con la finalidad de tener una documentación más completa y específica para el mantenimiento, esto es si en lo futuro se llevan a cabo modificaciones al sistema.

Para el sistema **DFC** es necesario tomar algunas consideraciones a las modificaciones llevadas a cabo en el ocultamiento interno de información :

- Para la clase **DATOS** se tiene :
  - Los atributos X, R, MX, MR, Tamaño, Cantidad son atributos públicos ya que otros objetos necesitan realizar referencia a ella.
  - Los atributos Tamaño y cantidad fueron creados de último momento, y se le asignan los valores de TamSubGpo y NoSubGpos. Pero éstos últimos se conservan como privados para asegurar la integridad de la clase. Esto es, toda referencia de otro objeto a los atributos TamSubGpo y NoSubGpos se realizará a través de Tamaño y Cantidad.
  - Los atributos MX y MR, a pesar de que son atributos derivados de otras clases, se consideran para completar la información de la clase.
  - Los atributos Datos, NoSubGpos y TamSubGpo son atributos privados.
- Para la clase **DIAGRAMACONTROL** :
  - Los atributos LC, LIC, LSC y Elementos son atributos públicos.
  - Se consideran tres métodos privados (TablaD2, TablaD3 y Tabla D4) que contiene las tablas necesarias para obtener los límites de control. Dichos

métodos son accesados a través del método publico LimiteControl.

- Para la clase **REGLA** :
  - Todos los atributos y métodos se consideran como públicos.

Los métodos : Medias y LimControl de la clase DIAGRAMACONTROL se integran en un sólo método denominado LimiteControl con los siguientes parámetros de entrada-salida :

Entrada :

- X ( )
- R ( )
- NoSubGpos
- TamSubGpo
- Tablas

Salida :

- LC
- LIC
- LSC

Las operaciones que se realizan son las mismas que en Medias y LimControl.

3.4. ARQUITECTURAS COMÚNES.

Existen varios tipos de arquitecturas comunes, dependiendo del tipo de sistema. Los sistemas pueden ser :(Rumbaugh)

- Transformación en lote.
- Transformación continua.
- Medio interactivo.
- Simulación dinámica.
- Sistemas en tiempo real.
- Manejo de transacciones.

Dependiendo del tipo de sistema, varia la importancia del modelo de objeto, modelo dinámico y modelo funcional, y con ello el énfasis que se aplica a determinada etapa de diseño.

3.4.1. TRANSFORMACIÓN EN LOTE.

Una transformación en lote, es una transformación secuencial de entrada-salida, esto es, en el momento en que se produce la entrada, la acción inmediata es la salida (respuesta). Entre la entrada y salida no existe ninguna interacción con el mundo exterior.

El aspecto fundamental de una interacción en lote es el modelo funcional, que especifica la forma en que los valores de entrada se transforman para producir los valores de salida.

3.4.2. TRANSFORMACIÓN CONTÍNUA.

Una transformación continua es una transformación en donde la salida depende de una continua actualización de las entradas.

Para el sistema **DFC**, se considera este tipo de transformación continua, ya que para lograr el objetivo deseado se recurre a una continua actualización de los datos. Por ejemplo, el racáiculo posterior al primer resultado de los límites de control.

El modelo funcional, junto con el modelo del objeto definen la transformación. El modelo dinámico es de poca importancia en este tipo de sistemas.

Los pasos para el diseño de un sistema de transformación continua son :

1.- Diseñar y dibujar el diagrama de flujo de datos.
2.- Definir los objetivos que intervienen entre cada etapa (estado).
3.- Especificar todas las operaciones requeridas para obtener determinada etapa.
4.- Optimización.

3.4.3. MEDIO INTERACTIVO.

Un medio interactivo es un sistema en el que predomina las interacciones entre el sistema y algún agente externo. El agente externo es independiente del sistema. Por ello, las entradas no pueden ser controladas por el sistema. El modelo dinámico

predomina en el medio interactivo. Los objetos del modelo del objeto representan los elementos que interactúan. El modelo funcional describe las funciones que se ejecutan en respuesta a las entradas.

3.4.4. SIMULACIÓN DINÁMICA.

Una simulación dinámica modela un objeto del mundo real. En este tipo de sistemas el modelo de objeto es importante y por lo general complejo y corresponden a objetos del mundo real.

El modelo dinámico es parte importante de la simulación del sistema.

3.4.5. SISTEMAS EN TIEMPO REAL.

Un sistema en tiempo real es un sistema interactivo en donde la respuesta en el tiempo es de suma importancia. Esto es, el sistema debe garantizar la respuesta en un intervalo exacto de tiempo. Para garantizar la respuesta en el tiempo se determina los escenarios con problemas externos, sus causas y soluciones.

3.4.6. MANEJO DE TRANSACCIONES.

Un manejador de transacciones es un sistema de base de datos en donde la función principal es almacenar y accesar información.

El modelo de objeto es el más importante de los tres modelos, ya que representa los objetos que se va a manejar. El modelo dinámico muestra la distribución de información. El modelo funcional no es muy importante ya que las operaciones que generalmente se realizan son actualización y consulta.

3.5. RESUMEN DE CAPITULO.

En este capítulo se describe y se desarrolla la segunda etapa del proceso de desarrollo del sistema **DFC**, conforme la metodología seleccionada. **OMT**.

Dicha etapa, denominada diseño, se divide a su vez en dos subetapas :

1.- Diseño del sistema.

2.- Diseño del objeto.

La finalidad de la etapa de diseño es describir como se va a solucionar el problema. Para ello, en el diseño del sistema se realizan decisiones de alto nivel acerca de la arquitectura del sistema. En el diseño del objeto se construye un modelo basado en los resultados de la etapa del análisis, lo cual se identifican los detalles requeridos para la implantación.

En este momento, ya se tienen los elementos necesarios para la implantación, esto es, con el desarrollo realizado hasta el momento se tienen respuestas a las preguntas :

- **QUE** va a realizar el sistema ?.
- **COMO** se va a solucionar el problema ?.

Con el desarrollo realizado hasta el momento se posee una base sólida para la realización del sistema **DFC**, ya que, se han tomado todas las decisiones para el desarrollo del sistema, y con el material obtenido se logran las ventajas del desarrollo orientado a objetos, descritos en el capítulo 2, tales como un mantenimiento más sencillo, reducción de costos, mayor flexibilidad, entre otras características.

El trabajo restante es la implantación, que consiste en la traducción del análisis y diseño realizado hasta el momento. Dicha traducción se describe y se desarrolla en el capítulo IV.

CAPITULO IV

IMPLANTACIÓN

4.1. INTRODUCCIÓN.

La implantación es la última etapa para el desarrollo de un sistema. Dicha etapa consiste en la traducción del desarrollo realizado en las etapas de análisis y diseño. La implantación, puede realizarse a través de una programación orientada a objetos o una programación estructural, considerando una implantación en software.

La programación orientada a objetos posee grandes ventajas tales como : la reutilización de código, un sistema flexible para futuros cambios, un sistema robusto, la facilidad de programación en grupo.

Un lenguaje de programación sirve para dos propósitos relacionados : Proporciona un vehículo para que el programador especifique las acciones por ejecutar, y proporciona un conjunto de conceptos que le sirven al programador para pensar en lo que es factible realizar. La primera consideración requiere idealmente de un lenguaje cercano a la máquina, a fin de poder manejar todos los aspectos importantes de una máquina en forma sencilla, eficiente y razonablemente obvia para el programador. El segundo aspecto requiere idealmente un lenguaje cercano al problema a resolver para poder expresar directa y concisamente los conceptos de una solución.

Para la programación orientada a objetos es recomendable utilizar un lenguaje orientado a objetos, ya que con ello, se aprovecha al máximo las ventajas de la programación orientada a objetos y se facilita la implantación del análisis y diseño desarrollado con anterioridad.

Al momento de considerar el lenguaje de programación para llevar a cabo la implantación, se consideran ciertas etapas para garantizar una fiel traducción del trabajo realizado en las etapas de análisis y diseño.

A lo largo de este capítulo se describen las ventajas de la programación orientada a objetos, la implantación con un lenguaje de programación orientado a objetos, así como el origen, las cualidades y el potencial del lenguaje de programación C++.

El capítulo IV se desarrolla en tres secciones fundamentales :

La **SECCIÓN 4.2.** describe la programación orientada a objetos, las ventajas y las consideraciones necesarias para lograr el beneficio de la programación orientada a objetos.

La **SECCIÓN 4.3.** describe el lenguaje de programación orientado a objetos, sus características, las acciones necesarias para llevar a cabo la implantación de un análisis y diseño orientado a objetos, así como el origen y características del lenguaje de programación C++.

La **SECCIÓN 4.4.** describe la forma en que se desarrolla y se implanta la interfaz gráfica para el sistema **DFC**, así como las características del entorno gráfico Windows™.

4.2. PROGRAMACIÓN ORIENTADA A OBJETOS.

Un buen programa de cómputo además de satisfacer los requerimientos funcionales, tiene la capacidad de ser flexible para futuros cambios, lograr un sistema robusto a fallas inesperadas, entre algunas características.

Con la programación orientada a objetos se logra el desarrollo de un sistema con las siguientes ventajas :

- Reutilización de código.
- Flexibilidad para futuros cambios.
- Fortaleza a fallas inesperadas.
- Programación en grupo.

4.2.1. REUTILIZACIÓN DE CÓDIGO.

La reutilización de código reduce la programación para el desarrollo de sistemas, así como el costo en la ejecución de pruebas de corrida.

Es posible realizar la reutilización de código a través de una programación estructural, pero con la programación orientado a objetos se facilita y garantiza la reutilización de código.

Para garantizar la reutilización de código es importante considerar los siguientes aspectos :

- Elaborar métodos coherentes : Un método es coherente si sólo ejecuta una función o un grupo concreto de

funciones. Si una función realiza dos o más tareas, es recomendable dividir dichos métodos.

- Elaborar métodos pequeños : Si la función de un método es muy grande es recomendable fraccionarlo en pequeños métodos.
- Elaborar métodos consistentes : Es recomendable utilizar nombres, condiciones, argumentos y tipo de datos similares para los métodos similares.
- Considerar un alcance uniforme : Si existen varias combinaciones para una condición de entrada, es preferible elaborar un método específico para cada combinación.
- Evitar información global : Una variable o método global hace más vulnerable al sistema ya que, en cualquier momento puede variar su valor o comportamiento y obtener un sistema impredecible.
- Minimizar las referencias externas : Un sistema con muchas referencias externas dificulta la reutilización de código.
- Utilización de herencia : Con la herencia es posible reutilizar parte de código y aprovechar (heredar) sus atributos y métodos.
- Encapsular código externo : Al momento de encapsular atributos y métodos propias del objeto facilita la reutilización de código ya que conserva la integridad de dicho código.

4.2.2. FLEXIBILIDAD PARA FUTUROS CAMBIOS.

En el desarrollo de un sistema, siempre surgen futuras modificaciones, ya sea para mejorar el sistema o para agregar características al sistema.

Los aspectos para llevar a cabo la reutilización de código ayudan de cierta forma las modificaciones futuras, pero existen otros aspectos importantes como son :

- Ocultar la estructura de datos : La estructura de datos debe ser específica para cada método. Si se exporta la estructura de datos se limita la flexibilidad para algún cambio futuro.

- Evitar múltiples asociaciones o métodos : Un método debe poseer un conocimiento limitado del objeto y su entorno, esto es, un método tiene conocimiento del objeto al que se le aplica alguna función pero desconoce la relación del objeto con su entorno.

4.2.3. FORTALEZA A FALLAS INESPERADAS.

Un método es robusto si éste no falla al momento de recibir algún parámetro improvisto. Con la construcción de métodos robustos se logra un sistema más eficiente. Para lograrlo se consideran los siguientes aspectos :

- Protección contra errores : Un sistema debe considerar todas las entradas posibles. Con la finalidad de que el sistema no realice comportamientos inesperados con la entrada de valores incorrectos.
- Validar argumentos : Es importante verificar los argumentos de toda operación externa para evitar comportamientos inesperados.
- Evitar límites predefinidos : Al momento de utilizar memoria dinámica se recomienda no limitar en lo posible su tamaño. Con la finalidad de poseer un margen de memoria para situaciones o valores inesperados.

4.2.4. PROGRAMACIÓN EN GRUPO.

La programación en grupo hace referencia al desarrollo de un sistema completo por un equipo de desarrolladores para lo cual se recomiendan los siguientes aspectos :

- No realizar una programación prematura : Antes de comenzar la programación es preciso definir el equipo de trabajo así como el análisis y diseño del sistema.
- Construir métodos sencillos : Se recomienda construir métodos sencillos con la finalidad de que todo el personal del equipo comprenda la finalidad y funcionamiento del método.

- Elegir nombres sencillos : Los nombres de los métodos deberán ser sencillos y comprensibles, esto es, utilizar nombres cortos y que expresen la finalidad del sistema.
- Utilizar diccionario de datos : Es recomendable crear un diccionario de datos que contenga la descripción de todas las clases, los métodos, las asociaciones, al cual tiene acceso el equipo de desarrollo.

4.3. LENGUAJE DE PROGRAMACIÓN ORIENTADO A OBJETOS.

La implantación natural de un análisis y diseño orientado a objetos se realiza a través de un lenguaje orientado a objetos. Esta es bastante sencilla, dado que la estructura del análisis y diseño orientado a objetos es muy similar a la estructura del lenguaje orientado a objetos. En general un lenguaje orientado a objetos soporta objetos (con la combinación de atributos y métodos), polimorfismo y herencia.

Existen diversos lenguajes orientado a objetos, tales como, C++™, Smalltalk™, CLOS™, EIFFEL™ por mencionar algunos.

El sistema **DFC** se desarrolla en lenguaje C++, considerando los siguientes aspectos :

- **Tamaño pequeño** : Es un lenguaje que contiene más operaciones y combinaciones de operaciones que palabras clave. Con ello se tienen menos reglas de sintaxis que otros lenguajes.
- **Velocidad** : La combinación de un lenguaje pequeño, un sistema de ejecución pequeño y un lenguaje cercano al hardware hace que la ejecución de muchos programas de C++ se aproxime a la de sus equivalentes en lenguaje ensamblador.
- **Manipulación de bits** : Como los orígenes están muy ligados al sistema operativo UNIX el lenguaje proporciona un amplio conjunto de operadores de manipulación de bits.
- **Variables apuntador** : Un sistema operativo tiene áreas específicas de memoria. Esta capacidad también aumenta la velocidad de ejecución de un programa. El

lenguaje C++ cubre con estos requisitos de diseño utilizando apuntadores.

- **Biblioteca de funciones especiales** : Existen muchas bibliotecas de funciones comerciales y disponibles para los compiladores de C++.
- **Constructores de clase y encapsulamiento de datos** : Los constructores de clase son el vehículo fundamental para la programación orientada a objetos. Una definición de clase puede encapsular todas las declaraciones de datos, los valores iniciales y el conjunto de operaciones, llamados métodos, para la abstracción de datos. Los objetos pueden ser declarados para una clase dada y se pueden enviar mensajes a objetos. Adicionalmente, cada objeto de una clase específica puede contener su propio conjunto privado y público de datos representativos de esa clase.
- **Constructores y destructores** : Los métodos constructores y destructores se utilizan para garantizar la inicialización de los datos definidos dentro de un objeto de una clase específica.
- **Mensajes** : El objeto es la pieza básica de la programación orientada a objetos. Cada objeto responde a un mensaje determinando una acción apropiada, basándose en la naturaleza del mensaje.
- **Sobrecarga de operadores** : Con C++ podemos dar varios significados al conjunto de operadores y funciones predefinidos, o definidos por el usuario, suministrados con el compilador.
- **Clase derivada** : Una clase derivada es como una subclase de una clase específica.
- **El uso de polimorfismo en funciones virtuales** : El polimorfismo induce a una estructura de árbol de clases padre y sus subclases.

Para la implantación a un lenguaje orientado a objetos se llevan a cabo las siguientes acciones :

- Declaración de clases.
- Creación de objetos.
- Llamada de operaciones.
- Implantación de asociaciones.

La implantación del sistema **DFC** se desarrolla en el Anexo 4.

4.3.1. DECLARACIÓN DE CLASES.

El primer paso para la implantación de un diseño orientado a objetos es la declaración de las clases.

Cada atributo y método del diagrama de objeto se declara en su correspondiente clase. Considerando los atributos y métodos, públicos y privados.

A manera de ilustración, se describe a continuación, la implantación de la clase **DATOS** del sistema **DFC** :

```
class DATOS {
    float Datos [50][8];
    int NoSubGpos;
    int TamSubGpo;

    public :
        float dat [50][8];
        float X [50];
        float R [50];
        float MX;
        float MR;
        int Tamano;
        int Cantidad;

        int Recupera(char[15]);
        void Guarda(char[15]);
        void Captura(void);
        void CalX (void);
        void CalR (void);
        void CalMXR (void);
};
```

4.3.2. CREACIÓN DE OBJETOS.

Después de declarar las clases es necesario crear los objetos para su manipulación. Al momento de crear un objeto, el lenguaje le asigna un espacio en memoria y un identificador único.

Un objeto puede ser destruido al momento en que termina su utilidad, con la finalidad de liberar espacio en memoria.

Para el sistema DFC, la creación de objetos se realiza de la siguiente manera :

```
DATOS Medicion;
DIAGRAMACONTROL Diagrama;
REGLA ATT1, ATT2, ATT3, ATT4;
```

En donde el objeto Medicion es de la clase **DATOS**, el objeto Diagrama de la clase **DIAGRAMACONTROL** y los objetos ATT1, ATT2, ATT3, ATT4 son de la clase **REGLA**.

4.3.3. LLAMADA DE OPERACIONES.

En la mayoría de los casos, una operación tiene implícito un atributo, y pueden tener más atributos. Las operaciones realizan referencias a los atributos para modificarla o realizan referencias sólo para su lectura.

Para el sistema **DFC**, las operaciones del método Captura y CalX para la clase **DATOS** se implanta de la siguiente manera :

```
void DATOS::Captura (void)  // Descripción del método Captura
{
    int x,y;
    for ( x=0 ; x<50 ; x++ )
    {
        for ( y=0 ; y<8 ; y++ )
        {
            Datos [x][y] = dat [x][y];
        }
    }
    NoSubGpos = Cantidad;
    TamSubGpo = Tamano;
}
```

```
void DATOS::CalX(void)    // Descripción del método CalX
{
    int i, j;
    float XProv;
    for (j=0 ; j<NoSubGpos ; j++)
    {
        XProv = 0;
        for (i=0 ; i<TamSubGpo ; i++)
        {
            XProv = XProv + Datos[j][i];
        }
        X[j] = XProv/TamSubGpo;
    }
}
```

4.3.4. IMPLANTACIÓN DE ASOCIACIONES.

La implantación de asociaciones se lleva a cabo a través de referencias de un objeto a otro, con apuntadores o con referencia a un atributo público de otro objeto.

4.4. ENTORNO GRÁFICO.

Los compiladores de C++ y sus herramientas asociadas para Windows™, proporcionan al desarrollador de C++ la posibilidad de compilar y enlazar programas con Windows™ de Microsoft™. Windows™ Reúne menús plegables y controles que aceptan pulsaciones de ratón y ejecutan acciones en relación a éstas.

Las aplicaciones descritas para el entorno de Windows™ de 16 bits pueden ejecutarse con Windows 3.1. ™, Windows 95™, o Windows NT™.

El sistema DFC se desarrolló en una plataforma de 16 bits, por lo tanto, pueden ejecutarse en Windows 3.1. ™, Windows 95™ y en Windows NT™.

4.4.1. CARACETRISTICAS DE WINDOWS.

Para comprender la forma en que se realiza la programación en el entorno gráfico Windows, primero describiremos sus características más importantes.

Windows es un entorno de ventanas gráficas, multitarea, que permite a los programas escritos específicamente para Windows tener un aspecto y una estructura de ordenes consistentes. Esta característica hace que los programas sean fáciles de manejar.

Windows posee las siguientes características importantes :

- **Interfaz de usuario estándar** : Entre las características más importantes y más notables es la interfaz de usuario orientado a gráficos. Esto es, la interfaz con el usuario utiliza dibujos para representar dispositivos, archivos, subdirectorios y muchas órdenes y acciones del sistema operativo.
- **Multitarea** : Un entorno operativo multitarea permite al usuario tener varios programas o varias instancias de un mismo programa, ejecutándose concurrentemente. Cada programa ocupa una ventana rectangular en la pantalla, y es posible elegir entre varios programas, cambiar de tamaño de la ventana e intercambiar información de una ventana a otra.
- **Gestión de memoria** : Windows es capaz de compactar espacio de memoria libre moviendo bloques de código y datos en la memoria del sistema.
  Acceso directo a memoria extendida.
  Hasta 16 megabytes de memoria virtual como resultado de intercambiar páginas de memoria hacia y desde disco.
- **Salida por cola** : Windows recibe todas las entradas desde el teclado, ratón y reloj en una cola del sistema. Es la cola de trabajo la que redirecciona la entrada al programa apropiado, copiándola desde la cola del sistema a la cola del programa.
- **Mensajes** : El principal medio que se utiliza para difundir información en el entorno multitarea es el sistema de mensajes, un mensaje puede verse como una notificación de que ha ocurrido algún suceso de interés que necesita o no de una acción especial.

- **Cursores e iconos independientes del dispositivo,** que selecciona automáticamente la imagen apropiada del dispositivo especificado, entre un conjunto predefinido de imágenes ofrecidas por el programa.

4.4.2. PROGRAMACIÓN EN WINDOWS.

Como mencionamos anteriormente, Windows utiliza los mensajes para difundir información, y con ello notifica al programa que ha ocurrido un suceso interesante. Todo mensaje se almacena en una cola del programa.

Los siguientes cuatro parámetros están asociados con todos los mensajes, independientemente de su tipo :

1.- Un manejador de ventanas. (Palabra de 16 bits con Windows 3.1. y de 32 bits con Windows 95 y Windows NT). Para el sistema **DFC** está declarado con **HWND**.

2.- Un tipo de mensaje (Palabra de 16 bits bajo Windows 3.1. y de 32 bits con Windows 95 y Windows NT) Para el sistema **DFC** está declarado con **UINT**.

3.- Un parámetro word (Palabra de 16 bits con Windows 3.1. y de 32 bits con Windows 95 y Windows NT). Para el sistema **DFC** el parámetro está definido con **WPARAM**.

4.- Un parámetro long (Palabra de 32 bits). Para el sistema **DFC** el parámetro está definido con **LPARAM**.

La construcción de un sistema en ambiente gráfico Windows con C++, específicamente Borland C++, es posible realizarlo a través de dos métodos : El primero de ellos está basado en mensajes para la comunicación con las funciones de Windows, y el segundo se basa en un entorno integrado de desarrollo en C++ (En Borland se le conoce como ObjectWindows). Que en sí, es una colección de objetos que describe las características estándar de Windows. El desarrollo del sistema DFC se realizó mediante mensajes para la comunicación con las funciones de Windows.

Para llevar a cabo la implantación del sistema, son necesarios cuatro archivos, que son los siguientes :

1.- Un archivo de definición (*.def), que aporta una importante información sobre el prólogo, epílogo y exportación del compilador.

Para el caso del sistema **DFC** dicho archivo se denomina DFC.def, y se lista en el Anexo 4.

2.- Un archivo de recursos (*.rc), que contienen los menús, cuadros de dialogo, cursores, iconos que se manejaran en el sistema. Para el sistema **DFC** se denomina DFC.rc. Listado en el Anexo 4. Cabe señalar que para realizar dicho archivo se utilizó el taller de recursos Borland. (*Borland Resource Workshop*).

3.- El archivo de código (*.cpp), que contiene el programa en sí, con las clases definidas en el diseño, así como las declaraciones necesarias para manejar los mensajes para el entorno gráfico. Para el sistema DFC se define como DFC.cpp.

4.- Un archivo que contiene los componentes para la compilación. (*.mak). Dicho archivo realiza todo el trabajo relacionado con compilar y enlazar. Para el sistema DFC se denomina como DFC.mak. y se lista en el Anexo 4.

4.4.3. IMPLANTACIÓN EN C++.

El desarrollo del código para el sistema **DFC** está organizado de la siguiente manera : (Anexo 4)

1.- Definición de la clase DATOS.
2.- Definición de la clase DIAGRAMACONTROL.
3.- Definición de la clase REGLA.
4.- Declaración de los procedimientos a utilizar.
5.- Definición del la clase MAIN
6.- Definición del a clase WINDOW
7.- Declaración de objetos y variables globales.
8.- La función WINMAIN
9.- Definición del procedimiento Nuevo1DlaProc
10.- Definición del procedimiento Nuevo22DlaProc
11.- Definición del procedimiento Nuevo23DlaProc
12.- Definición del procedimiento Nuevo24DlaProc
13.- Definición del procedimiento Nuevo25DlaProc
14.- Definición del procedimiento GuardaComoDlaProc
15.- Definición del procedimiento AbrirDlaProc
16.- Definición del procedimiento AbrirErrorDlaProc

17.- Definición del procedimiento EliminarDlaProc
18.- Definición del procedimiento ErrorEliminarDlaProc
19.- Definición del procedimiento CEPDlaProc
20.- Definición del procedimiento ErrorNuevoDlaProc
21.- Definición del procedimiento Ayuda1DlaProc
22.- Definición del procedimiento Ayuda2DlaProc
23.- Definición del procedimiento Ayuda3DlaProc
24.- Definición del procedimiento AcercaDeDlaProc
25.- Definición del procedimiento WndProc.

En la definición de la clase DATOS, DIAGRAMASCONTROL y REGLA, sólo se tradujo lo desarrollado en el análisis y diseño del sistema DFC. Cabe señalar que en la definición de cada clase, primero se describió la clase en sí, esto es, sus atributos y sus métodos, y posteriormente se describieron los métodos correspondientes a dicha clase.

En la definición de la clase MAIN y WINDOW se define la clase ventana, que contiene los atributos de las ventanas, tales atributos como, el color, el tipo de letra, la colocación, la barra de título, iconos, etc. Los atributos se localizan en el objeto wcApp de la clase WNDCLASS definida en windows.h. El atributo lpfnWndProc del objeto wcApp, recibe el puntero a la función de la ventana que realizará todas las tareas de la ventana. Esto es, en este momento se define el procedimiento WndProc, como el procedimiento principal que llevará a cabo todas las funciones de la ventana. Así mismo contiene los métodos para la creación de la ventana (CreateWindow), Visualizar y actualizar la ventana (ShowWindow, UpdateWindows).

WinMain es el cuerpo principal del programa para Windows. Esta función actua como el punto de entrada de la aplicación y se comporta de manera similar a la función main en un programa en C estándar. La función WinMain se encarga de lo siguiente :

- Registrar las clases de la ventana de la aplicación.
- Ejecutar la inicializaciones necesarias.
- Crear e iniciar los bucles de proceso de mensajes de la aplicación.
- Terminar el programa una vez recibido el mensaje WM_QUIT.

La definición de los procedimientos, hacen referencia e interactua con cuadros de diálogos descritos en DFC.rc, y actúan

dependiendo del cuadro de diálogo y la orden ejecutada en el mismo.

El procedimiento WndProc, es el procedimiento principal, el cual maneja los mensajes de la ventana principal, y llama a todos los procedimientos cuando son solicitados.

Como resultado de la implantación realizada, se logra un sistema con los diálogos descritos en el manual del usuario del Anexo 5.

4.5. RESUMEN DE CAPITULO.

En este capítulo se describe y se desarrolla la implantación del análisis y diseño desarrollado para el sistema **DFC**. Se describen las ventajas de considerar una programación orientado a objetos, así como las consideraciones necesarias para lograr la programación orientada a objetos.

Se desarrolla la implantación del sistema **DFC** en un lenguaje de programación orientado a objetos, considerando que la implantación natural de un análisis y diseño orientado a objetos es utilizando una programación y lenguaje orientado a objetos.

Se describe y se desarrolla las características y ventajas del entorno gráfico Windows™, así como los elementos necesarios para lograr la programación del sistema **DFC** en un entorno gráfico. Así mismo, se describen los bloques de código desarrollados para lograr la implantación del sistema **DFC**.

Con el desarrollo de este capítulo, se comprueba el potencial de un desarrollo orientado a objetos, ya que, el tiempo necesario para lograr la traducción es corta, así mismo al poseer las características del sistema en un análisis y diseño previo, esta etapa se convierte en una etapa mecánica disminuyendo el tiempo en la corrección de errores de sintaxis.

CONCLUSIONES

COCLUSIONES

Con la elaboración de esta tesis. Se cumplieron con los siguientes objetivos :

1.- Se logro desarrollar un sistema para la detección de inestabilidad o falta de control estadístico de algún proceso industrial, mediante la introducción de datos obtenidos en una selección de muestras. La detección de falta de control estadístico se logra mediante la utilización de las reglas de corridas de AT&T.

2.- Mediante el desarrollo del modelo del objeto, modelo dinámico y modelo funcional de la etapa de análisis se logra un conocimiento claro del problema a resolver, esto es, lo **QUE** va a realizar el sistema.

3.- Se comprobó la facilidad de comunicación con el personal que utiliza el sistema, esto es, una comunicación clara de los profesionales del área de la aplicación con el desarrollador del sistema. Todo ello, garantiza la realización de un sistema que cumpla con las especificaciones del usuario.

4.- A partir del diseño orientado a objetos, y con base en el desarrollo del análisis se logra tomar decisiones de manera determinante en la forma de **COMO** se soluciona el problema.

5.- Se visualizo la facilidad de implantación de un análisis y diseño orientado a objetos en una programación orientado a objetos, ya que la traducción en natural y sencilla.

6.- Se logro implantar el sistema en un lenguaje de programación orientado a objetos, Borland C++, que si bien es complejo por sus características de bajo nivel se logra un sistema eficiente.

Con el desarrollo de esta tesis, se mecanizó la visión de la orientación a objetos y esto implica una gran ventaja para futuros desarrollos, ya que se ha comprobado las ventajas de la orientación a objetos, y en la actualidad se está aplicando en diversas áreas, tales como :

- Bases de datos orientado a objetos
- Lenguajes de programación orientado a objetos
- Lenguajes visuales orientado a objetos.
- Sistemas operativos orientado a objetos.

Es importante señalar que con los objetivos logrados, se amplia el campo de desarrollo y aplicación del área de cómputo.

ANEXO 1

ANEXO 1

METODOLOGIAS PARA EL ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS.

DISEÑO ORIENTADO A OBJETOS. GRADY BOOCH.

- Se enfoca más en la fase de diseño que en la fase de análisis.
- Abarca modelos dinámicos y modelos estáticos.
- Utiliza cinco diagramas.
    1.- Diagrama de clase.
    2.- Diagrama de objeto.
    3.- Diagrama de transición de estados.
    4.- Diagrama de tiempos.
    5.- Diagrama de módulos.

INGENIERIA DE SOFTWARE ORIENTADO A OBJETOS. IVAR JACOBSON.

- Metodología dinámica.
- Utiliza empaques. (Construcción de eventos comenzando por un actor)
- Obtiene jerarquía de clases para la clasificación de objetos.
- Las operaciones en los objetos se definen por las interfaces de los mismos.
- Utiliza seis diagramas :
    1.- Diagrama de empaque.
    2.- Diagrama del dominio del objeto.
    3.- Diagrama de prueba.
    4.- Diagrama de análisis.
    5.- Diagrama de diseño.
    6.- Diagrama de implantación.

DISEÑO POR MANEJO DE RESPONSABILIDAD. REBECA WIRFS-BROCK

- Metodología dinámica.
- Se basa en el encapsulado de la estructura y el comportamiento del objeto.
- El diseño se basa en una estructura de cliente-servidor.

TECNICA DE MODELADO DE OBJETO **JAMES RUMBAUGH.**

- Metodología estática.
- Se construye un modelo de aplicación y después de agregan detalles de implantación.
- Utiliza tres diagramas :
    1.- Diagrama de objetos.
    2.- Diagrama de estados.
    3.- Diagrama de flujo de datos.

ANALISIS DE COMPORTAMIENTO DEL OBJETO. **PARCPLACE SYSTEMS.**

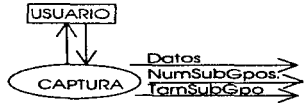
- Metodología dinámica.
- se basa en la utilización de escritos, fichas de modelo de objeto y glosarios.
- Especifica el comportamiento del objeto antes de la estructura del objeto.

ANEXO 2

ANEXO 2.

DISEÑO DE ALGORITMOS.

Captura : Rutina para capturar los valores de : Datos, NumSubGpos y TamSubGpo.



ALGORITMO :

```
Entrada valor NumSubGpos
    Verifica que no sea mayor a rango
Entrada valor TamSubGpo
    Verifica que no sea mayor a rango
Comienza ciclo j=0 hasta j=NumSubGpos.
    Comienza ciclo i=0 hasta i=TamSubGpo
        Entrada  Datos( j,i )
    Termina ciclo i
Termina ciclo j
```

CalX : Rutina para calcular el arreglo con los valores de la media de cada subgrupo.

Entrada :

- Datos.
- TamSubGpo.
- NumSubGpos.

Salida :

- X ( ) : Arreglo con los valores de la media.

```
ALGORTIMO :

Xprov = 0
Comienza ciclo j = 0 hasta j = NumSubGpos
    Comienza ciclo i = 0 hasta i = TamSubGpo
        Xprov = Xprov + Datos ( j , i )
    Termina ciclo i
    X ( j ) = Xprov / TamSubGpo
Termina ciclo j
```

CalR : Rutina para calcular el arreglo bidimencional con los valores de rango de cada subgrupo.

Entrada :

- Datos
- TamSubGpo
- NumSubGpos.

Salida :

- R ( ) : Arreglo con los valores de rango.

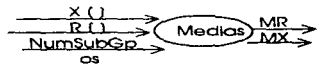

```
ALGORITMO :

Rmax = 0
Comienza ciclo j = 0 hasta j = NumSubGpos
    Rmin = Datos ( j , 1 )
    Comienza ciclo i = 0 hasta i = TamSubGpo
        Si Rmax es menor a Datos ( j , i )
            Rmax = Datos ( j , i )
        Si Rmin es mayor a Datos ( j , i )
            Rmin = Datos ( j , i )
    Termina ciclo i
    R ( j ) = RMax - Rmin
Termina ciclo j.
```

MEDIAS : Calcula las medias de los arreglos de las medias y rangos.

Entrada :
- X ( )
- R ( )
- NumSubGpos

Salida :
- MX
- MR



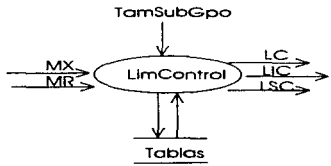
ALGORITMO :

RProv = 0
MProv = 0
Comienza ciclo I = 0 hasta I = numSubGpos
  Rprov = Rprov + R (I)
  MProv = MProv + M (I)
Termina ciclo I
MR = Rprov / NumSubGpos
MX = MProv / NumSubGpos.

LimControl : Calcula los límites de control (LC, LIC, LSC) de los diagramas de control.

Entrada :
- MX
- MR
- TamSubGpo
- Tablas.

Salida :
- LC
- LIC
- LSC



Algoritmo :

Si es para diagrama de X. (recibe como parámetro d2)
LC = MX
LSC = MX + ((3 * MR)/(d2 * (√ TamSubGpo)))
LIC = MX - ((3 * MR)/(d2 * (√ TamSubGpo)))

Si es para diagrama de R. (Recibe como parámetro D3 y D4)
LC = MR
LSC = D4 * MR
LIC = D3 * MR

CalcAreaSup : Calcula el área superior el límite central de control, en donde se aplica la regla de corrida en proceso.

Entrada :
- LC
- LSC
- Múltiplo

Salida :
- AreaSup

Algoritmo :

AreaSup = ((LSC - LC) / 3) * Múltiplo

CalcAreaInf : Calcula el área inferior el límite central de control, en donde se aplica la regla de corrida en proceso.

Entrada :
- LC
- LIC
- Múltiplo

Salida :
- AreaInf

Algoritmo :

AreaSup = ((LC - LIC) / 3) * Múltiplo

CalcPuntosFuera :Calcula los elementos o subgrupos que rompen con la regla de corrida solicitada.

Entrada :
- PuntosGra
- PuntosCons
- PuntosArea
- AreaSup
- AreaInf
- NoSubGpos

Salida :
- PuntosFuera

```
Algoritmo :

J = 0 ;
I = 0 ;
R = 0 ;
Cont = 0 ;
Mientras ( I < (NoSubGpos - PuntosCons + 1 ))
     Mientras ( R < PuntosCons )
          Si PuntosGra(I+R) > AreaSup
               Cont++
               Si Cont = PuntosArea
                    SubGpoFuera( J ) = I + R
                    J++
                    Cont = 0
                    R++ ;
               Si no
                    R++
          Si no
               R++
     Termina ciclo
          I++
          R = 0
          Cont = 0
Termina ciclo
```

ANEXO 3

ANEXO 3

IMPLANTACIÓN DE CONTROL

DATOS

Selecciona forma de introducir datos.
1.- En lote (Batch)
2.- Interactivo.
Si seleccionó en lote :
Introduce nombre del archivo.
Recupera archivo de datos.
Asigna valores a las variables.
- NoSubGpos.
- TamSubGpo
- Datos.

Verifica consistencia de las variables :
- NoSubGpos < 100
- TamSubGpo < 10
- Cantidad de datos= (NoSubGpos)*(TamSubGpo)

Si la consistencia correcta
Prosigue con el programa
Si la consistencia incorrecta
Señal de error y repite el proceso.

Si seleccionó interactivo
Introduce y asigna valor a NoSubGpos
Verifica que NoSubGpos < 100
Si no cumple repite proceso.

Introduce y asigna valor a TamSubGpo
Verifica que TamSubGpo < 10
Si no cumple repite proceso.

Introduce y asigna valor a datos
Repite (NoSubGpo * TamSubGpo) veces.

DIAGRAMACONTROL

Asigna valores a PuntosGra
Calcula los atributos :
- LC
- LIC
- LSC

Considerando los atributos de la clase DATOS.
Despliega gráfica.

REGLAS

El usuario introduce :
- Múltiplo
- PuntosCons
- PuntosArea

Calcula y asigna :
- AreaSup
- AreaInf

Determina puntos fuera de control y asigna a :
- PuntosFuera

ANEXO 4

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <fstream.h>
#include <math.h>

#include "dfc.h"

#define PALETTESIZE          256


//.....//
//                                        DEFINICION DE LA CLASE DATOS                                        //
class DATOS {
        float Datos [50][8];
        int NoSubGpos;
        int TamSubGpo;

        public :
                float dat [50][8];
                float X [50];
                float R [50];
                float MX;
                float MR;
                int Tamano;
                int Cantidad;

                int Recupera(char[15]);
                void Guarda(char[15]);
                void Captura(void);
                void CalX (void);
                void CalR (void);
                void CalMXR (void);
        };


void DATOS::Captura (void)              //      Definición del método Captura
{
        int x,y;
        for ( x=0 ; x<50 ; x++ )
        {
                for ( y=0 ; y<8 ; y++ )
                {
                        Datos [x][y] = dat [x][y];
                }
        }
        NoSubGpos = Cantidad;
        TamSubGpo = Tamano;
}
```

```
void DATOS::Guarda(char Nombre[15])   //          Definición del método Guarda
{
        int w,y;
        fstream x;

        x.open(Nombre,ios::out);
                x << NoSubGpos << "\n";
                x << TamSubGpo << "\n";
                for ( w=0 ; w<NoSubGpos ; w++ )
                {
                        for ( y=0 ; y<TamSubGpo ; y++ )
                        {
                                x << Datos[w][y] << "\n";
                        }
                }
        x.close();
}


int DATOS::Recupera(char Nombre[15])  //          Definición del método Recupera
{
        int w,y,err;
        fstream x;

        x.open(Nombre,ios::in);
                if (!x)
                {
                        err = 1;
                        return err;
                }
                float buffer;
                x >> float (buffer);
                Cantidad = buffer;
                x >> float (buffer);
                Tamano = buffer;
                for ( w=0 ; w<Cantidad ; w++ )
                {
                        for ( y=0 ; y<Tamano ; y++ )
                        {
                                x >> float (buffer);
                                dat[w][y] = buffer;
                        }
                }
        x.close();
        err = 0;
        return err;
}
```

```
void DATOS::CalX(void)              //      Definición del método CalX
{
        int i, j;
        float XProv;
        for (j=0 ; j<NoSubGpos ; j++)
        {
                XProv = 0;
                for (i=0 ; i<TamSubGpo ; i++)
                {
                        XProv = XProv + Datos[j][i];
                }
                X[j] = XProv/TamSubGpo;
        }
}

void DATOS::CalR(void)              //      Definición del método CalR
{
        int i,j;
        float RMax,RMin;
        for (j=0 ; j<NoSubGpos ; j++)
        {
                RMax = 0;
                RMin = Datos[j][0];
                for (i=0 ; i<TamSubGpo ; i++)
                {
                        if (RMax < Datos[j][i])
                                RMax = Datos[j][i];
                        if (RMin > Datos[j][i])
                                RMin = Datos[j][i];
                }
                R[j] = RMax - RMin;
        }
}

void DATOS::CalMXR(void)            //      Definición del método CalMXR
{
        int i;
        float ProvR, ProvX;

        ProvR = 0;
        ProvX = 0;
        for (i=0 ; i<Cantidad ; i++)
        {
                ProvR = ProvR + R[i];
                ProvX = ProvX + X[i];
        }
        MR = ProvR / Cantidad;
        MX = ProvX / Cantidad;
}
//.....//
```

```
//.....//
//                    DEFINICION DE LA CLASE  DIAGRAMACONTROL                    //

class DIAGRAMACONTROL
{
        float TablaD2(int);
        float TablaD3(int);
        float TablaD4(int);
        public :

                float LC, LSC, LIC;

                void LimiteControl(int,float,float,float);
};

float DIAGRAMACONTROL::TablaD2(int a) //        Definición del método TablaD2
{
        if ( a == 2 )
                return 1.128;
        if ( a == 3 )
                return 1.693;
        if ( a == 4 )
                return 2.059;
        if ( a == 5 )
                return 2.326;
        if ( a == 6 )
                return 2.534;
        if ( a == 7 )
                return 2.704;
        if ( a == 8 )
                return 2.847;
}

float DIAGRAMACONTROL::TablaD3(int a) //        Definición del método TablaD3
{
        if ( a == 2 )
                return 0;
        if ( a == 3 )
                return 0;
        if ( a == 4 )
                return 0;
        if ( a == 5 )
                return 0;
        if ( a == 6 )
                return 0;
        if ( a == 7 )
                return 0.08;
        if ( a == 8 )
                return 0.14;
};
```

```
float DIAGRAMACONTROL::TablaD4(int a)          //              Definición del método TablaD4
{
        if ( a == 2 )
                return 3.27;
        if ( a == 3 )
                return 2.57;
        if ( a == 4 )
                return 2.28;
        if ( a == 5 )
                return 2.11;
        if ( a == 6 )
                return 2.00;
        if ( a == 7 )
                return 1.92;
        if ( a == 8 )
                return 1.86;
}


void DIAGRAMACONTROL::LimiteControl(int Tipo,float TamSubGpo, float MX, float MR)
{
        if (Tipo == 0)                  //Definición del método LimiteControl
        {
                LC = MX;
                LSC = MX + ((3*MR) / (TablaD2(TamSubGpo)*(sqrt(TamSubGpo))));
                LIC = MX - ((3*MR) / (TablaD2(TamSubGpo)*(sqrt(TamSubGpo))));

        }

        if (Tipo == 1)
        {
                LC = MR;
                LSC = MR * (TablaD4(TamSubGpo));
                LIC = MR * (TablaD3(TamSubGpo));

        }
}
//-------------------------------------------------------------------------------------------------------------------------------------------//
```

```
//.....//
//                                   DEFINICION DE LA CLASE REGLA                                   //
class REGLA
{
        public :
                float AreaSup;
                float AreaInf;
                int PuntosCons;
                int PuntosArea;
                int Multiplo;
                int SubGpoFuera[25];
                float PuntosGra[50];

                void CalcAreaSup (float, float);
                void CalcAreaInf (float, float);
                void CalcPuntosFuera(int);
};


void REGLA::CalcAreaSup(float LC, float LSC)
{                       //  Definición del método CalAreaSup
        AreaSup = LC + (((LSC-LC)/3) * Multiplo);
}


void REGLA::CalcAreaInf(float LC, float LIC)
{                       //  Definicion del método CalcAreaInf
        AreaInf = LC - (((LC-LIC)/3) * Multiplo);
}


void REGLA::CalcPuntosFuera(int NoSubGpos)
{                   //      Definición del método CalcPuntosFuera
        int j,i,R,Cont;
        j=1;
        i=0;
        R=0;
        Cont=0;
        while (j<(NoSubGpos-PuntosCons+1))
        {
                while (R < PuntosCons)
                {
                        if  (PuntosGra[i+R]>AreaSup)
                        {
                                Cont++;
                                if ( Cont == PuntosArea )
                                {
                                        SubGpoFuera[j] = i+R;
                                        j++;
```

```
                                        Cont = 0;
                                        R++;
                                }
                                else
                                {
                                        R++;
                                }
                        }
                        else
                        {
                                R++;
                        }
                }
                i++;
                R = 0;
                Cont = 0;
        }
        i = 0;
        R = 0;
        Cont = 0;
        while (i<(NoSubEpos-PuntosCons+1))
        {
                while (R < PuntosCons)
                {
                        if (PuntosGra[i+R]<AreaInf)
                        {
                                Cont++;
                                if ( Cont == PuntosArea )
                                {
                                        SubEpoFuera[j] = i+R;
                                        j++;
                                        Cont = 0;
                                        R++;
                                }
                                else
                                {
                                        R++;
                                }
                        }
                        else
                        {
                                R++;
                        }
                }
                i++;
                R = 0;
                Cont = 0;
        }
        SubEpoFuera[0]=j;
}
//.....//
```

```
//.....//
//                                   DEFINICION DE FUNCIONES PASCAL                                   //
extern "C" {
        LONG FAR PASCAL WndProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Nuevo21DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Nuevo22DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Nuevo23DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Nuevo24DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Nuevo25DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL GuardarComoDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL AbrirDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL AbrirErrorDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL EliminarDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL ErrorEliminarDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL ErrorNuevoDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL CEPDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Ayuda1DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Ayuda2DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL Ayuda3DiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        BOOL FAR PASCAL AcercaDeDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam);
        }
//.....//


//.....//
//                                   DEFINICION DE LA CLASE MAIN                                      //
        class Main
        {
                public:
                static HANDLE hInst;
                static HANDLE hPrevInst;
                static int nCmdShow;
                static int MessageLoop(void);
        };

        HANDLE Main::hInst = 0;
        HANDLE Main::hPrevInst = 0;
        int Main::nCmdShow = 0;

        int Main::MessageLoop(void)
        {
                MSG lpMsg;
                while (GetMessage(&lpMsg,NULL,0,0))
                {
                        TranslateMessage(&lpMsg);
                        DispatchMessage(&lpMsg);
                }
                return lpMsg.wParam;
        }
//.....//
```

```
//.....//
//                    DEFINICION DE LA CLASE WINDOW                    //
class Window
{
        protected:
                HWND hWnd;
        public:
                HWND getHandle(void){
                return hWnd;
        }
        BOOL Show(int nCmdShow){
                return ShowWindow(hWnd,nCmdShow);
        }
        void Update(void){
                UpdateWindow(hWnd);
        }
};


class MainWindow:public Window
{
        private:
                static char szProgName[];
                static char szApplName[];
        public:
                static void Register(void)
                {
                WNDCLASS wcApp;
                wcApp.lpszClassName = szProgName;
                wcApp.hInstance = Main::hInst;
                wcApp.lpfnWndProc = WndProc;
                wcApp.hCursor = LoadCursor(NULL,IDC_ARROW);
                wcApp.hIcon = NULL;
                wcApp.lpszMenuName = "MENU_1";
                wcApp.hbrBackground = GetStockObject(WHITE_BRUSH);
                wcApp.style = CS_HREDRAW | CS_VREDRAW;
                wcApp.cbClsExtra = 0;
                wcApp.cbWndExtra = 0;

                if (! RegisterClass(&wcApp))
                        exit (FALSE);
        }

        MainWindow(void)
        {
                hWnd = CreateWindow(szProgName,
                                                "CONTROL ESTADISTICO DE PROCESOS.  OFC ",
                                                WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,
                                                CW_USEDEFAULT,CW_USEDEFAULT,
                                                CW_USEDEFAULT,(HWND)NULL,(HWND)NULL,
```

```
                                                                    Main::hInst, (LPSTR)NULL);
                if (!hWnd)
                        exit (FALSE);

                ShowMain::nCmdShow);
                        Update();
                }
        };

        char MainWindow::szProgName[]="ProgName";
        char MainWindow::szApplName[]="MENU_1";
//.....//


//.....//
//                        DECLARACION DE OBJETOS Y VARIABLES GLOBALES A UTILIZAR                //
DATOS Medicion;
DIAGRAMACONTROL Diagrama;
REGLA ATT1, ATT2, ATT3, ATT4;

char NomArchivo[15];
char *Titulo;
char *Tipo;
int error;
int SubGpoEliminar;
int GrupoFuera[25];
float ElementoGra[50];
float GraAreaSup, GraAreaInf;

HANDLE hPal;
MPLOGPALETTE pLogPal;
//.....//


//.....//
//                                WINMAIN                                                       //
int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst,
                                                        LPSTR lpszCmdLine, int nCmdShow)
{
int x,y;
for ( x=0 ; x<60 ; x++ )        // Inicializacion de variables
{                       // de la clase DATOS.
        for ( y=0 ; y<8 ; y++ )
        {
                Medicion.dat[x][y] = 0;
        }
        ElementoGra[x] = 0;
}
```

```
        Medicion.Cantidad = 1;
        Medicion.Tamano = 1;
        Medicion.Captura();
        strcpy(NomArchivo,"Nuevo.dfc");
        Titulo = ".";
        Tipo = ".";

        Diagrama.LC = 0;
        Diagrama.LIC = 0;
        Diagrama.LSC = 0;
        ATT1.PuntosCons = 1;            // Asigna valores a las clases REGLA
        ATT1.PuntosArea = 1;
        ATT1.Multiplo = 3;
        ATT2.PuntosCons = 3;
        ATT2.PuntosArea = 2;
        ATT2.Multiplo = 2;
        ATT3.PuntosCons = 5;
        ATT3.PuntosArea = 4;
        ATT3.Multiplo = 1;
        ATT4.PuntosCons = 8;
        ATT4.PuntosArea = 8;
        ATT4.Multiplo = 0;

        Main::hInst=hInst;
        Main::hPrevInst=hPrevInst;
        Main::nCmdShow=nCmdShow;

        if (!Main::hPrevInst){
                MainWindow::Register();
        }

        MainWindow MainWnd;
        return Main::MessageLoop();
}
//.....//



//.....//
//                                                    DEFINICION DEL PROCEDIMIENTO NUEVO21                                                    //
BOOL FAR PASCAL NuevoZ1DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                return FALSE;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
```

```
Medicion.Cantidad = GetDigItemInt(hWnd,DM_Cantidad,NULL,0);
Medicion.Tamano = GetDigItemInt(hWnd,DM_Tamano,NULL,0);
Medicion.dat[0][0] = GetDigItemInt(hWnd,DM_Dato1,NULL,0);
Medicion.dat[0][1] = GetDigItemInt(hWnd,DM_Dato2,NULL,0);
Medicion.dat[0][2] = GetDigItemInt(hWnd,DM_Dato3,NULL,0);
Medicion.dat[0][3] = GetDigItemInt(hWnd,DM_Dato4,NULL,0);
Medicion.dat[0][4] = GetDigItemInt(hWnd,DM_Dato5,NULL,0);
Medicion.dat[0][5] = GetDigItemInt(hWnd,DM_Dato6,NULL,0);
Medicion.dat[0][6] = GetDigItemInt(hWnd,DM_Dato7,NULL,0);
Medicion.dat[0][7] = GetDigItemInt(hWnd,DM_Dato8,NULL,0);
Medicion.dat[1][0] = GetDigItemInt(hWnd,DM_Dato9,NULL,0);
Medicion.dat[1][1] = GetDigItemInt(hWnd,DM_Dato10,NULL,0);
Medicion.dat[1][2] = GetDigItemInt(hWnd,DM_Dato11,NULL,0);
Medicion.dat[1][3] = GetDigItemInt(hWnd,DM_Dato12,NULL,0);
Medicion.dat[1][4] = GetDigItemInt(hWnd,DM_Dato13,NULL,0);
Medicion.dat[1][5] = GetDigItemInt(hWnd,DM_Dato14,NULL,0);
Medicion.dat[1][6] = GetDigItemInt(hWnd,DM_Dato15,NULL,0);
Medicion.dat[1][7] = GetDigItemInt(hWnd,DM_Dato16,NULL,0);
Medicion.dat[2][0] = GetDigItemInt(hWnd,DM_Dato17,NULL,0);
Medicion.dat[2][1] = GetDigItemInt(hWnd,DM_Dato18,NULL,0);
Medicion.dat[2][2] = GetDigItemInt(hWnd,DM_Dato19,NULL,0);
Medicion.dat[2][3] = GetDigItemInt(hWnd,DM_Dato20,NULL,0);
Medicion.dat[2][4] = GetDigItemInt(hWnd,DM_Dato21,NULL,0);
Medicion.dat[2][5] = GetDigItemInt(hWnd,DM_Dato22,NULL,0);
Medicion.dat[2][6] = GetDigItemInt(hWnd,DM_Dato23,NULL,0);
Medicion.dat[2][7] = GetDigItemInt(hWnd,DM_Dato24,NULL,0);
Medicion.dat[3][0] = GetDigItemInt(hWnd,DM_Dato25,NULL,0);
Medicion.dat[3][1] = GetDigItemInt(hWnd,DM_Dato26,NULL,0);
Medicion.dat[3][2] = GetDigItemInt(hWnd,DM_Dato27,NULL,0);
Medicion.dat[3][3] = GetDigItemInt(hWnd,DM_Dato28,NULL,0);
Medicion.dat[3][4] = GetDigItemInt(hWnd,DM_Dato29,NULL,0);
Medicion.dat[3][5] = GetDigItemInt(hWnd,DM_Dato30,NULL,0);
Medicion.dat[3][6] = GetDigItemInt(hWnd,DM_Dato31,NULL,0);
Medicion.dat[3][7] = GetDigItemInt(hWnd,DM_Dato32,NULL,0);
Medicion.dat[4][0] = GetDigItemInt(hWnd,DM_Dato33,NULL,0);
Medicion.dat[4][1] = GetDigItemInt(hWnd,DM_Dato34,NULL,0);
Medicion.dat[4][2] = GetDigItemInt(hWnd,DM_Dato35,NULL,0);
Medicion.dat[4][3] = GetDigItemInt(hWnd,DM_Dato36,NULL,0);
Medicion.dat[4][4] = GetDigItemInt(hWnd,DM_Dato37,NULL,0);
Medicion.dat[4][5] = GetDigItemInt(hWnd,DM_Dato38,NULL,0);
Medicion.dat[4][6] = GetDigItemInt(hWnd,DM_Dato39,NULL,0);
Medicion.dat[4][7] = GetDigItemInt(hWnd,DM_Dato40,NULL,0);
Medicion.dat[5][0] = GetDigItemInt(hWnd,DM_Dato41,NULL,0);
Medicion.dat[5][1] = GetDigItemInt(hWnd,DM_Dato42,NULL,0);
Medicion.dat[5][2] = GetDigItemInt(hWnd,DM_Dato43,NULL,0);
Medicion.dat[5][3] = GetDigItemInt(hWnd,DM_Dato44,NULL,0);
Medicion.dat[5][4] = GetDigItemInt(hWnd,DM_Dato45,NULL,0);
Medicion.dat[5][5] = GetDigItemInt(hWnd,DM_Dato46,NULL,0);
Medicion.dat[5][6] = GetDigItemInt(hWnd,DM_Dato47,NULL,0);
Medicion.dat[5][7] = GetDigItemInt(hWnd,DM_Dato48,NULL,0);
Medicion.dat[6][0] = GetDigItemInt(hWnd,DM_Dato49,NULL,0);
```

```
                                        Medicion.dat[6][1] = GetDlgItemInt(hWnd,DM_Dato50,NULL,0);
                                        Medicion.dat[6][2] = GetDlgItemInt(hWnd,DM_Dato51,NULL,0);
                                        Medicion.dat[6][3] = GetDlgItemInt(hWnd,DM_Dato52,NULL,0);
                                        Medicion.dat[6][4] = GetDlgItemInt(hWnd,DM_Dato53,NULL,0);
                                        Medicion.dat[6][5] = GetDlgItemInt(hWnd,DM_Dato54,NULL,0);
                                        Medicion.dat[6][6] = GetDlgItemInt(hWnd,DM_Dato55,NULL,0);
                                        Medicion.dat[6][7] = GetDlgItemInt(hWnd,DM_Dato56,NULL,0);
                                        Medicion.dat[7][0] = GetDlgItemInt(hWnd,DM_Dato57,NULL,0);
                                        Medicion.dat[7][1] = GetDlgItemInt(hWnd,DM_Dato58,NULL,0);
                                        Medicion.dat[7][2] = GetDlgItemInt(hWnd,DM_Dato59,NULL,0);
                                        Medicion.dat[7][3] = GetDlgItemInt(hWnd,DM_Dato60,NULL,0);
                                        Medicion.dat[7][4] = GetDlgItemInt(hWnd,DM_Dato61,NULL,0);
                                        Medicion.dat[7][5] = GetDlgItemInt(hWnd,DM_Dato62,NULL,0);
                                        Medicion.dat[7][6] = GetDlgItemInt(hWnd,DM_Dato63,NULL,0);
                                        Medicion.dat[7][7] = GetDlgItemInt(hWnd,DM_Dato64,NULL,0);
                                        Medicion.dat[8][0] = GetDlgItemInt(hWnd,DM_Dato65,NULL,0);
                                        Medicion.dat[8][1] = GetDlgItemInt(hWnd,DM_Dato66,NULL,0);
                                        Medicion.dat[8][2] = GetDlgItemInt(hWnd,DM_Dato67,NULL,0);
                                        Medicion.dat[8][3] = GetDlgItemInt(hWnd,DM_Dato68,NULL,0);
                                        Medicion.dat[8][4] = GetDlgItemInt(hWnd,DM_Dato69,NULL,0);
                                        Medicion.dat[8][5] = GetDlgItemInt(hWnd,DM_Dato70,NULL,0);
                                        Medicion.dat[8][6] = GetDlgItemInt(hWnd,DM_Dato71,NULL,0);
                                        Medicion.dat[8][7] = GetDlgItemInt(hWnd,DM_Dato72,NULL,0);
                                        Medicion.dat[9][0] = GetDlgItemInt(hWnd,DM_Dato73,NULL,0);
                                        Medicion.dat[9][1] = GetDlgItemInt(hWnd,DM_Dato74,NULL,0);
                                        Medicion.dat[9][2] = GetDlgItemInt(hWnd,DM_Dato75,NULL,0);
                                        Medicion.dat[9][3] = GetDlgItemInt(hWnd,DM_Dato76,NULL,0);
                                        Medicion.dat[9][4] = GetDlgItemInt(hWnd,DM_Dato77,NULL,0);
                                        Medicion.dat[9][5] = GetDlgItemInt(hWnd,DM_Dato78,NULL,0);
                                        Medicion.dat[9][6] = GetDlgItemInt(hWnd,DM_Dato79,NULL,0);
                                        Medicion.dat[9][7] = GetDlgItemInt(hWnd,DM_Dato80,NULL,0);
                                        EndDialog (hWnd,TRUE);
                        break;

                        case IDCANCEL:
                                        EndDialog (hWnd,FALSE);
                        break;
                        default :
                                return FALSE;
                        }
                break;
                default:
                        return FALSE;
                }
return TRUE;
}
//.....//
```

```
//.....//
//                    DEFINICION DEL PROCEDIMIENTO NUEVO22                    //
BOOL FAR PASCAL Nuevo22DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                return FALSE;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                                        Medicion.dat[10][0] = GetDlgItemInt(hWnd,DM_Dato81,NULL,0);
                                        Medicion.dat[10][1] = GetDlgItemInt(hWnd,DM_Dato82,NULL,0);
                                        Medicion.dat[10][2] = GetDlgItemInt(hWnd,DM_Dato83,NULL,0);
                                        Medicion.dat[10][3] = GetDlgItemInt(hWnd,DM_Dato84,NULL,0);
                                        Medicion.dat[10][4] = GetDlgItemInt(hWnd,DM_Dato85,NULL,0);
                                        Medicion.dat[10][5] = GetDlgItemInt(hWnd,DM_Dato86,NULL,0);
                                        Medicion.dat[10][6] = GetDlgItemInt(hWnd,DM_Dato87,NULL,0);
                                        Medicion.dat[10][7] = GetDlgItemInt(hWnd,DM_Dato88,NULL,0);
                                        Medicion.dat[11][0] = GetDlgItemInt(hWnd,DM_Dato89,NULL,0);
                                        Medicion.dat[11][1] = GetDlgItemInt(hWnd,DM_Dato90,NULL,0);
                                        Medicion.dat[11][2] = GetDlgItemInt(hWnd,DM_Dato91,NULL,0);
                                        Medicion.dat[11][3] = GetDlgItemInt(hWnd,DM_Dato92,NULL,0);
                                        Medicion.dat[11][4] = GetDlgItemInt(hWnd,DM_Dato93,NULL,0);
                                        Medicion.dat[11][5] = GetDlgItemInt(hWnd,DM_Dato94,NULL,0);
                                        Medicion.dat[11][6] = GetDlgItemInt(hWnd,DM_Dato95,NULL,0);
                                        Medicion.dat[11][7] = GetDlgItemInt(hWnd,DM_Dato96,NULL,0);
                                        Medicion.dat[12][0] = GetDlgItemInt(hWnd,DM_Dato97,NULL,0);
                                        Medicion.dat[12][1] = GetDlgItemInt(hWnd,DM_Dato98,NULL,0);
                                        Medicion.dat[12][2] = GetDlgItemInt(hWnd,DM_Dato99,NULL,0);
                                        Medicion.dat[12][3] = GetDlgItemInt(hWnd,DM_Dato100,NULL,0);
                                        Medicion.dat[12][4] = GetDlgItemInt(hWnd,DM_Dato101,NULL,0);
                                        Medicion.dat[12][5] = GetDlgItemInt(hWnd,DM_Dato102,NULL,0);
                                        Medicion.dat[12][6] = GetDlgItemInt(hWnd,DM_Dato103,NULL,0);
                                        Medicion.dat[12][7] = GetDlgItemInt(hWnd,DM_Dato104,NULL,0);
                                        Medicion.dat[13][0] = GetDlgItemInt(hWnd,DM_Dato105,NULL,0);
                                        Medicion.dat[13][1] = GetDlgItemInt(hWnd,DM_Dato106,NULL,0);
                                        Medicion.dat[13][2] = GetDlgItemInt(hWnd,DM_Dato107,NULL,0);
                                        Medicion.dat[13][3] = GetDlgItemInt(hWnd,DM_Dato108,NULL,0);
                                        Medicion.dat[13][4] = GetDlgItemInt(hWnd,DM_Dato109,NULL,0);
                                        Medicion.dat[13][5] = GetDlgItemInt(hWnd,DM_Dato110,NULL,0);
                                        Medicion.dat[13][6] = GetDlgItemInt(hWnd,DM_Dato111,NULL,0);
                                        Medicion.dat[13][7] = GetDlgItemInt(hWnd,DM_Dato112,NULL,0);
                                        Medicion.dat[14][0] = GetDlgItemInt(hWnd,DM_Dato113,NULL,0);
                                        Medicion.dat[14][1] = GetDlgItemInt(hWnd,DM_Dato114,NULL,0);
                                        Medicion.dat[14][2] = GetDlgItemInt(hWnd,DM_Dato115,NULL,0);
                                        Medicion.dat[14][3] = GetDlgItemInt(hWnd,DM_Dato116,NULL,0);
                                        Medicion.dat[14][4] = GetDlgItemInt(hWnd,DM_Dato117,NULL,0);
                                        Medicion.dat[14][5] = GetDlgItemInt(hWnd,DM_Dato118,NULL,0);
```

```
                    Medicion.dat[14][6] = GetDlgItemInt(hWnd,DM_Dato119,NULL,0);
                    Medicion.dat[14][7] = GetDlgItemInt(hWnd,DM_Dato120,NULL,0);
                    Medicion.dat[15][0] = GetDlgItemInt(hWnd,DM_Dato121,NULL,0);
                    Medicion.dat[15][1] = GetDlgItemInt(hWnd,DM_Dato122,NULL,0);
                    Medicion.dat[15][2] = GetDlgItemInt(hWnd,DM_Dato123,NULL,0);
                    Medicion.dat[15][3] = GetDlgItemInt(hWnd,DM_Dato124,NULL,0);
                    Medicion.dat[15][4] = GetDlgItemInt(hWnd,DM_Dato125,NULL,0);
                    Medicion.dat[15][5] = GetDlgItemInt(hWnd,DM_Dato126,NULL,0);
                    Medicion.dat[15][6] = GetDlgItemInt(hWnd,DM_Dato127,NULL,0);
                    Medicion.dat[15][7] = GetDlgItemInt(hWnd,DM_Dato128,NULL,0);
                    Medicion.dat[16][0] = GetDlgItemInt(hWnd,DM_Dato129,NULL,0);
                    Medicion.dat[16][1] = GetDlgItemInt(hWnd,DM_Dato130,NULL,0);
                    Medicion.dat[16][2] = GetDlgItemInt(hWnd,DM_Dato131,NULL,0);
                    Medicion.dat[16][3] = GetDlgItemInt(hWnd,DM_Dato132,NULL,0);
                    Medicion.dat[16][4] = GetDlgItemInt(hWnd,DM_Dato133,NULL,0);
                    Medicion.dat[16][5] = GetDlgItemInt(hWnd,DM_Dato134,NULL,0);
                    Medicion.dat[16][6] = GetDlgItemInt(hWnd,DM_Dato135,NULL,0);
                    Medicion.dat[16][7] = GetDlgItemInt(hWnd,DM_Dato136,NULL,0);
                    Medicion.dat[17][0] = GetDlgItemInt(hWnd,DM_Dato137,NULL,0);
                    Medicion.dat[17][1] = GetDlgItemInt(hWnd,DM_Dato138,NULL,0);
                    Medicion.dat[17][2] = GetDlgItemInt(hWnd,DM_Dato139,NULL,0);
                    Medicion.dat[17][3] = GetDlgItemInt(hWnd,DM_Dato140,NULL,0);
                    Medicion.dat[17][4] = GetDlgItemInt(hWnd,DM_Dato141,NULL,0);
                    Medicion.dat[17][5] = GetDlgItemInt(hWnd,DM_Dato142,NULL,0);
                    Medicion.dat[17][6] = GetDlgItemInt(hWnd,DM_Dato143,NULL,0);
                    Medicion.dat[17][7] = GetDlgItemInt(hWnd,DM_Dato144,NULL,0);
                    Medicion.dat[18][0] = GetDlgItemInt(hWnd,DM_Dato145,NULL,0);
                    Medicion.dat[18][1] = GetDlgItemInt(hWnd,DM_Dato146,NULL,0);
                    Medicion.dat[18][2] = GetDlgItemInt(hWnd,DM_Dato147,NULL,0);
                    Medicion.dat[18][3] = GetDlgItemInt(hWnd,DM_Dato148,NULL,0);
                    Medicion.dat[18][4] = GetDlgItemInt(hWnd,DM_Dato149,NULL,0);
                    Medicion.dat[18][5] = GetDlgItemInt(hWnd,DM_Dato150,NULL,0);
                    Medicion.dat[18][6] = GetDlgItemInt(hWnd,DM_Dato151,NULL,0);
                    Medicion.dat[18][7] = GetDlgItemInt(hWnd,DM_Dato152,NULL,0);
                    Medicion.dat[19][0] = GetDlgItemInt(hWnd,DM_Dato153,NULL,0);
                    Medicion.dat[19][1] = GetDlgItemInt(hWnd,DM_Dato154,NULL,0);
                    Medicion.dat[19][2] = GetDlgItemInt(hWnd,DM_Dato155,NULL,0);
                    Medicion.dat[19][3] = GetDlgItemInt(hWnd,DM_Dato156,NULL,0);
                    Medicion.dat[19][4] = GetDlgItemInt(hWnd,DM_Dato157,NULL,0);
                    Medicion.dat[19][5] = GetDlgItemInt(hWnd,DM_Dato158,NULL,0);
                    Medicion.dat[19][6] = GetDlgItemInt(hWnd,DM_Dato159,NULL,0);
                    Medicion.dat[19][7] = GetDlgItemInt(hWnd,DM_Dato160,NULL,0);
                    EndDialog(hWnd,TRUE);
break;

case IDCANCEL:
                    EndDialog (hWnd,FALSE);
break;
default :
        return FALSE;
}
```

```
        break;
        default:
                return FALSE;
        }
return TRUE;
}
//.....//



//.....//
//                                  DEFINICION DEL PROCEDIMIENTO NUEV023                                  //
BOOL FAR PASCAL Nuevo23DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                return FALSE;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                                        Medicion.dat[20][0]=GetDlgItemInt(hWnd,DM_Dato161,NULL,0);
                                        Medicion.dat[20][1]=GetDlgItemInt(hWnd,DM_Dato162,NULL,0);
                                        Medicion.dat[20][2]=GetDlgItemInt(hWnd,DM_Dato163,NULL,0);
                                        Medicion.dat[20][3]=GetDlgItemInt(hWnd,DM_Dato164,NULL,0);
                                        Medicion.dat[20][4]=GetDlgItemInt(hWnd,DM_Dato165,NULL,0);
                                        Medicion.dat[20][5]=GetDlgItemInt(hWnd,DM_Dato166,NULL,0);
                                        Medicion.dat[20][6]=GetDlgItemInt(hWnd,DM_Dato167,NULL,0);
                                        Medicion.dat[20][7]=GetDlgItemInt(hWnd,DM_Dato168,NULL,0);
                                        Medicion.dat[21][0]=GetDlgItemInt(hWnd,DM_Dato169,NULL,0);
                                        Medicion.dat[21][1]=GetDlgItemInt(hWnd,DM_Dato170,NULL,0);
                                        Medicion.dat[21][2]=GetDlgItemInt(hWnd,DM_Dato171,NULL,0);
                                        Medicion.dat[21][3]=GetDlgItemInt(hWnd,DM_Dato172,NULL,0);
                                        Medicion.dat[21][4]=GetDlgItemInt(hWnd,DM_Dato173,NULL,0);
                                        Medicion.dat[21][5]=GetDlgItemInt(hWnd,DM_Dato174,NULL,0);
                                        Medicion.dat[21][6]=GetDlgItemInt(hWnd,DM_Dato175,NULL,0);
                                        Medicion.dat[21][7]=GetDlgItemInt(hWnd,DM_Dato176,NULL,0);
                                        Medicion.dat[22][0]=GetDlgItemInt(hWnd,DM_Dato177,NULL,0);
                                        Medicion.dat[22][1]=GetDlgItemInt(hWnd,DM_Dato178,NULL,0);
                                        Medicion.dat[22][2]=GetDlgItemInt(hWnd,DM_Dato179,NULL,0);
                                        Medicion.dat[22][3]=GetDlgItemInt(hWnd,DM_Dato180,NULL,0);
                                        Medicion.dat[22][4]=GetDlgItemInt(hWnd,DM_Dato181,NULL,0);
                                        Medicion.dat[22][5]=GetDlgItemInt(hWnd,DM_Dato182,NULL,0);
                                        Medicion.dat[22][6]=GetDlgItemInt(hWnd,DM_Dato183,NULL,0);
                                        Medicion.dat[22][7]=GetDlgItemInt(hWnd,DM_Dato184,NULL,0);
                                        Medicion.dat[23][0]=GetDlgItemInt(hWnd,DM_Dato185,NULL,0);
                                        Medicion.dat[23][1]=GetDlgItemInt(hWnd,DM_Dato186,NULL,0);
                                        Medicion.dat[23][2]=GetDlgItemInt(hWnd,DM_Dato187,NULL,0);
```

```
Medicion.dat[23][3] = GetDlgItemInt(hWnd,DM_Dato188,NULL,0);
Medicion.dat[23][4] = GetDlgItemInt(hWnd,DM_Dato189,NULL,0);
Medicion.dat[23][5] = GetDlgItemInt(hWnd,DM_Dato190,NULL,0);
Medicion.dat[23][6] = GetDlgItemInt(hWnd,DM_Dato191,NULL,0);
Medicion.dat[23][7] = GetDlgItemInt(hWnd,DM_Dato192,NULL,0);
Medicion.dat[24][0] = GetDlgItemInt(hWnd,DM_Dato193,NULL,0);
Medicion.dat[24][1] = GetDlgItemInt(hWnd,DM_Dato194,NULL,0);
Medicion.dat[24][2] = GetDlgItemInt(hWnd,DM_Dato195,NULL,0);
Medicion.dat[24][3] = GetDlgItemInt(hWnd,DM_Dato196,NULL,0);
Medicion.dat[24][4] = GetDlgItemInt(hWnd,DM_Dato197,NULL,0);
Medicion.dat[24][5] = GetDlgItemInt(hWnd,DM_Dato198,NULL,0);
Medicion.dat[24][6] = GetDlgItemInt(hWnd,DM_Dato199,NULL,0);
Medicion.dat[24][7] = GetDlgItemInt(hWnd,DM_Dato200,NULL,0);
Medicion.dat[25][0] = GetDlgItemInt(hWnd,DM_Dato201,NULL,0);
Medicion.dat[25][1] = GetDlgItemInt(hWnd,DM_Dato202,NULL,0);
Medicion.dat[25][2] = GetDlgItemInt(hWnd,DM_Dato203,NULL,0);
Medicion.dat[25][3] = GetDlgItemInt(hWnd,DM_Dato204,NULL,0);
Medicion.dat[25][4] = GetDlgItemInt(hWnd,DM_Dato205,NULL,0);
Medicion.dat[25][5] = GetDlgItemInt(hWnd,DM_Dato206,NULL,0);
Medicion.dat[25][6] = GetDlgItemInt(hWnd,DM_Dato207,NULL,0);
Medicion.dat[25][7] = GetDlgItemInt(hWnd,DM_Dato208,NULL,0);
Medicion.dat[26][0] = GetDlgItemInt(hWnd,DM_Dato209,NULL,0);
Medicion.dat[26][1] = GetDlgItemInt(hWnd,DM_Dato210,NULL,0);
Medicion.dat[26][2] = GetDlgItemInt(hWnd,DM_Dato211,NULL,0);
Medicion.dat[26][3] = GetDlgItemInt(hWnd,DM_Dato212,NULL,0);
Medicion.dat[26][4] = GetDlgItemInt(hWnd,DM_Dato213,NULL,0);
Medicion.dat[26][5] = GetDlgItemInt(hWnd,DM_Dato214,NULL,0);
Medicion.dat[26][6] = GetDlgItemInt(hWnd,DM_Dato215,NULL,0);
Medicion.dat[26][7] = GetDlgItemInt(hWnd,DM_Dato216,NULL,0);
Medicion.dat[27][0] = GetDlgItemInt(hWnd,DM_Dato217,NULL,0);
Medicion.dat[27][1] = GetDlgItemInt(hWnd,DM_Dato218,NULL,0);
Medicion.dat[27][2] = GetDlgItemInt(hWnd,DM_Dato219,NULL,0);
Medicion.dat[27][3] = GetDlgItemInt(hWnd,DM_Dato220,NULL,0);
Medicion.dat[27][4] = GetDlgItemInt(hWnd,DM_Dato221,NULL,0);
Medicion.dat[27][5] = GetDlgItemInt(hWnd,DM_Dato222,NULL,0);
Medicion.dat[27][6] = GetDlgItemInt(hWnd,DM_Dato223,NULL,0);
Medicion.dat[27][7] = GetDlgItemInt(hWnd,DM_Dato224,NULL,0);
Medicion.dat[28][0] = GetDlgItemInt(hWnd,DM_Dato225,NULL,0);
Medicion.dat[28][1] = GetDlgItemInt(hWnd,DM_Dato226,NULL,0);
Medicion.dat[28][2] = GetDlgItemInt(hWnd,DM_Dato227,NULL,0);
Medicion.dat[28][3] = GetDlgItemInt(hWnd,DM_Dato228,NULL,0);
Medicion.dat[28][4] = GetDlgItemInt(hWnd,DM_Dato229,NULL,0);
Medicion.dat[28][5] = GetDlgItemInt(hWnd,DM_Dato230,NULL,0);
Medicion.dat[28][6] = GetDlgItemInt(hWnd,DM_Dato231,NULL,0);
Medicion.dat[28][7] = GetDlgItemInt(hWnd,DM_Dato232,NULL,0);
Medicion.dat[29][0] = GetDlgItemInt(hWnd,DM_Dato233,NULL,0);
Medicion.dat[29][1] = GetDlgItemInt(hWnd,DM_Dato234,NULL,0);
Medicion.dat[29][2] = GetDlgItemInt(hWnd,DM_Dato235,NULL,0);
Medicion.dat[29][3] = GetDlgItemInt(hWnd,DM_Dato236,NULL,0);
Medicion.dat[29][4] = GetDlgItemInt(hWnd,DM_Dato237,NULL,0);
Medicion.dat[29][5] = GetDlgItemInt(hWnd,DM_Dato238,NULL,0);
```

```
                                        Medicion.dat[29][6] = GetDlgItemInt(hWnd,DM_Dato239,NULL,0);
                                        Medicion.dat[29][7] = GetDlgItemInt(hWnd,DM_Dato240,NULL,0);
                                        EndDialog(hWnd,TRUE);
                        break;

                        case IDCANCEL:
                                        EndDialog (hWnd,FALSE);
                        break;
                        default :
                                return FALSE;
                        }
                break;
                default:
                        return FALSE;
                }
return TRUE;
}
//.....//



//.....//
//                                        DEFINICION DEL PROCEDIMIENTO NUEVO24                                        //
BOOL FAR PASCAL Nuevo24DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                return FALSE;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                                        Medicion.dat[30][0] = GetDlgItemInt(hWnd,DM_Dato241,NULL,0);
                                        Medicion.dat[30][1] = GetDlgItemInt(hWnd,DM_Dato242,NULL,0);
                                        Medicion.dat[30][2] = GetDlgItemInt(hWnd,DM_Dato243,NULL,0);
                                        Medicion.dat[30][3] = GetDlgItemInt(hWnd,DM_Dato244,NULL,0);
                                        Medicion.dat[30][4] = GetDlgItemInt(hWnd,DM_Dato245,NULL,0);
                                        Medicion.dat[30][5] = GetDlgItemInt(hWnd,DM_Dato246,NULL,0);
                                        Medicion.dat[30][6] = GetDlgItemInt(hWnd,DM_Dato247,NULL,0);
                                        Medicion.dat[30][7] = GetDlgItemInt(hWnd,DM_Dato248,NULL,0);
                                        Medicion.dat[31][0] = GetDlgItemInt(hWnd,DM_Dato249,NULL,0);
                                        Medicion.dat[31][1] = GetDlgItemInt(hWnd,DM_Dato250,NULL,0);
                                        Medicion.dat[31][2] = GetDlgItemInt(hWnd,DM_Dato251,NULL,0);
                                        Medicion.dat[31][3] = GetDlgItemInt(hWnd,DM_Dato252,NULL,0);
                                        Medicion.dat[31][4] = GetDlgItemInt(hWnd,DM_Dato253,NULL,0);
                                        Medicion.dat[31][5] = GetDlgItemInt(hWnd,DM_Dato254,NULL,0);
                                        Medicion.dat[31][6] = GetDlgItemInt(hWnd,DM_Dato255,NULL,0);
                                        Medicion.dat[31][7] = GetDlgItemInt(hWnd,DM_Dato256,NULL,0);
```

```
Medicion.dat[32][0] = GetDigItemInt(hWnd,DM_Dato257,NULL,0);
Medicion.dat[32][1] = GetDigItemInt(hWnd,DM_Dato258,NULL,0);
Medicion.dat[32][2] = GetDigItemInt(hWnd,DM_Dato259,NULL,0);
Medicion.dat[32][3] = GetDigItemInt(hWnd,DM_Dato260,NULL,0);
Medicion.dat[32][4] = GetDigItemInt(hWnd,DM_Dato261,NULL,0);
Medicion.dat[32][5] = GetDigItemInt(hWnd,DM_Dato262,NULL,0);
Medicion.dat[32][6] = GetDigItemInt(hWnd,DM_Dato263,NULL,0);
Medicion.dat[32][7] = GetDigItemInt(hWnd,DM_Dato264,NULL,0);
Medicion.dat[33][0] = GetDigItemInt(hWnd,DM_Dato265,NULL,0);
Medicion.dat[33][1] = GetDigItemInt(hWnd,DM_Dato266,NULL,0);
Medicion.dat[33][2] = GetDigItemInt(hWnd,DM_Dato267,NULL,0);
Medicion.dat[33][3] = GetDigItemInt(hWnd,DM_Dato268,NULL,0);
Medicion.dat[33][4] = GetDigItemInt(hWnd,DM_Dato269,NULL,0);
Medicion.dat[33][5] = GetDigItemInt(hWnd,DM_Dato270,NULL,0);
Medicion.dat[33][6] = GetDigItemInt(hWnd,DM_Dato271,NULL,0);
Medicion.dat[33][7] = GetDigItemInt(hWnd,DM_Dato272,NULL,0);
Medicion.dat[34][0] = GetDigItemInt(hWnd,DM_Dato273,NULL,0);
Medicion.dat[34][1] = GetDigItemInt(hWnd,DM_Dato274,NULL,0);
Medicion.dat[34][2] = GetDigItemInt(hWnd,DM_Dato275,NULL,0);
Medicion.dat[34][3] = GetDigItemInt(hWnd,DM_Dato276,NULL,0);
Medicion.dat[34][4] = GetDigItemInt(hWnd,DM_Dato277,NULL,0);
Medicion.dat[34][5] = GetDigItemInt(hWnd,DM_Dato278,NULL,0);
Medicion.dat[34][6] = GetDigItemInt(hWnd,DM_Dato279,NULL,0);
Medicion.dat[34][7] = GetDigItemInt(hWnd,DM_Dato280,NULL,0);
Medicion.dat[35][0] = GetDigItemInt(hWnd,DM_Dato281,NULL,0);
Medicion.dat[35][1] = GetDigItemInt(hWnd,DM_Dato282,NULL,0);
Medicion.dat[35][2] = GetDigItemInt(hWnd,DM_Dato283,NULL,0);
Medicion.dat[35][3] = GetDigItemInt(hWnd,DM_Dato284,NULL,0);
Medicion.dat[35][4] = GetDigItemInt(hWnd,DM_Dato285,NULL,0);
Medicion.dat[35][5] = GetDigItemInt(hWnd,DM_Dato286,NULL,0);
Medicion.dat[35][6] = GetDigItemInt(hWnd,DM_Dato287,NULL,0);
Medicion.dat[35][7] = GetDigItemInt(hWnd,DM_Dato288,NULL,0);
Medicion.dat[36][0] = GetDigItemInt(hWnd,DM_Dato289,NULL,0);
Medicion.dat[36][1] = GetDigItemInt(hWnd,DM_Dato290,NULL,0);
Medicion.dat[36][2] = GetDigItemInt(hWnd,DM_Dato291,NULL,0);
Medicion.dat[36][3] = GetDigItemInt(hWnd,DM_Dato292,NULL,0);
Medicion.dat[36][4] = GetDigItemInt(hWnd,DM_Dato293,NULL,0);
Medicion.dat[36][5] = GetDigItemInt(hWnd,DM_Dato294,NULL,0);
Medicion.dat[36][6] = GetDigItemInt(hWnd,DM_Dato295,NULL,0);
Medicion.dat[36][7] = GetDigItemInt(hWnd,DM_Dato296,NULL,0);
Medicion.dat[37][0] = GetDigItemInt(hWnd,DM_Dato297,NULL,0);
Medicion.dat[37][1] = GetDigItemInt(hWnd,DM_Dato298,NULL,0);
Medicion.dat[37][2] = GetDigItemInt(hWnd,DM_Dato299,NULL,0);
Medicion.dat[37][3] = GetDigItemInt(hWnd,DM_Dato300,NULL,0);
Medicion.dat[37][4] = GetDigItemInt(hWnd,DM_Dato301,NULL,0);
Medicion.dat[37][5] = GetDigItemInt(hWnd,DM_Dato302,NULL,0);
Medicion.dat[37][6] = GetDigItemInt(hWnd,DM_Dato303,NULL,0);
Medicion.dat[37][7] = GetDigItemInt(hWnd,DM_Dato304,NULL,0);
Medicion.dat[38][0] = GetDigItemInt(hWnd,DM_Dato305,NULL,0);
Medicion.dat[38][1] = GetDigItemInt(hWnd,DM_Dato306,NULL,0);
Medicion.dat[38][2] = GetDigItemInt(hWnd,DM_Dato307,NULL,0);
```

```
                                        Medicion.dat[38][3] = GetDlgItemInt(hWnd,DM_Dato308,NULL,0);
                                        Medicion.dat[38][4] = GetDlgItemInt(hWnd,DM_Dato309,NULL,0);
                                        Medicion.dat[38][5] = GetDlgItemInt(hWnd,DM_Dato310,NULL,0);
                                        Medicion.dat[38][6] = GetDlgItemInt(hWnd,DM_Dato311,NULL,0);
                                        Medicion.dat[38][7] = GetDlgItemInt(hWnd,DM_Dato312,NULL,0);
                                        Medicion.dat[39][0] = GetDlgItemInt(hWnd,DM_Dato313,NULL,0);
                                        Medicion.dat[39][1] = GetDlgItemInt(hWnd,DM_Dato314,NULL,0);
                                        Medicion.dat[39][2] = GetDlgItemInt(hWnd,DM_Dato315,NULL,0);
                                        Medicion.dat[39][3] = GetDlgItemInt(hWnd,DM_Dato316,NULL,0);
                                        Medicion.dat[39][4] = GetDlgItemInt(hWnd,DM_Dato317,NULL,0);
                                        Medicion.dat[39][5] = GetDlgItemInt(hWnd,DM_Dato318,NULL,0);
                                        Medicion.dat[39][6] = GetDlgItemInt(hWnd,DM_Dato319,NULL,0);
                                        Medicion.dat[39][7] = GetDlgItemInt(hWnd,DM_Dato320,NULL,0);
                                        EndDialog(hWnd,TRUE);
                        break;

                        case IDCANCEL:
                                        EndDialog (hWnd,FALSE);
                        break;
                        default :
                                return FALSE;
                        }
                break;
                default:
                        return FALSE;
                }
return TRUE;
}
// ..........//



// ..........//
//                                        DEFINICION DEL PROCEDIMIENTO NUEVO25                                        //
BOOL FAR PASCAL Nuevo25DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                return FALSE;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                                        Medicion.dat[40][0] = GetDlgItemInt(hWnd,DM_Dato321,NULL,0);
                                        Medicion.dat[40][1] = GetDlgItemInt(hWnd,DM_Dato322,NULL,0);
                                        Medicion.dat[40][2] = GetDlgItemInt(hWnd,DM_Dato323,NULL,0);
                                        Medicion.dat[40][3] = GetDlgItemInt(hWnd,DM_Dato324,NULL,0);
                                        Medicion.dat[40][4] = GetDlgItemInt(hWnd,DM_Dato325,NULL,0);
```

```
Medicion.dat[40][5] = GetDlgItemInt(hWnd,DM_Dato326,NULL,0);
Medicion.dat[40][6] = GetDlgItemInt(hWnd,DM_Dato327,NULL,0);
Medicion.dat[40][7] = GetDlgItemInt(hWnd,DM_Dato328,NULL,0);
Medicion.dat[41][0] = GetDlgItemInt(hWnd,DM_Dato329,NULL,0);
Medicion.dat[41][1] = GetDlgItemInt(hWnd,DM_Dato330,NULL,0);
Medicion.dat[41][2] = GetDlgItemInt(hWnd,DM_Dato331,NULL,0);
Medicion.dat[41][3] = GetDlgItemInt(hWnd,DM_Dato332,NULL,0);
Medicion.dat[41][4] = GetDlgItemInt(hWnd,DM_Dato333,NULL,0);
Medicion.dat[41][5] = GetDlgItemInt(hWnd,DM_Dato334,NULL,0);
Medicion.dat[41][6] = GetDlgItemInt(hWnd,DM_Dato335,NULL,0);
Medicion.dat[41][7] = GetDlgItemInt(hWnd,DM_Dato336,NULL,0);
Medicion.dat[42][0] = GetDlgItemInt(hWnd,DM_Dato337,NULL,0);
Medicion.dat[42][1] = GetDlgItemInt(hWnd,DM_Dato338,NULL,0);
Medicion.dat[42][2] = GetDlgItemInt(hWnd,DM_Dato339,NULL,0);
Medicion.dat[42][3] = GetDlgItemInt(hWnd,DM_Dato340,NULL,0);
Medicion.dat[42][4] = GetDlgItemInt(hWnd,DM_Dato341,NULL,0);
Medicion.dat[42][5] = GetDlgItemInt(hWnd,DM_Dato342,NULL,0);
Medicion.dat[42][6] = GetDlgItemInt(hWnd,DM_Dato343,NULL,0);
Medicion.dat[42][7] = GetDlgItemInt(hWnd,DM_Dato344,NULL,0);
Medicion.dat[43][0] = GetDlgItemInt(hWnd,DM_Dato345,NULL,0);
Medicion.dat[43][1] = GetDlgItemInt(hWnd,DM_Dato346,NULL,0);
Medicion.dat[43][2] = GetDlgItemInt(hWnd,DM_Dato347,NULL,0);
Medicion.dat[43][3] = GetDlgItemInt(hWnd,DM_Dato348,NULL,0);
Medicion.dat[43][4] = GetDlgItemInt(hWnd,DM_Dato349,NULL,0);
Medicion.dat[43][5] = GetDlgItemInt(hWnd,DM_Dato350,NULL,0);
Medicion.dat[43][6] = GetDlgItemInt(hWnd,DM_Dato351,NULL,0);
Medicion.dat[43][7] = GetDlgItemInt(hWnd,DM_Dato352,NULL,0);
Medicion.dat[44][0] = GetDlgItemInt(hWnd,DM_Dato353,NULL,0);
Medicion.dat[44][1] = GetDlgItemInt(hWnd,DM_Dato354,NULL,0);
Medicion.dat[44][2] = GetDlgItemInt(hWnd,DM_Dato355,NULL,0);
Medicion.dat[44][3] = GetDlgItemInt(hWnd,DM_Dato356,NULL,0);
Medicion.dat[44][4] = GetDlgItemInt(hWnd,DM_Dato357,NULL,0);
Medicion.dat[44][5] = GetDlgItemInt(hWnd,DM_Dato358,NULL,0);
Medicion.dat[44][6] = GetDlgItemInt(hWnd,DM_Dato359,NULL,0);
Medicion.dat[44][7] = GetDlgItemInt(hWnd,DM_Dato360,NULL,0);
Medicion.dat[45][0] = GetDlgItemInt(hWnd,DM_Dato361,NULL,0);
Medicion.dat[45][1] = GetDlgItemInt(hWnd,DM_Dato362,NULL,0);
Medicion.dat[45][2] = GetDlgItemInt(hWnd,DM_Dato363,NULL,0);
Medicion.dat[45][3] = GetDlgItemInt(hWnd,DM_Dato364,NULL,0);
Medicion.dat[45][4] = GetDlgItemInt(hWnd,DM_Dato365,NULL,0);
Medicion.dat[45][5] = GetDlgItemInt(hWnd,DM_Dato366,NULL,0);
Medicion.dat[45][6] = GetDlgItemInt(hWnd,DM_Dato367,NULL,0);
Medicion.dat[45][7] = GetDlgItemInt(hWnd,DM_Dato368,NULL,0);
Medicion.dat[46][0] = GetDlgItemInt(hWnd,DM_Dato369,NULL,0);
Medicion.dat[46][1] = GetDlgItemInt(hWnd,DM_Dato370,NULL,0);
Medicion.dat[46][2] = GetDlgItemInt(hWnd,DM_Dato371,NULL,0);
Medicion.dat[46][3] = GetDlgItemInt(hWnd,DM_Dato372,NULL,0);
Medicion.dat[46][4] = GetDlgItemInt(hWnd,DM_Dato373,NULL,0);
Medicion.dat[46][5] = GetDlgItemInt(hWnd,DM_Dato374,NULL,0);
Medicion.dat[46][6] = GetDlgItemInt(hWnd,DM_Dato375,NULL,0);
Medicion.dat[46][7] = GetDlgItemInt(hWnd,DM_Dato376,NULL,0);
```

```
                                        Medicion.dat[47][0] = GetDlgItemInt(hWnd,DM_Dato377,NULL,0);
                                        Medicion.dat[47][1] = GetDlgItemInt(hWnd,DM_Dato378,NULL,0);
                                        Medicion.dat[47][2] = GetDlgItemInt(hWnd,DM_Dato379,NULL,0);
                                        Medicion.dat[47][3] = GetDlgItemInt(hWnd,DM_Dato380,NULL,0);
                                        Medicion.dat[47][4] = GetDlgItemInt(hWnd,DM_Dato381,NULL,0);
                                        Medicion.dat[47][5] = GetDlgItemInt(hWnd,DM_Dato382,NULL,0);
                                        Medicion.dat[47][6] = GetDlgItemInt(hWnd,DM_Dato383,NULL,0);
                                        Medicion.dat[47][7] = GetDlgItemInt(hWnd,DM_Dato384,NULL,0);
                                        Medicion.dat[48][0] = GetDlgItemInt(hWnd,DM_Dato385,NULL,0);
                                        Medicion.dat[48][1] = GetDlgItemInt(hWnd,DM_Dato386,NULL,0);
                                        Medicion.dat[48][2] = GetDlgItemInt(hWnd,DM_Dato387,NULL,0);
                                        Medicion.dat[48][3] = GetDlgItemInt(hWnd,DM_Dato388,NULL,0);
                                        Medicion.dat[48][4] = GetDlgItemInt(hWnd,DM_Dato389,NULL,0);
                                        Medicion.dat[48][5] = GetDlgItemInt(hWnd,DM_Dato390,NULL,0);
                                        Medicion.dat[48][6] = GetDlgItemInt(hWnd,DM_Dato391,NULL,0);
                                        Medicion.dat[48][7] = GetDlgItemInt(hWnd,DM_Dato392,NULL,0);
                                        Medicion.dat[49][0] = GetDlgItemInt(hWnd,DM_Dato393,NULL,0);
                                        Medicion.dat[49][1] = GetDlgItemInt(hWnd,DM_Dato394,NULL,0);
                                        Medicion.dat[49][2] = GetDlgItemInt(hWnd,DM_Dato395,NULL,0);
                                        Medicion.dat[49][3] = GetDlgItemInt(hWnd,DM_Dato396,NULL,0);
                                        Medicion.dat[49][4] = GetDlgItemInt(hWnd,DM_Dato397,NULL,0);
                                        Medicion.dat[49][5] = GetDlgItemInt(hWnd,DM_Dato398,NULL,0);
                                        Medicion.dat[49][6] = GetDlgItemInt(hWnd,DM_Dato399,NULL,0);
                                        Medicion.dat[49][7] = GetDlgItemInt(hWnd,DM_Dato400,NULL,0);
                                        EndDialog(hWnd,TRUE);
                        break;

                        case IDCANCEL:
                                        EndDialog (hWnd,FALSE);
                        break;
                        default :
                                return FALSE;
                        }
                break;
                default:
                        return FALSE;
                }
return TRUE;
}
//.....//
```

```
//.....//
//                    DEFINICION DEL PROCEDIMIENTO GuardaComoDiaProc                    //
BOOL FAR PASCAL GuardarComoDiaProc (HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        GetDlgItemText(hWnd,DM_Como,NomArchivo,15);
                        Medicion.Guarda(NomArchivo);
                        EndDialog (hWnd,TRUE);
                break;

                case IDCANCEL:
                        EndDialog (hWnd,FALSE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//


//.....//
//                    DEFINICION DEL PROCEDIMIENTO AbrirDiaProc                         //
BOOL FAR PASCAL AbrirDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {

        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
```

```
                        GetDlgItemText(hWnd,DM_Abrir,NomArchivo,15);
                        error = Medicion.Recupera(NomArchivo);
                        EndDialog (hWnd,TRUE);
                break;

                case IDCANCEL:
                        EndDialog (hWnd,FALSE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//


//.....//
//                                DEFINICION DEL PROCEDIMIENTO AbrirErrorDiaProc                                //
//.....//
BOOL FAR PASCAL AbrirErrorDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//
```

```
//.....//
//                                   DEFINICION DEL PROCEDIMIENTO EliminarDiaProc                                   //
//.....//
BOOL FAR PASCAL EliminarDiaProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        SubGpoEliminar = GetDlgItemInt(hWnd,DM_Eliminar,NULL,0);
                        EndDialog (hWnd,TRUE);
                break;

                case IDCANCEL:
                        EndDialog (hWnd,FALSE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;

        }
return TRUE;
}
//.....//


//.....//
//                                   DEFINICION DEL PROCEDIMIENTO ErrorEliminarDiaProc                              //
BOOL FAR PASCAL ErrorEliminarDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
```

```
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//

//.....//
//                                DEFINICION DEL PROCEDIMIENTO CEPDiaProc                                //
BOOL FAR PASCAL CEPDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//
```

```
// ..........//
//                                   DEFINICION DEL PROCEDIMIENTO Error Nuevo                                   //
BOOL FAR PASCAL ErrorNuevoDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
         break;

         default :
                return FALSE;
         }
return TRUE;
}
// ..........//


// ..........//
//                                   DEFINICION DEL PROCEDIMIENTO AyudaIDiaProc                                 //
BOOL FAR PASCAL AyudaIDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
         switch (messg)
         {
         case WM_INITDIALOG:
                 break;

         case WM_COMMAND:
                 switch (LOWORD(wParam))
                 {
                 case IDOK:
                         EndDialog (hWnd,TRUE);
                 break;

                 default :
                         return FALSE;
                 }
          break;
```

```
        default :
                return FALSE;
        }
return TRUE;
}
//.....//



//.....//
//                                    DEFINICION DEL PROCEDIMIENTO Ayuda2DiaProc                                    //
BOOL FAR PASCAL Ayuda2DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK,
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//



//.....//
//                                    DEFINICION DEL PROCEDIMIENTO Ayuda3DiaProc                                    //
BOOL FAR PASCAL Ayuda3DiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;
```

```
        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//

//.....//
//                                DEFINICION DEL PROCEDIMIENTO AcercaDeDiaProc                    //
BOOL FAR PASCAL AcercaDeDiaProc(HWND hWnd, UINT messg, WPARAM wParam,LPARAM lParam)
{
        switch (messg)
        {
        case WM_INITDIALOG:
                break;

        case WM_COMMAND:
                switch (LOWORD(wParam))
                {
                case IDOK:
                        EndDialog (hWnd,TRUE);
                break;

                default :
                        return FALSE;
                }
        break;

        default :
                return FALSE;
        }
return TRUE;
}
//.....//
```

```
//.....//
//                                    DEFINICION DEL PROCEDIMIENTO WndProc                                    //
LONG FAR PASCAL WndProc(HWND hWnd, UINT messg, WPARAM wParam, LPARAM lParam)
{
HDC hdc;
PAINTSTRUCT ps;

static int xClientView, yClientView;
static FARPROC fpfnNuevo21DiaProc, fpfnNuevo22DiaProc;
static FARPROC fpfnNuevo23DiaProc, fpfnNuevo24DiaProc, fpfnNuevo25DiaProc;
static FARPROC fpfnGuardarComoDiaProc, fpfnAbrirDiaProc;
static FARPROC fpfnAbrirErrorDiaProc, fpfnEliminarDiaProc;
static FARPROC fpfnErrorEliminarDiaProc, fpfnErrorNuevoDiaProc;
static FARPROC fpfnCEPDiaProc, fpfnAyuda1DiaProc, fpfnAyuda2DiaProc;
static FARPROC fpfnAyuda3DiaProc, fpfnAcercaDeDiaProc;
static HWND hInst2, hInst3, hInst4, hInst5, hInst6, hInst7, hInst8;
static HWND hInst9, hInst10, hInst11, hInst12, hInst13, hInst14, hInst15;
static HWND hInst16, hInst17;

int j, i, s;
float LICGraX, UnidadX, UnidadY, Base;
int X1,X2,Y1,Y2,LY1,LY2;
static HBRUSH hBrush;
static HFONT hOFont, hNFont;

UnidadX = 500 / Medicion.Cantidad;
UnidadY = 120 / (1 + Diagrama.LSC - Diagrama.LC);
LICGraX = 200 + ((Diagrama.LC - Diagrama.LIC) * UnidadY);
Base = 200 + (Diagrama.LC * UnidadY);
LY1 = Base - (GraAreaSup * UnidadY);
LY2 = Base - (GraAreaInf * UnidadY);

switch (messg)
{
        case WM_SIZE:
                xClientView=LOWORD(lParam);
                yClientView=LOWORD(lParam);
                break;

        case WM_CREATE:
                hInst2=((LPCREATESTRUCT) lParam) ->hInstance;
                fpfnNuevo21DiaProc=MakeProcInstance((FARPROC)Nuevo21DiaProc,hInst2);
                hInst3=((LPCREATESTRUCT) lParam) ->hInstance;
                fpfnNuevo22DiaProc=MakeProcInstance((FARPROC)Nuevo22DiaProc,hInst3);
                hInst4=((LPCREATESTRUCT) lParam) ->hInstance;
                fpfnNuevo23DiaProc=MakeProcInstance((FARPROC)Nuevo23DiaProc,hInst4);
                hInst5=((LPCREATESTRUCT) lParam) ->hInstance;
                fpfnNuevo24DiaProc=MakeProcInstance((FARPROC)Nuevo24DiaProc,hInst5);
                hInst6=((LPCREATESTRUCT) lParam) ->hInstance;
                fpfnNuevo25DiaProc=MakeProcInstance((FARPROC)Nuevo25DiaProc,hInst6);
                hInst7=((LPCREATESTRUCT) lParam) ->hInstance;
```

```
fpfnGuardarComoDiaProc = MakeProcInstance((FARPROC)GuardarComoDiaProc,hInst7);
hInst8 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAbrirDiaProc = MakeProcInstance((FARPROC)AbrirDiaProc,hInst8);
hInst9 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAbrirErrorDiaProc = MakeProcInstance((FARPROC)AbrirErrorDiaProc,hInst9);
hInst10 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnEliminarDiaProc = MakeProcInstance((FARPROC)EliminarDiaProc,hInst10);
hInst11 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnErrorEliminarDiaProc = MakeProcInstance((FARPROC)ErrorEliminarDiaProc,hInst11);
hInst12 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnErrorNuevoDiaProc = MakeProcInstance((FARPROC)ErrorNuevoDiaProc,hInst12);
hInst13 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnCEPDiaProc = MakeProcInstance((FARPROC)CEPDiaProc,hInst13);
hInst14 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAyuda1DiaProc = MakeProcInstance((FARPROC)Ayuda1DiaProc,hInst14);
hInst15 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAyuda2DiaProc = MakeProcInstance((FARPROC)Ayuda2DiaProc,hInst15);
hInst16 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAyuda3DiaProc = MakeProcInstance((FARPROC)Ayuda3DiaProc,hInst16);
hInst17 = ((LPCREATESTRUCT) lParam) - > hInstance;
fpfnAcercaDeDiaProc = MakeProcInstance((FARPROC)AcercaDeDiaProc,hInst17);

pLogPal = (NPLOGPALETTE) LocalAlloc(LMEM_FIXED,(sizeof(LOGPALETTE) +
                    (sizeof(PALETTEENTRY) * (PALETTESIZE))));
pLogPal - > palVersion = 0x300;
pLogPal - > palNumEntries = PALETTESIZE;

//NEGRO
pLogPal - > palPalEntry[0].peRed = 0x00;
pLogPal - > palPalEntry[0].peGreen = 0x00;
pLogPal - > palPalEntry[0].peBlue = 0x00;
pLogPal - > palPalEntry[0].peFlags = (BYTE) 0;

//AZUL
pLogPal - > palPalEntry[1].peRed = 0x00;
pLogPal - > palPalEntry[1].peGreen = 0x00;
pLogPal - > palPalEntry[1].peBlue = 0xFF;
pLogPal - > palPalEntry[1].peFlags = (BYTE) 0;

//ROJO
pLogPal - > palPalEntry[2].peRed = 0xFF;
pLogPal - > palPalEntry[2].peGreen = 0x00;
pLogPal - > palPalEntry[2].peBlue = 0x00;
pLogPal - > palPalEntry[2].peFlags = (BYTE) 0;

hPal = CreatePalette(pLogPal);
break;


case WM_PAINT:
{
```

```
hdc = BeginPaint(hWnd,&ps);
SelectPalette(hdc,hPal,1);
RealizePalette(hdc);

// Activa el puerto de monitor y el modo de direccionamiento //

SetMapMode(hdc,MM_ISOTROPIC);
SetWindowExt(hdc,640,400);
SetViewportExt(hdc,xClientView,yClientView);
SetViewportOrg(hdc,0,0);

hNFont = CreateFont(12,12,0,0,700,
                    FALSE,TRUE,FALSE,OEM_CHARSET,
                    OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
                    DEFAULT_QUALITY,
                    VARIABLE_PITCH | FF_ROMAN,
                    "Roman");
hOFont = SelectObject(hdc,hNFont);
TextOut(hdc,240,15,"DIAGRAMA DE ",12);
TextOut(hdc,396,15,Titulo,1);
TextOut(hdc,210,370,"REGLA DE AT&T ",14);
TextOut(hdc,390,370,Tipo,1);
hNFont = CreateFont(9,9,0,0,400,
                    FALSE,FALSE,FALSE,OEM_CHARSET,
                    OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
                    DEFAULT_QUALITY,
                    VARIABLE_PITCH | FF_ROMAN,
                    "Reman");
hOFont = SelectObject(hdc,hNFont);
TextOut(hdc,130,400,
                "Subgrupos que rompen con la regla de corrida",44);
TextOut(hdc,60,76,"LSC",3);
TextOut(hdc,25,196,"LC",2);
TextOut(hdc,60,(LICGraX-4),"LIC",3);

// Dibuja los ejes coordenados

MoveToEx(hdc,100,50,NULL);
LineTo(hdc,100,350);
MoveToEx(hdc,50,200,NULL);
LineTo(hdc,600,200);
for (i = 0 ; i < (Medicion.Cantidad/5) ; i + + )
{
        MoveToEx(hdc,(100 + (i*5*UnidadX)),197,NULL);
        LineTo(hdc,(100 + (i*5*UnidadX)),203);
}
MoveToEx(hdc,95,80,NULL);
LineTo(hdc,105,80);
MoveToEx(hdc,95,LICGraX,NULL);
```

```
LineTo(hdc,105,LICGraX);
MoveToEx(hdc,95,LY1,NULL);
LineTo(hdc,600,LY1);
MoveToEx(hdc,95,LY2,NULL);
LineTo(hdc,600,LY2);
MoveToEx(hdc,(97+UnidadX),(Base-(ElementoGra[0]*UnidadY)),NULL);
for (i=0 ; i<Medicion.Cantidad ; i++ )
{
        X1 = 97 + (i * UnidadX) + UnidadX;
        X2 = 103 + (i * UnidadX) + UnidadX;
        Y1 = Base - (ElementoGra[i] * UnidadY);
        Y2 = Base - (ElementoGra[i] * UnidadY) + 6;
        if (Y1 > 350)
        {
                Y1 = 350;
                Y2 = 356;
        }
        if (Y1 < 50)
        {
                Y1 = 50;
                Y2 = 56;
        }
        LineTo(hdc,X1,Y1);
        s = 0;
        for (j=1 ; j < GrupoFuera[0] ; j++ )
        {
                if ( i == GrupoFuera[j] )
                {
                        s = 1;
                }
        }

        if ( s == 1 )
        {
                hBrush=CreateSolidBrush(PALETTEINDEX(2));
                SelectObject(hdc,hBrush);
        }
        else
        {
                hBrush=CreateSolidBrush(PALETTEINDEX(1));
                SelectObject(hdc,hBrush);
        }
        Rectangle (hdc,X1,Y1,X2,Y2);

}
hBrush=CreateSolidBrush(PALETTEINDEX(2));
SelectObject(hdc,hBrush);
Rectangle(hdc,115,398,125,408);

SelectObject(hdc,hOFont);
DeleteObject(hNFont);
```

```
            ValidateRect(hWnd,NULL);
            EndPaint(hWnd,&ps);
    }
    break;

    case WM_DESTROY:
            PostQuitMessage(0);
    break;

    case WM_COMMAND:
            switch (LOWORD(wParam))
            {
                    case IDM_Nuevo1:
                            DialogBox(hInst2,"Nuevo21DlgBox",hWnd,fpfnNuevo21DlaProc);
                            if ( (Medicion.Tamano > 8) | (Medicion.Cantidad > 50) )
                            {
                                    DialogBox(hInst12,"ErrorNuevoDlgBox",hWnd,

                            fpfnErrorNuevoDiaProc);
                            }
                            else
                            {
                                    Medicion.Captura();
                                    Medicion.CalX();
                                    Medicion.CalR();
                            }
                    break;

                    case IDM_Nuevo2:
                            DialogBox(hInst3,"Nuevo22DlgBox",hWnd,fpfnNuevo22DiaProc);
                            Medicion.Captura();
                            Medicion.CalX();
                            Medicion.CalR();
                    break;

                    case IDM_Nuevo3:
                            DialogBox(hInst4,"Nuevo23DlgBox",hWnd,fpfnNuevo23DiaProc);
                            Medicion.Captura();
                            Medicion.CalX();
                            Medicion.CalR();
                    break;

                    case IDM_Nuevo4:
                            DialogBox(hInst5,"Nuevo24DlgBox",hWnd,fpfnNuevo24DiaProc);
                            Medicion.Captura();
                            Medicion.CalX();
                            Medicion.CalR();
                    break;

                    case IDM_Nuevo5:
```

```
            DialogBox(hInst6,"Nuevo25DlgBox",hWnd,fpfnNuevo25DiaProc);
            Medicion.Captura();
            Medicion.CalX();
            Medicion.CalR();
break;

case IDM_Abrir:
            DialogBox(hInst8,"AbrirDlgBox",hWnd,fpfnAbrirDiaProc);
            if (error == 1)
            {
                        DialogBox(hInst9,"AbrirErrorDlgBox",hWnd,

            fpfnAbrirErrorDiaProc);
            }
            else
            {
                        Medicion.Captura();
                        Medicion.CalX();
                        Medicion.CalR();
            }
break;

case IDM_Guardar:
            Medicion.Guarda(NomArchivo);
break;

case IDM_Como:
            DialogBox(hInst7,"GuardarComoDlgBox",hWnd,

            fpfnGuardarComoDiaProc);
break;

case IDM_Salir:
            SendMessage(hWnd,WM_CLOSE,0,0L);
break;

case IDM_XRegtal:
            for (i=0 ; i<Medicion.Cantidad ; i++)
            {
                        ATT1.PuntosGra[i] = Medicion.X[i];
                        ElementoGra[i] = Medicion.X[i];
            }
            Medicion.CalMXR();
            Diagrama.LimiteControl(0,Medicion.Tamano,Medicion.MX,Medicion.MR);
            ATT1.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
            ATT1.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
            GraAreaSup = ATT1.AreaSup;
            GraAreaInf = ATT1.AreaInf;
            Titulo = "X";
            Tipo = "I";
            ATT1.CalcPuntosFuera(Medicion.Cantidad);
```

```
            for (i=0 ; i<25 ; i++)
            {
                    GrupoFuera[i] = ATT1.SubGpoFuera[i];
            }

            InvalidateRect(hWnd,NULL,TRUE);
            UpdateWindow(hWnd);
break;

case IDM_XRegla2:
            for (i=0 ; i<Medicion.Cantidad ; i++)
            {
                    ATT2.PuntosGra[i] = Medicion.X[i];
                    ElementoGra[i] = Medicion.X[i];
            }
            Medicion.CalMXR();
            Diagrama.LimiteControl(0,Medicion.Tamano,Medicion.MX,Medicion.MR);
            ATT2.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
            ATT2.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
            GraAreaSup = ATT2.AreaSup;
            GraAreaInf = ATT2.AreaInf;
            Titulo = "X";
            Tipo = "2";
            ATT2.CalcPuntosFuera(Medicion.Cantidad);
            for (i=0 ; i<25 ; i++)
            {
                    GrupoFuera[i] = ATT2.SubGpoFuera[i];
            }
            InvalidateRect(hWnd,NULL,TRUE);
            UpdateWindow(hWnd);
break;

case IDM_XRegla3:
            for (i=0 ; i<Medicion.Cantidad ; i++)
            {
                    ATT3.PuntosGra[i] = Medicion.X[i];
                    ElementoGra[i] = Medicion.X[i];
            }
            Medicion.CalMXR();
            Diagrama.LimiteControl(0,Medicion.Tamano,Medicion.MX,Medicion.MR);
            ATT3.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
            ATT3.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
            GraAreaSup = ATT3.AreaSup;
            GraAreaInf = ATT3.AreaInf;
            Titulo = "X";
            Tipo = "3";
            ATT3.CalcPuntosFuera(Medicion.Cantidad);
            for (i=0 ; i<25 ; i++)
            {
                    GrupoFuera[i] = ATT3.SubGpoFuera[i];
            }
```

```
            InvalidateRect(hWnd,NULL,TRUE);
            UpdateWindow(hWnd);
        break;

        case IDM_XRegla4:
            for (i=0 ; i<Medicion.Cantidad ; i++)
            {
                ATT4.PuntosGra[i] = Medicion.X[i];
                ElementoGra[i] = Medicion.X[i];
            }
            Medicion.CalMXR();
            Diagrama.LimiteControl(0,Medicion.Tamano,Medicion.MX,Medicion.MR);
            ATT4.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
            ATT4.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
            GraAreaSup = ATT4.AreaSup;
            GraAreaInf = ATT4.AreaInf;
            Titulo = "X";
            Tipo = "4";
            ATT4.CalcPuntosFuera(Medicion.Cantidad);
            for (i=0 ; i<25 ; i++)
            {
                GrupoFuera[i] = ATT4.SubGpoFuera[i];
            }
            InvalidateRect(hWnd,NULL,TRUE);
            UpdateWindow(hWnd);
        break;

        case IDM_RRegla1:
            for (i=0 ; i<Medicion.Cantidad ; i++)
            {
                ATT1.PuntosGra[i] = Medicion.R[i];
                ElementoGra[i] = Medicion.R[i];
            }
            Medicion.CalMXR();
            Diagrama.LimiteControl(1,Medicion.Tamano,Medicion.MX,Medicion.MR);
            ATT1.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
            ATT1.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
            GraAreaSup = ATT1.AreaSup;
            GraAreaInf = ATT1.AreaInf;
            Titulo = "R";
            Tipo = "1";
            ATT1.CalcPuntosFuera(Medicion.Cantidad);
            for (i=0 ; i<25 ; i++)
            {
                GrupoFuera[i] = ATT1.SubGpoFuera[i];
            }
            InvalidateRect(hWnd,NULL,TRUE);
            UpdateWindow(hWnd);
        break;

        case IDM_RRegla2:
```

```
                for (i=0 ; i<Medicion.Cantidad ; i++)
                {
                        ATT2.PuntosGra[i] = Medicion.R[i];
                        ElementoGra[i] = Medicion.R[i];
                }
                Medicion.CalMXR();
                Diagrama.LimiteControl(1,Medicion.Tamano,Medicion.MX,Medicion.MR);
                ATT2.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
                ATT2.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
                GraAreaSup = ATT2.AreaSup;
                GraAreaInf = ATT2.AreaInf;
                Titulo = "R";
                Tipo = "2";
                ATT2.CalcPuntosFuera(Medicion.Cantidad);
                for (i=0 ; i<25 ; i++)
                {
                        GrupoFuera[i] = ATT2.SubGpoFuera[i];
                }
                InvalidateRect(hWnd,NULL,TRUE);
                UpdateWindow(hWnd);
break;

case IDM_RRegla3:
                for (i=0 ; i<Medicion.Cantidad ; i++)
                {
                        ATT3.PuntosGra[i] = Medicion.R[i];
                        ElementoGra[i] = Medicion.R[i];
                }
                Medicion.CalMXR();
                Diagrama.LimiteControl(1,Medicion.Tamano,Medicion.MX,Medicion.MR);
                ATT3.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
                ATT3.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
                GraAreaSup = ATT3.AreaSup;
                GraAreaInf = ATT3.AreaInf;
                Titulo = "R";
                Tipo = "3";
                ATT3.CalcPuntosFuera(Medicion.Cantidad);
                for (i=0 ; i<25 ; i++)
                {
                        GrupoFuera[i] = ATT3.SubGpoFuera[i];
                }
                InvalidateRect(hWnd,NULL,TRUE);
                UpdateWindow(hWnd);
break;

case IDM_RRegla4:
                for (i=0 ; i<Medicion.Cantidad ; i++)
                {
                        ATT4.PuntosGra[i] = Medicion.R[i];
                        ElementoGra[i] = Medicion.R[i];
                }
```

```
                Medicion.CalMXR();
                Diagrama.LimiteControl(1,Medicion.Tamano,Medicion.MX,Medicion.MR);
                ATT4.CalcAreaSup(Diagrama.LC,Diagrama.LSC);
                ATT4.CalcAreaInf(Diagrama.LC,Diagrama.LIC);
                GraAreaSup = ATT4.AreaSup;
                GraAreaInf = ATT4.AreaInf;
                Titulo = "R";
                Tipo = "4";
                ATT4.CalcPuntosFuera(Medicion.Cantidad);
                for (i=0 ; i<25 ; i++)
                {
                        GrupoFuera[i] = ATT4.SubGpoFuera[i];
                }
                InvalidateRect(hWnd,NULL,TRUE);
                UpdateWindow(hWnd);
        break;

        case IDM_Eliminar:
                DialogBox(hInst10,"EliminarDlgBox",hWnd,fpfnEliminarDlaProc);
                if (SubGpoEliminar > Medicion.Cantidad)
                {
                        DialogBox(hInst11,"ErrorEliminarDlgBox",hWnd,fpfnErrorEliminarDlaProc);
                }
                else
                {
                        for (i=SubGpoEliminar ; i<Medicion.Cantidad ; i++)
                        {
                                Medicion.X[i-1] = Medicion.X[i];
                                Medicion.R[i-1] = Medicion.R[i];
                        }
                        Medicion.Cantidad--;
                }
        break;

        case IDM_CEP:
                DialogBox(hInst13,"CEPDlgBox",hWnd,fpfnCEPDlaProc);
        break;

        case IDM_AArchivo:
                DialogBox(hInst14,"Ayuda1DlgBox",hWnd,fpfnAyuda1DlaProc);
                DialogBox(hInst15,"Ayuda2DlgBox",hWnd,fpfnAyuda2DlaProc);
        break;

        case IDM_AEdicion:
                DialogBox(hInst16,"Ayuda3DlgBox",hWnd,fpfnAyuda3DlaProc);
        break;

        case IDM_Acerca:
                DialogBox(hInst17,"AcercaDeDlgBox",hWnd,fpfnAcercaDeDlaProc);
        break;
```

```
                        default:
                                break;
                }
                break;

default:
        return DefWindowProc(hWnd,messg,wParam,lParam);
}
return 0L;
}
//-------------------------------------------------------------------------------------------------------------------//
```

ANEXO 5

MANUAL DEL USUARIO.

En este anexo se describe la utilización del sistema DFC.
Se divide en seis secciones :

1.- Introducir datos nuevos.
2.- Guardar datos
3.- Abrir archivo con datos.
4.- Desplegar los diagramas de control.
5.- Eliminar subgrupo fuera de control.
6.- Ayuda.

1.- INTRODUCIR DATOS NUEVOS.

Para introducir el conjunto de valores, resultado de una selección de muestras, es necesario introducir los parámetros de : cantidad de subgrupos y tamaño de los subgrupos, dentro del diálogo **Subgrupos 1-10**. Fig. A5-2.

Es importante considerar los siguientes parámetros al momento de introducir los valores :

- 0 < Tamaño de los subgrupos < = 8
- 0 < Cantidad de subgrupos < = 50

La cantidad de valores deberá ser coherente, esto es, el total de valores es igual a :

Tamaño de los subgrupos * Cantidad de subgrupos.

Cada valor deberá introducirse en el casillero correspondiente. Para cambiar de casillero se podrá realizar a través del ratón (mouse) o desplazarse a través del tabulador.

Para introducir los valores, existen cinco cuadros de diálogos, (Fig. A5-2, A5-3, A5-4, A5-5, A5-6) cada uno de los cuales se introducen los valores de 10 subgrupos, para ello, se selecciona el diálogo a través del menú ARCHIVO-NUEVO-SUBGRUPOS ..... (Fig A5-1). Los valores se introducen en forma similar a una hoja de cálculo, es decir, una relación de columnas y renglones. Después de introducir los valores de 10 subgrupos y verificar que son correctos, se selecciona **OK** o se presiona retorno.

Es importante recordar que, para modificar los valores de determinado subgrupo, deberá introducirse los valores de los 10 subgrupos.

Si se selecciona algún cuadro de diálogo incorrecto, sólo se selecciona **CANCEL**. Ya que si se acepta el diálogo (**OK**) sin introducir valor alguno, los valores de dicho diálogo serán iguales a cero.

FIGURA A5-1 PANTALLA CON MENU DE ARCHIVO

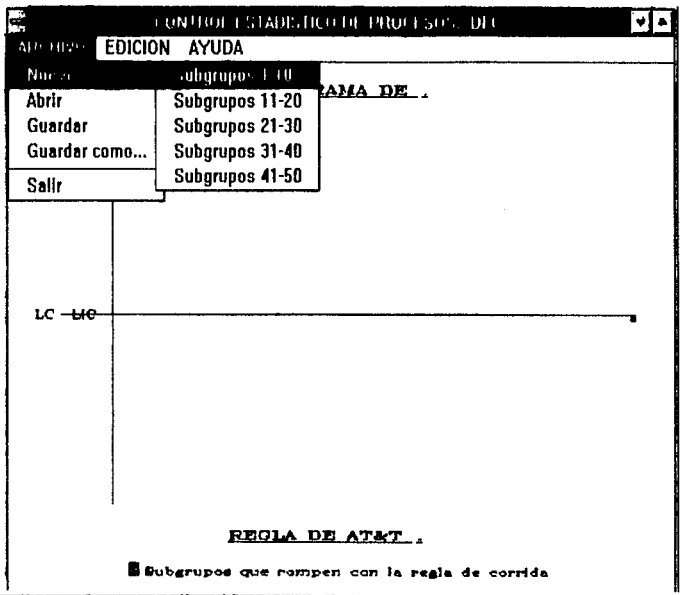


FIGURA A5-2 CUADRO DE DIÁLOGO NUEVO-SUBGRUPO 1-10

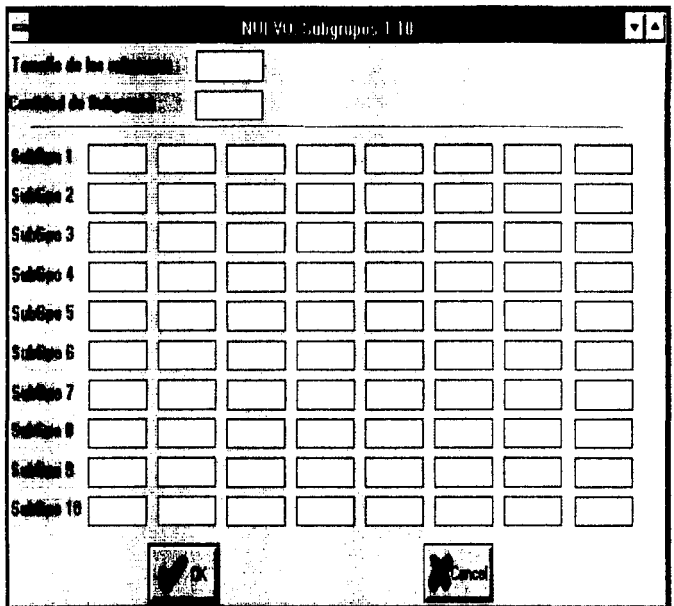


FIGURA A5-3 CUADRO DE DIÁLOGO NUEVO-SUBGRUPO 11-20

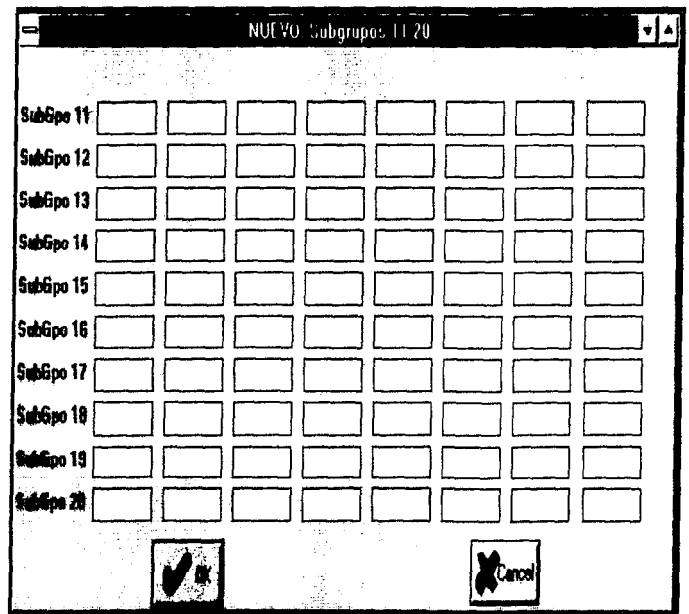


FIGURA A5-4 CUADRO DE DIÁLOGO NUEVO-SUBGRUPO 21-30

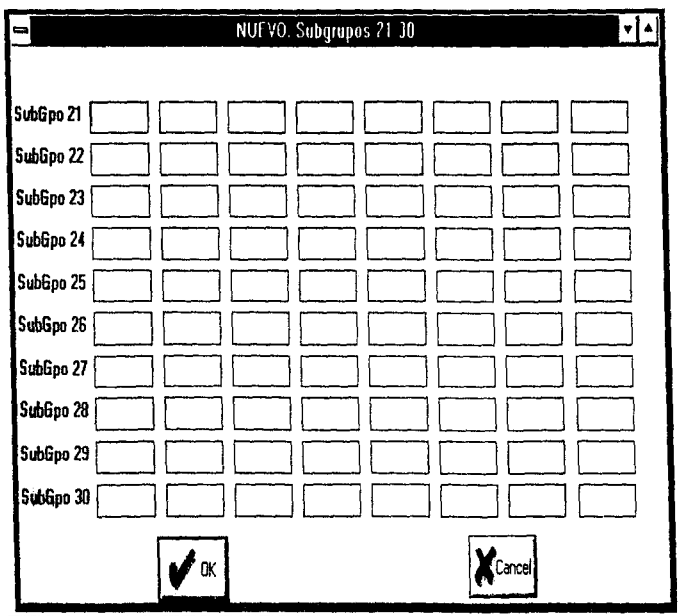


FIGURA A5-5 CUADRO DE DIÁLOGO NUEVO-SUBGRUPO 31-40

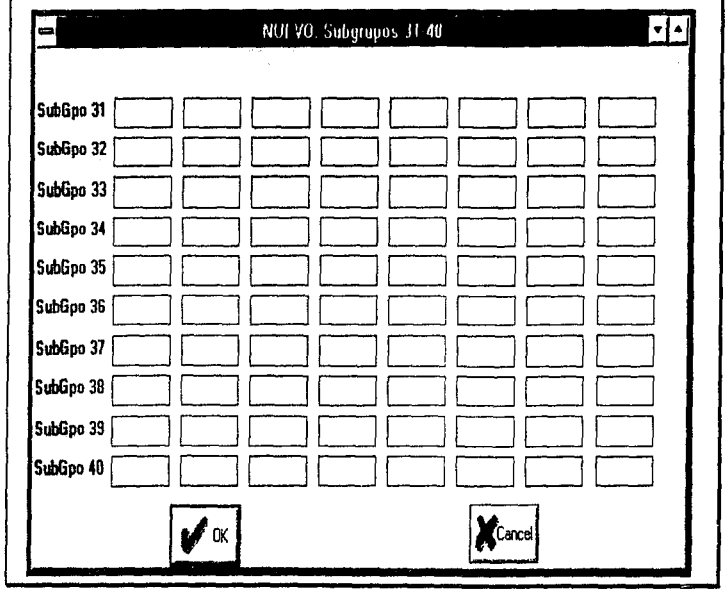
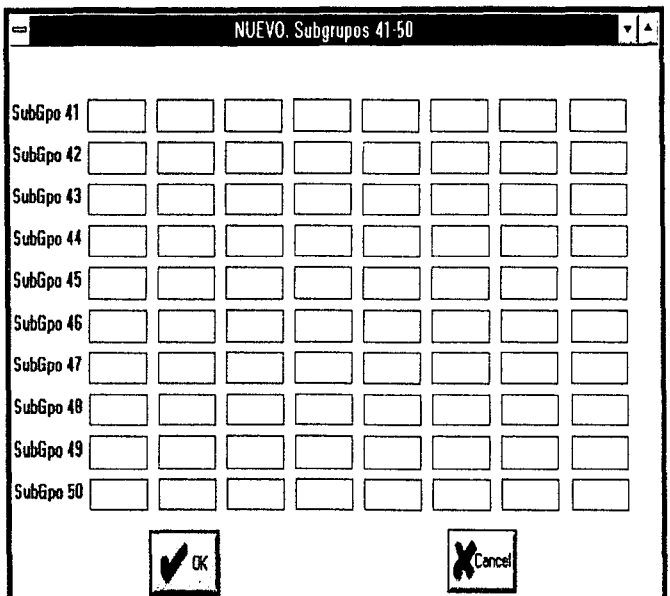


FIGURA A5-6 CUADRO DE DIÁLOGO NUEVO-SUBGRUPO 31-40



2.- GUARDAR DATOS.

Los datos previamente capturados pueden ser almacenados en disco para su posterior utilización. La realización de dicha operación puede realizarse a través, de la función Guardar y Guardar como... del menú ARCHIVO. (Fig. A5-1).

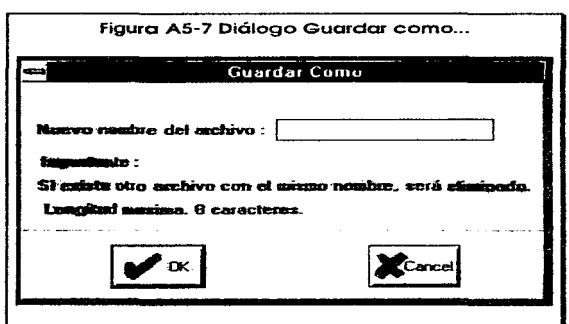
Para almacenar los datos que han sido capturados en su totalidad, esto es, datos nuevos, es necesario seleccionar la función Guardar como.... Al seleccionar dicha función aparece un diálogo con el título Guardar como (Fig A5-7). En dicho diálogo aparece un espacio en el cual se introduce el nombre del archivo. Al momento de introducir el nombre es necesario asignar la unidad en donde se desee guardar el archivo. Se recomienda asignar al nombre del archivo una extensión (**.DFC**), con la finalidad de identificar el tipo de archivo. Por ejemplo :

```
a :\nuevo.dfc
```

La función Guardar del menú ARCHIVO, almacena los datos con un nombre de archivo preestablecido, esto es, el nombre con el cual se almaceno anteriormente. En caso de utilizar esta función, con datos nuevos, se almacena con el nombre **nuevo.dfc**.

Es importante considerar que el sistema no detecta un nombre de archivo repetido. Esto es, si existe un archivo con el mismo nombre con el que se desea guardar, será eliminado automáticamente.

Figura A5-7 Diálogo Guardar como...



3.- ABRIR UN ARCHIVO CON DATOS.

Para recuperar un archivo con datos almacenados previamente, se selecciona la función Abrir del menú ARCHIVO.(Fig A5-1). Al seleccionar dicha función se abre un diálogo con título Abrir (Fig A5-8), que contiene un espacio para introducir el nombre del archivo. Y deberá considerarse lo siguiente :

- Sólo recuperará un archivo almacenado por el mismo sistema.
- Deberá especificar la unidad en donde se encuentra el archivo.
- Deberá introducir la extensión .DFC

En caso de existir algún error ya sea por : inexistencia del archivo, unidad de disco no válida, se desplegara un diálogo con el título ERROR (Fig A5-9)

FIGURA A5-8 Diálogo para abrir un archivo

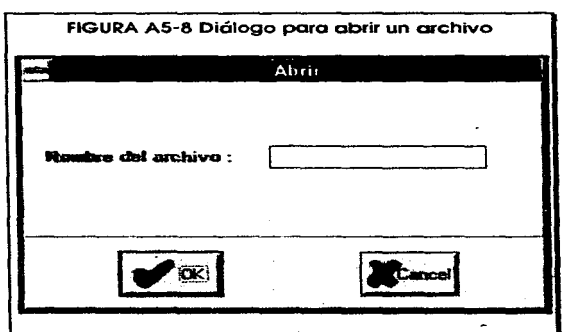
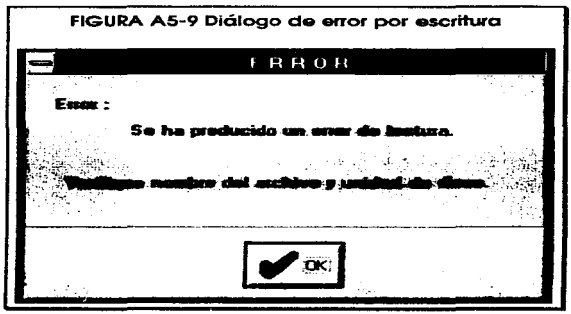


FIGURA A5-9 Diálogo de error por escritura



4.- DESPLEGAR LOS DIAGRAMAS DE CONTROL.

Una vez capturados los valores de la selección de muestras, o recuperados de un archivo capturados previamente por el sistema **DFC**, es posible visualizar los diagramas de control.

Para desplegar un diagrama de control, se selecciona el tipo de diagrama y el tipo de regla para corrida de AT&T. Mediante el menú EDICION-Diagrama X o el menú EDICION-Diagrama R. (Fig A5-10).

Después de seleccionar el tipo de diagrama de control, se despliega automáticamente el diagrama de control indicando los límites de control y puntos fuera de control correspondiente. (Fig A5-11).

FIGURA A5-10 MENU EDICION - DIAGRAMA X / DIAGRAMA R

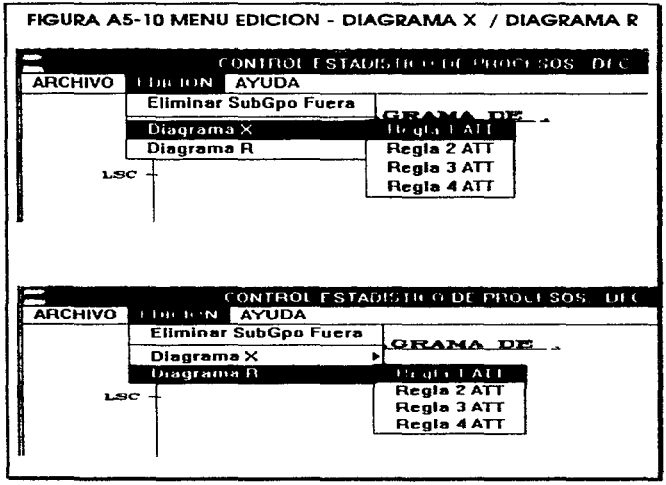
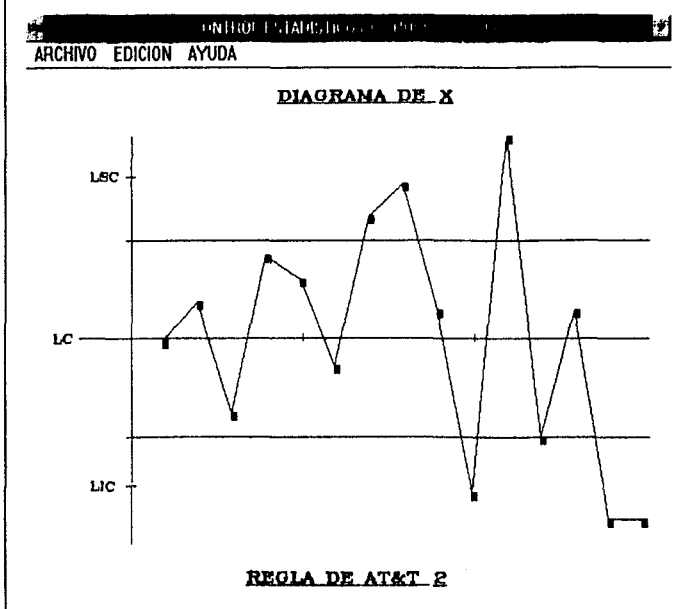


FIGURA A6-11 PANTALLA CON DIAGRAMA DE CONTROL X. REGLA DE CORRIDA AT&T 2.



5.- ELIMINAR SUBGRUPO FUERA DE CONTROL.

Después de analizar los diagramas de control para un grupo específica de datos, es posible eliminar aquellos subgrupos que rompen con la regla de control. (Puntos fuera de control). Para ello se selecciona el menú EDICION-Eliminar SubGpo Fuera (Fig A5-12) y se abre un diálogo con título Eliminación de subgrupo. (Fig A5-13), en dicho diálogo se introduce el número del subgrupo que se desea eliminar. La eliminación de los subgrupos fuera de control se realiza de manera unitaria.

Eliminados los subgrupos fuera de control, se repite el inciso 4. (Desplegar los diagramas de control), para desplegar el diagrama de control con recálculo.

FIGURA A5-12 MENU EDICION

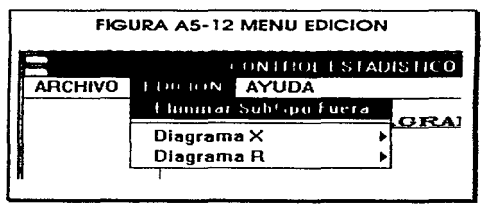
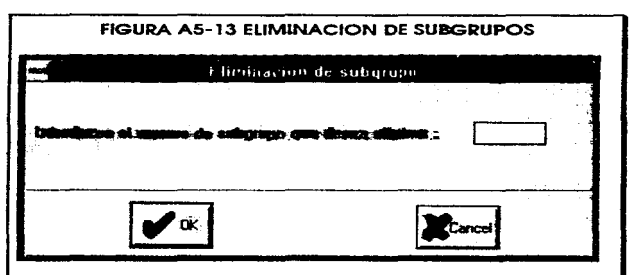


FIGURA A5-13 ELIMINACION DE SUBGRUPOS



6.- AYUDA.

El sistema **DFC** incluye un menú AYUDA (Fig A5-14) que describe el funcionamiento básico del sistema **DFC**.

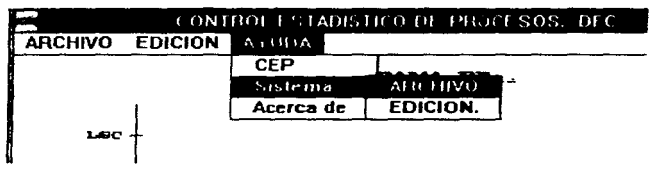
La opción CEP describe brevemente el control estadístico, el diagrama de control para X y el diagrama de control para R.

La opción Sistema-ARCHIVO describe brevemente las funciones del menú ARCHIVO.

La opción Sistema-EDICION describe brevemente las funciones del menú EDICION.

La opción Acerca de abre un diálogo con el nombre del autor e información general del sistema.

FIGURA A5-14 MENU AYUDA



BIBLIOGRAFÍA

BIBLIOGRAFIA.

(1) Booch, G., *Object-Oriented design with applications.*, The benjamin/Cummings publishing company inc, 1991.

(2) Deming, W.E., *Out of the Crisis*, MIT Press, Cambridge, MS, 1986.

(3) Faison, T., Borland C++ Object *Oriented Programming*, Sams Publishing, Indianapolis, 1994..

(4) Grant, E.L. y R.S. Leavenworth, *Statistical Quality Control*, 6/e, McGraw-Hill, New York, 1988.

(5) Hoyer, R.W. y W.C. Ellis, "A Graphical Exploration of SPC, Parts 1-2", *Quality Progress*, V.29, N.6, pp.65-73 y V.29, N.7, pp.57-64, 1996.

(6) Kiemele, M.J. y S. R. Schmidt, *Basic Statistics. Tools for Continuous Improvement*, 3/e, Air Academy Press, 1993.

(7) Nelson, L.S., "The Shewhart Control Chart -Tests for Special Causes", *Journal of Quality Technology*, V.16, N.4, pp.237-239, 1984.

(8) Pappas, C., Manual de C++ 4.0., McGraw-Hill, Madrid, 1994

(9) Rumbaugh, J., W. Prenerlani, M. Blaha, F Eddy, W Lorensen, *Object-Oriented modeling and design*, Prentice Hall, 1991.

(10) Shewhart, W.A., *Economic Control of Quality of Manufactured Product*, D. Van Nostrand Co., Princeton, NJ, 1931 (Reimpreso por la American Society for Quality Control, Milwaukee, WI, 1980).

(11) Stroustrup b., El lenguaje de programación C++, Addison-Wesley, 1991.

(12) Struebing, L. (comp.), "Quality Progress' 13th Annual QA/QC Software Directory", *Quality Progress*, V.29, N.4, pp. 31-59, 1996.

(13) Taylor, D. *Object Oriented Technology* : A manager's guide. 1990.

(14) Taylor, W.A., *Optimization & Variation Reduction in Quality*, McGraw-Hill, New York, 1992.

[15] Track D., R. Puttick, *Object technology in application development*, The Benjamin/Cummings publishing company inc, San José California, 1994.

[16] Western Electric, *Statistical Quality Control Handbook*, American Telephone and Telegraph Co., Chicago, IL, 1956.

[17] Wheeler, D.J. y D.S. Chambers, *Understanding Statistical Process Control*, 2/e, SPC Press, Knoxville, TN, 1992.