# UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

# ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON


# Sistema de Información para el Control de Procesos de Producción

**T E S I S**

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A:


CERVANTES TELLEZ FREDY

No. Cta. 9364503-8

***Asesor: Ing. Juan Gastaldi Pérez***


***Nezahualcoyotl 2005.***

# *Dedicatoria*

A mi hijo Saúl Joahan por existir.....

# *Agradecimientos*

A mi querida Universidad específicamente a la Escuela Nacional de Estudios Profesionales Aragón por permitirme haber formado parte de su comunidad escolar, lo que ha traído como consecuencia que hoy día me encuentre al final del camino para convertirme en un buen profesionista.

A mi asesor de tesis Ingeniero Juan Gastaldi Pérez por su importante labor en la obtención de mi título profesional. A mis revisores de tesis: Ingeniera Blanca Estela Cruz Luévano, Ingeniero Hugo Portilla Vázquez, Ingeniero José Antonio Ávila García e Ingeniero Adrián Paredes Romero; por su importante aportación que ha derivado en la elaboración de un trabajo de tesis con mejor calidad.

Al Ingeniero Marcelino Ochoa Martínez por haberme otorgado la oportunidad de reintegrarme a la empresa que con gran atino él dirige, bajo una particular modalidad de trabajo con la intención de concluir un importante ciclo de mi vida.

A mi padre Luis Cervantes por darme la libertad de decidir desde muy temprana edad, exigiéndome cuando lo ameritaba, lo cual hizo de mi una persona calculadora pero responsable; sobretodo por hacer un gran esfuerzo para otorgarme una educación desde la etapa más incipiente hasta la más reciente, aún en épocas muy difíciles.

A mi hermano Luis Rey que su granito de arena aportó para la elaboración de este documento.

A Juan David González Luna, mi ahora compadre, por ayudarme durante mi educación universitaria cuando yo no sabía ni programar mi vida.

A Alicia, a mis hermanos Gaby, Huero, Memo, La Flaca y Chío que en mayor o menor medida, alguna vez me apoyaron directa o indirectamente en mi proceso educacional.

En un plano ajeno a mi etapa escolar, pero fundamental en el aspecto emocional, a mi querida tía Estefana, a mi tío Simón y a mis primos Agustín, Miguel, Pedro y Álvaro quienes nos otorgaron a mi y a mis dos hermanos mayores, grandes momentos de felicidad cuando se trató de visitar ese rinconcito de nuestra provincia mexicana llamado Hidalgo, mientras los nubarrones en el horizonte pronosticaban tristes y aburridos días en aquel entonces.

Y a todos los que por cuestión de memoria no aparecen en esta página, pero que han participado en mi vida para bien.

# Contenido

# *Prefacio*

Este trabajo de tesis trata de enunciar un conjunto de herramientas y aplicación de las mismas para la interpretación de los procesos de producción, para con ello lograr la construcción de un sistema de información cuyo objetivo es aportar soluciones y mejoras al monitoreo de una cadena productiva. Y así generar información suficiente para una correcta toma de decisiones, si se requiere.

En esencia se pretendió ajustar este documento a los conceptos más básicos de la administración de la producción así como a los relativos a la ingeniería en computación enmarcados dentro de la ingeniería del software, de acuerdo a las necesidades que surgieron durante la elaboración tanto de este trabajo de tesis, como del sistema que lo complementa.

Naturalmente está sujeto a la crítica. Al integrarlo se percibió la limitación de conocimientos y experiencia, no obstante, es importante descubrir estas deficiencias para en un corto plazo subsanarlas. A pesar de ello continúa siendo un gran aporte todas y cada una de las líneas que conforman el contenido de este documento, pues la lectura del mismo ha de enriquecer el lenguaje, el manejo de las técnicas de programación, la manera de plasmar ideas en un sistema de información mediante una metodología actualizada, entre varios aspectos más.

La inquietud por abordar un tema de esta naturaleza surge de la cercanía a la cadena de ensamble de electrodomésticos de la empresa de origen alemán Bosch-Siemens Home durante un período de tiempo relativamente corto, empero, este trabajo no guarda relación o antecedente alguno a la misma. Además la compañía Desarrollos de Tecnología de Información (antes Sistemas Inteligentes Administrativos) aportó ideas y propuestas que fortalecieron la intención de abordar un tema de esta naturaleza, basada en la basta experiencia de la misma en el desarrollo e implementación de soluciones informáticas para el sector empresarial.

La estructura que guarda el presente documento está conformada por cinco temas, donde el primero de ellos comprende la conceptualización primordial de la producción, sin llegar a ser un tratado de la misma; se enuncian las metodologías empleadas tanto para la investigación e integración documental como para la etapa de análisis y diseño de sistemas; y se complementa con la teoría de UML y Bases de Datos de acuerdo a las necesidades surgidas a través del proceso que conllevó el desarrollo del sistema.

El tema dos describe las herramientas de software que se utilizaron para la creación del sistema, de tal manera que el lector adquiera un breve pero conciso conocimiento de las mismas, por lo que se pretendió ser lo más preciso posible. Se enuncia a la herramienta que permitió el modelamiento de los objetos de la base de datos, se cita a aquella en la que se programó las instrucciones requeridas para dar forma al sistema; y finalmente se aborda la teoría básica de la suite que permite la implementación y administración de los objetos de base de datos.

El tema tres tiene como propósito el análisis y diseño del sistema mediante la identificación de requerimientos generales y particulares donde varios de ellos se derivan de la correcta interpretación de un proceso general de producción. Una vez realizado lo anterior han sido plasmados en gráficos de UML los casos de uso y demás diagramas que ejemplifican las diversas actividades llevadas a cabo en una cadena productiva. Llegando a la parte final de esta

etapa con la elaboración de los prototipos de los principales programas que conforman al sistema de información.

El tema cuatro muestra la serie de elementos que resultan de la etapa de desarrollo; entre ellos se menciona y define el modelo conceptual, el diagrama entidad – relación y el diccionario de datos los cuales son parte fundamental para el almacenamiento de la información derivada de la utilización del sistema que se ha elaborado. Además se enuncian algunos de los principales procesos mediante el código fuente de los mismos.

Finalmente el tema cinco habla de los requerimientos tanto de hardware como de software, así como de los pasos que deben llevarse a cabo para la correcta implantación del sistema de información en cualquier empresa cuya actividad primordial sea la producción de bienes.

# *TEMA I: Introducción*

OBJETIVO: Enunciar lo relativo a los procesos de Producción, Metodologías y Arquitecturas para el desarrollo del Sistema.

# I.1 LA PRODUCCIÓN

## I.1.1 IMPORTANCIA DE LA PRODUCCIÓN

El Sistema a desarrollar se fundamentará en la Producción de bienes. La producción es un tema ampliamente fascinante y de actualidad, sobretodo tratándose de la producción en masa de artículos de consumo en miles de fábricas. Los productos varían desde tan prosaicos como la mercancía de máquinas hasta tan abstractos como ciertas cualidades del esparcimiento y la información. Todos son producidos por individuos, equipos, grupos y corporaciones, ya sea en casas-habitación y locales improvisados, o bien en laboratorios y fábricas. A pesar de las aparentes diferencias en cuanto a las materias primas, los procesos de obtención y los resultados finales tienen muchas semejanzas. En estas relaciones mutuas se basan todos los estudios sobre la producción que se llevan a cabo con el propósito de conservar los recursos naturales y aprovecharlos mejor.

## I.1.2 BREVE HISTORIA DE LOS ESTUDIOS SOBRE LA PRODUCCIÓN

A través de la historia los sistemas de Producción han sido fundamentales en la historia del hombre, sin embargo, nadie puede decir cuándo comenzó el hombre a estudiar la producción. Si nos basamos en las pruebas escritas, la fecha puede establecerse ya bien avanzada la historia, pero seguramente algunos de los primeros "directores" ponderaron mejores formas de producir ruedas rudimentarias, utensilios y ladrillos. Quizá los egipcios incluso tenían su propia versión del **PERT-Pyramid ERection Technique (Técnica para la Erección de Pirámides).**

En busca de la evidencia documental, debemos pasar por mencionar las maravillosas construcciones del Imperio Romano, las obras maestras del arte de la Edad Media, así como el desarrollo de los oficios en los gremios de esa época. Durante este último período la producción se caracterizó por la actividad individual y el uso de la energía muscular en lugar de la mecánica.

En los años 1700 las condiciones cambiaron rápidamente con el empleo de la energía suministrada por el vapor, la cual reemplazó a la muscular; el invento de máquinas herramientas que realizaban gran parte del trabajo manual y un sistema de educación que hacía hincapié en el intercambio de las piezas manufacturadas. Tales fueron los inicios de la revolución industrial y de muchos dolores de cabeza que aún aquejan a la dirección moderna. También aparecieron los primeros escritos sobre cómo curar estos dolores de cabeza.

Al principio del siglo XIX, las condiciones prevalecientes en una fábrica cualquiera eran deprimentes en comparación con las normas actuales. Laboraban niños de 5 a 12 años de edad en jornadas diarias de 12 a 13 horas, seis veces a la semana. El trabajo se realizaba en lugares oscuros e inseguros. Las actitudes de la dirección eran: tratar a los hombres como si fueran

máquinas, e implantar las políticas de **reducción de costos** por medio de la fuerza bruta. Aunque hubo excepciones, las guías de producción publicadas estaban orientadas principalmente hacia las mejoras físicas rudimentarias, usualmente en detrimento de la dignidad del trabajador. A pesar de esta falta de conciencia social, los conceptos sobre la producción propuestos por primera vez en la época citada incluyeron ideas tan avanzadas como la disposición de la planta en departamentos, la división de la mano de obra para el entrenamiento y el estudio del trabajo, **un flujo más ordenado de los materiales**, procedimientos mejorados para el **registro de costos** y planes de incentivo en los salarios.

Debido a diversos acontecimientos ocurridos al principio del siglo XX, se afianzaron los fundamentos de los estudios sobre la producción al hacerse más compatibles con las actitudes mecanicistas de las ciencias físicas. Los experimentos significativos que llevó a cabo Frederick W. Taylor, eran característicos del nuevo enfoque "científico". *Taylor* dirigió y analizó miles de pruebas para identificar las variables relativas a la producción. A partir de estas observaciones empíricas, diseñó métodos de trabajo en donde el hombre y la máquina eran una unidad, una unidad operante compuesta por un hombre inspirado por el incentivo del salario para dar servicio eficientemente a una máquina, de acuerdo con instrucciones exactas. Estableció la diferencia entre la planeación de actividades y su implementación y la ubicó en el área de la dirección profesional.

Los trabajos de Taylor estaban a tono con las investigaciones contemporáneas que entonces se consideraban científicas y, por lo tanto, incluyó sus conceptos en lo que llamó "dirección científica". Sus teorías recibieron tanto aclamaciones como injurias. Los críticos pronosticaron que sus puntos de vista mecanicistas apoyados por los expertos en eficiencia, deshumanizarían completamente la industria, pero otros los consideraron como la lógica aplicada a una nueva área prometedora. El que la gente haya estado o no de acuerdo con él no importa, ya que sus ideas y el fervor con que las explicaba impulsaron fuertemente la dirección industrial.

Un socio de Taylor extendió sus métodos analíticos a series de operaciones. Henry L. Gantt desarrolló métodos para establecer la secuencia de las actividades de la producción, los cuales aún hoy en día, se emplean. Con su tratamiento menos restringido de las operaciones hombre-máquina y los conceptos atractivos de organización y motivación a la teoría inicial de Taylor.

El pensamiento orientado hacia las operaciones tomó nuevo vigor de la unión entre la ingeniería y la psicología —unión que se logró tanto en el sentido literal como figurado, gracias al trabajo en equipo de los esposos Frank y Lillian Gilbreth; las actitudes mecanicistas del ingeniero Frank fueron disminuidas por las actitudes humanistas de la psicóloga Lillian. Juntos mostraron que los patrones del movimiento humano básico son comunes a muchas situaciones de trabajo diferentes. Su análisis de los micro-movimientos para mejorar las operaciones manuales iniciaron los estudios de tiempos y movimientos y el empleo de películas en el diseño del trabajo.

En la década de 1920 a 1930, las cosas se volvieron más complicadas conforme se fue reconociendo que la gente no siempre se comportaba como intuitivamente se esperaba y que las

complejidades de los nuevos procesos de producción requerían más controles. Como fue demostrado por los famosos estudios Hawthorne, el incentivo de mejores salarios o condiciones de trabajo no siempre conducía a aumentos proporcionales en la producción; factores psicológicos tales como la moral y la atención también influían. El trabajo de Walter Shewhart suministró medios de control estadístico para asegurar la precisión de piezas intercambiables requeridas por las técnicas de producción en masa iniciadas por Henry Ford. Quizás aún más importante fue que cuando se aplicaron los controles estadísticos de Shewhart se vio que se tenían que considerar todos los factores interactuantes del diseño del producto, la disposición de la planta, la capacidad del trabajador, las condiciones ambientales, los materiales y las actitudes de los clientes. Tales consideraciones naturalmente condujeron al estudio de los sistemas de toda la producción, más que de las partes aisladas.

Al principio de los cuarenta, durante la guerra, se comenzó a aplicar un enfoque interdisciplinario a los estudios de los sistemas. Los primeros en hacerlo fueron los grupos británicos de **Investigación Operacional**. Desde luego los miembros de dichos grupos no eran expertos en las áreas estudiadas, puesto que aplicaron las metodologías científicas aceptadas a problemas que nunca antes se habían sometido a tales análisis. Que los resultados hayan sido favorables no debe ser sorprendente, ya que existen analogías entre la naturaleza y los trabajos del hombre con los conceptos tomados de las ciencias físicas, al aplicarse a problemas de la dirección, de estructura semejante, produjeron un acervo de técnicas para la toma de decisiones, que aún se aprovecha en la actualidad. No obstante, su origen militar, el enfoque de la Investigación de Operaciones (o la íntimamente relacionada "ciencia de la dirección") llegó a convertirse en una base para las aplicaciones industriales.

Los años 40 también presenciaron el nacimiento de la **computadora electrónica**. En la actualidad su influencia es clara en toda la industria. Muchos empleados de oficina temen que traiga una segunda revolución industrial que, esta vez, les afectará a ellos. La dirección de nivel intermedio observa que las decisiones dependen cada vez más de las computadoras y, lo que tal vez sea peor de los controles por medio de computadora y de las decisiones automáticas. En la cúspide y en la base de la pirámide formada por la organización, el impacto ha sido leve; a menudo, sólo trajo el orgullo de tener una computadora.

A fin de esclarecer algunas nociones confusas acerca de la computadora conviene concentrarse en lo que se ha logrado y en lo que queda por hacer. Muchas técnicas matemáticas que se dan por asentadas y comprobadas, no serían factibles sin la excelsa velocidad de cálculo que caracteriza al equipo de cómputo. Por supuesto, debe lograrse que los problemas sean "programables", es decir, estructuralmente adaptables a los cálculos de la máquina. En ello estriba la función que le toca al hombre en la moderna sociedad hombre-máquina. El hombre debe recoger los datos necesarios (ayudado por máquinas que guardan los registros), reconocer el tipo de problema y el formato de su posible solución, desarrollar o seleccionar un programa apropiado, e interpretar o modificar los resultados de la máquina. En forma equivalente, las capacidades de las computadoras deben emplearse si se pretende relacionar y evaluar las muchas variables en sistemas complejos de producción. Por enfrentarse a los problemas y desafíos que ya han creado tanto el hombre, que toma las decisiones, como la computadora que le ayuda, ambos deben continuar desarrollándose a la par de ellos.

## I.1.3 LA PRODUCCIÓN Y LOS SISTEMAS DE PRODUCCIÓN

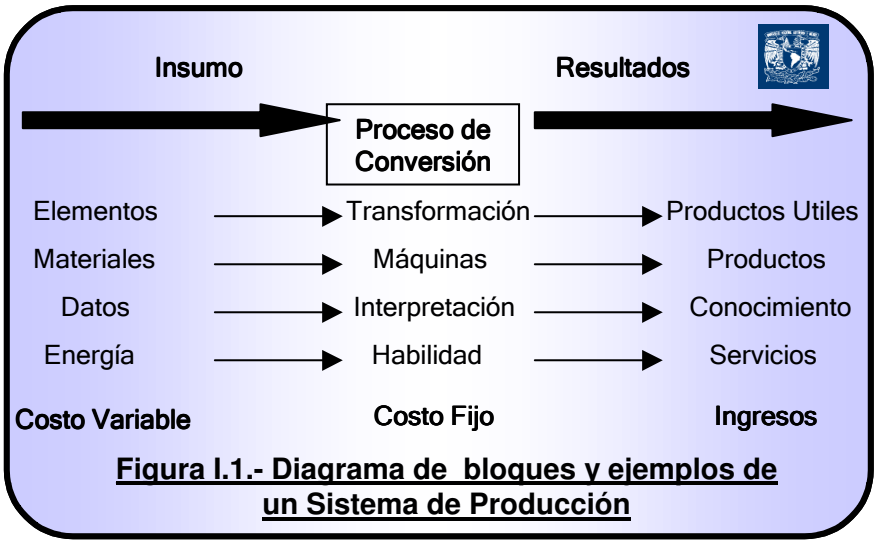



**Figura I.1.- Diagrama de bloques y ejemplos de un Sistema de Producción**

En resumen y tomando de base la historia de los sistemas de producción se puede determinar que la *producción es el acto intencional de producir algo útil.* Para incluir el concepto de sistema la definición de producción se modifica dando por resultado la siguiente definición: *un sistema de producción es el proceso específico por medio del cual los elementos se transforman en productos útiles.* Un proceso es un procedimiento organizado para lograr la conversión de insumos en resultados, como se muestra en la figura I.1.

Una unidad de producción normalmente requiere de varios tipos de insumos. En un proceso industrial los insumos dan cuenta de la mayor parte del costo variable de producción. Los medios de conversión están asociados con el costo fijo, y la producción con los ingresos. La contabilidad elemental asevera que la utilidad depende de la relación de los costos variables y fijos con respecto a los ingresos, es decir de la interacción de costos de insumo y de conversión con los ingresos obtenidos a base de la producción.

Cualquier sistema es una colección de componentes que interactúan entre sí. Cada componente podría ser un sistema en sí mismo en un orden descendente de sencillez. Los sistemas se distinguen por sus objetivos; el objetivo de un sistema podría ser producir un componente que se va a ensamblar con otros componentes para alcanzar el objetivo que es un producto final. Se requieren técnicas más elaboradas para tratar con sistemas más complejos. Esto se puede interpretar como una carrera de relevos entre el desarrollo de sistemas cada vez más complejos y el desarrollo de métodos eficientes de dirección para controlarlos. Quizá el futuro del hombre depende de quién sea el ganador, y es aquí donde una Aplicación de Software orientada al Análisis y Control de Producción, significará un importante factor adicional para determinar quién es ese ganador.

## I.1.4 LOS ANTECEDENTES DEL MOVIMIENTO DE PRODUCTIVIDAD EN MÉXICO

Antes de 1955 varias empresas del país ya empleaban diversas técnicas para elevar la productividad. Pero eran muy pocas y dispersas. Difícilmente podían influir como ejemplo sobre el resto de las empresas que, cobijadas por un mercado doméstico cautivo y exportaciones competitivas sólo porque los países entonces industrializados aún no alcanzaban su plena recuperación de la guerra, desviaban sus inversiones hacia otros negocios en lugar de vigorizar sus estructuras de producción, organización y desarrollo de recursos humanos para echar los cimientos de un sano y sólido futuro y de una efectiva competitividad. No fue sino hasta 1955 cuando se establecieron en México las bases de un movimiento organizado de productividad con pretensiones de alcance nacional. Concurrieron varias circunstancias para que se iniciara el movimiento pero, indudablemente una de las condiciones claves fue el análisis de unos industriales mexicanos lúcidos al observar lo que se estaba gestando en Europa para recuperar el poderío industrial devastado por la guerra y que, obviamente, iba a significar una importante competencia en los mercados mundiales.

En julio de 1955 se fundó el Centro Industrial de Productividad (CIP), a raíz de un convenio entre los gobiernos de México y Estados Unidos de Norteamérica. En dicho acuerdo Estados Unidos convino en suministrar a México ayuda técnica para el mejoramiento de la productividad, fundamentalmente en la industria. Por su parte, México se comprometió a crear el órgano para canalizar a la empresa mexicana dicho auxilio técnico. De esta manera nació el CIP.

Nunca llegó a efectuarse un estudio de evaluación por lo que fue difícil precisar cuales fueron los resultados cualitativos de la acción del CIP. Sin embargo, los trabajos efectuados en las empresas, el número de gerentes, técnicos y supervisores que directa o indirectamente fueron involucrados, y las acciones que continuaron en varias empresas al concluir los programas del CIP, hacen suponer fundamentalmente que fue el inicio de una conciencia de productividad en el país, que quedó un poco olvidada al desaparecer como tal el CIP, cuando concluyó el convenio entre los dos países.

# I.2 PROPUESTA DE TESIS

## I.2.1 FUNDAMENTOS

Durante el transcurso de la carrera se vieron materias tales como Sistemas de Información, Bases de Datos, Ingeniería de Programación, Programación Estructurada y Características de Lenguaje e inclusive Contabilidad y Costos, las cuales dan bastante sustento teórico para que yo disponga los conocimientos necesarios para el desarrollo de esta Tesis y del Sistema que la complementa.

En la empresa que actualmente laboro se me ha facilitado su infraestructura para el desarrollo del Software que sustenta mi Tesis. Además manejo adecuadamente la herramienta PowerBuilder, las Bases de Datos, de tal manera que ello facilitará el avance considerable que lograré para finalmente titularme.

## I.2.2 CAMPO DE ACCIÓN

Desde el punto de vista de la Ingeniería en Computación se ubica en el área relativa a la Ingeniería de Software, sin embargo, cabe hacer énfasis que se complementa de manera muy importante con la Ingeniería Industrial para interpretar adecuadamente los diversos procesos de producción.

## I.2.3 OBJETIVOS

*General.* Obtención del Título de Ingeniero en Computación a través del diseño y desarrollo del Sistema de Información para el Control de Procesos de Producción, el cual programaré en PowerBuilder (FrontEnd) y SQL-Server (BackEnd), bajo la arquitectura Cliente-Servidor además de complementarlo con la redacción que conformará este trabajo de Tesis.

*Particulares.*

- Entender la importancia de los sistemas informáticos como herramienta auxiliar en la administración de la producción.
- Desarrollar un sistema de información que permita la monitorización de los procesos productivos, mediante el manejo de Etapas y Fases de Producción.
- Que el sistema en la medida de lo posible, se ajuste fácilmente a la administración de producción de cualquier industria.
- Demostrar que mediante un análisis bien estructurado es posible generar una herramienta de software muy completa, utilizando UML y el modelo de Tres Capas.

## I.2.4 HIPÓTESIS

Es susceptible y viable la monitorización de los Procesos de Producción, ensamble o maquila a través de un Sistema de Información, logrando con ello un registro de piezas producidas e inclusive un registro del control de calidad en tiempo real. Asimismo es posible planificar la producción como los costes implicados en la misma, y por lo tanto optimizar los recursos a lo largo de toda la cadena productiva a través de hojas de trabajo o ensamble, las cuales indicarán las horas dedicadas y unidades obtenidas de un determinado producto, lo que derivará en la obtención de tiempos y costes de cada proceso, tanto humanos, como materiales.

## I.2.5 VARIABLES

- Independientes:
  - Monitorización de los Sistemas Productivos.
  - Planificar la producción como los costes implicados en la misma.
  - Horas y unidades de producto dedicadas por un trabajo implicado en el proceso de productivo de un determinado producto.
- Dependientes:
  - Registro de piezas producidas.
  - Registro del control de calidad de la producción en tiempo real.
  - Optimización de recursos a lo largo de toda la cadena productiva.
  - Tiempos y costes derivados de cada proceso, tanto humanos, como materiales.

## I.2.6 PLANTEAMIENTO DEL PROBLEMA

Analizar los sistemas de producción de manufactura debido a la pobre gestión que provoca la escasa monitorización de planta en tiempo real así como la escasa posibilidad de estructurar los datos de producción para que se integren con la toma de decisiones en planta, por lo que el seguimiento de las Ordenes de Producción depende única y exclusivamente del factor humano (desde su inicio hasta su fin), en varias empresas de la industria mexicana actual. No se mantiene un control estricto de la actividad de los recursos humanos de cada ciclo de producción

y difícilmente se comparte la información para conjuntarlos con su gestión empresarial (ERP). Por lo que se debe definir, supervisar y monitorear todo el Sistema Productivo, con una herramienta de alcance real, de tal manera que gestione toda la documentación de planta de la forma más automática posible.

Además se desconoce los tiempos, piezas y situación de cada fase de producción, esto conlleva una pérdida de tiempo para la toma de decisiones en lo que corresponde con la producción de planta.

El sistema de información a desarrollar debe permitir que el personal especializado sea capaz de resolver las siguientes preguntas mediante su utilización:

- ¿La Producción se sujeta estrictamente al Plan de Producción?
- ¿Productividad?
- ¿Qué lapsos tengo por avería, mantenimiento, holguras...?
- ¿Qué etapas del proceso resultan más tardadas?
- ¿Rechazos? ... ¿Por qué?
- ¿Puedo informar a un cliente de la situación real de su pedido?
- ¿En que etapa del proceso se encuentra la Orden de Producción?
- ¿La Merma y Desperdicios va en aumento?
- ¿El mantenimiento de las listas de componentes es el adecuado?
- ¿Cuánto, cuándo y cómo se debe producir?
- ¿Cuáles son las capacidades reales de producción?

El control y seguimiento automático de las órdenes de producción que se están ejecutando en cada una de las máquinas de una planta de fabricación en serie a partir de los datos suministrados por los departamentos de planificación, es el objetivo primordial que motiva que se desarrolle este sistema de información. Dando por resultado las respuestas a las preguntas anteriormente citadas. Las órdenes de trabajo que se han de ejecutar en una sección de fabricación y para un periodo determinado de tiempo deberán entrar en el sistema que calculará su distribución en máquinas, las necesidades de material para cada una de ellas, así como la secuencia a desarrollar. Una vez que el proceso de producción está en marcha se controlarán las incidencias, las alarmas, la realización de los test de calidad, las faltas de material y finalmente la obtención del producto terminado.

Cabe destacar que el control interno de la producción es vital para la supervivencia de toda empresa maquiladora. Donde existe DESCONTROL existe también FUGA de RECURSOS

o bien desperdicio considerable de insumos y materia prima, además de tiempos muertos. Los empleados deshonestos que, por desgracia, los hay en todas las empresas, poseen la habilidad para detectar dónde pueden meter la mano sin ser descubiertos, PRECISAMENTE POR FALTA DE CONTROLES adecuados y cuando menos se da uno cuenta, grandes cantidades de dinero, en forma de productos finales, insumos y materias y también en forma de billetes, habrá desaparecido. Por lo que, uno de los múltiples objetivos que persigue el SISTEMA DE INFORMACIÓN PARA EL CONTROL DE PROCESOS DE PRODUCCIÓN es lograr un CONTROL EFECTIVO, mediante la información que este genere para localizar las irregularidades o anomalías, y así dar los elementos para que el personal indicado plantee posibles alternativas de solución.

# I.3 MARCO TEÓRICO

## I.3.1 SOFTWARE ADMINISTRATIVO

Desde hace dos décadas la aparición paulatina de fabricantes que han apostado al desarrollo de software administrativo para la micro, pequeña y mediana empresa (MPyME) ha ido en aumento, y no es para menos, pues este sector empresarial genera 64% del empleo y 42% del PIB en México -según datos de la Secretaría de Economía-, por lo que representa un mercado muy atractivo. En la figura I.2 se visualiza un panorama general de las oportunidades de desarrollo de software en la MPyME.

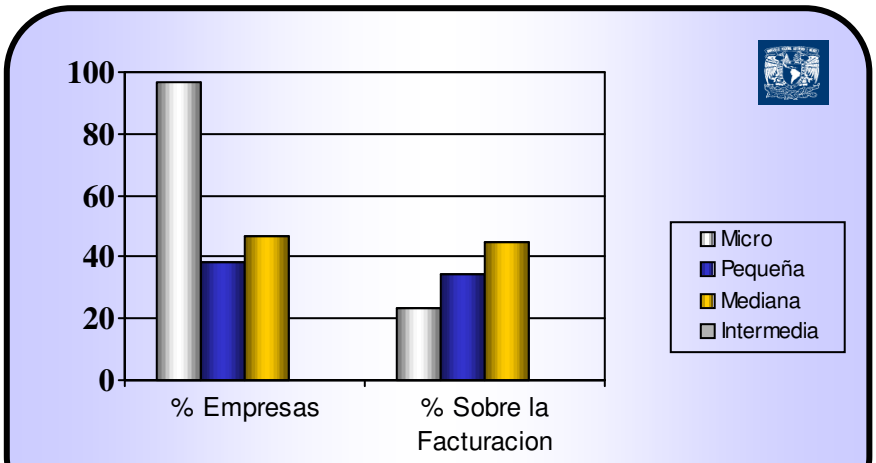



**Figura I.2.- La MPyME en Porcentajes**

Este tipo de herramientas permiten llevar un mejor control administrativo, contable, fiscal, de nómina, operaciones bancarias y punto de venta, entre otros atributos, y existen factores que podrían impulsar aún más su penetración en el mercado, representando una solución para aquellas empresas que buscan eficientar sus procesos.

Cabe mencionar que, según un estudio de la Secretaría de Economía, la edad de la mitad de los empresarios de la MPyME oscila entre 40 y 60 años, un 79% nació en área urbana y un 47% cuenta con nivel de estudios universitarios.

Los rasgos característicos destacan estructuras tradicionales con negocios familiares y altos costos de operación, bajos niveles de facturación, falta de recursos suficientes para inversión y poco acceso a esquemas de financiamiento, aunado a una falta de conocimiento técnico profundo y de capacitación en materia de TI.

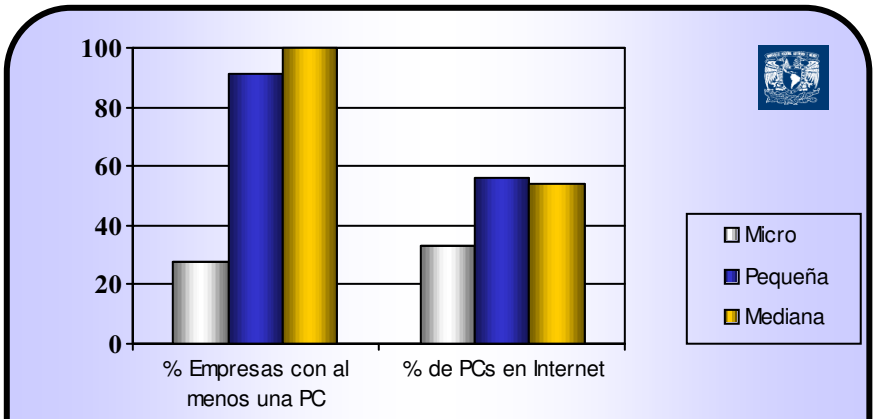



**Figura I.3.- Potencial de MPyME: PCs, Internet, la cadena**

Sin embargo, este mercado representa un negocio potencial en un momento en que la industria de TI apenas ha conquistado el 30%. Para los desarrolladores de software administrativo representa una importante oportunidad, ya que la mayor inversión hecha hasta el momento en la MPyME ha sido para infraestructura básica, por lo que aún queda pendiente migrar esta inversión inicial a soluciones más completas que respondan a las necesidades de los negocios.

En este sentido, para tener un panorama más amplio se cita la oferta en productos y esquemas comerciales de ocho empresas que desarrollan software empresarial, entre los que participan empresas con experiencia de más de 15 años y otras de reciente incursión: Sistemas Inteligentes Administrativos (SIA), Aspel, Computación en Acción (Compac), Estrategia Empresarial, Macro Pro, Microsip, Grupo SP y Vertisoft, quienes a pesar de tener productos similares, han agregado valor en distintos aspectos de sus sistemas de información.

***Sistemas Inteligentes Administrativos (SIA) (www.sia.com.mx)***

SIA es una empresa del Grupo MPS fundada en 1996. Desde su creación su principal objetivo ha sido proveer soluciones informáticas para la gestión y operación de empresas que se encuentren insertadas en la cadena de valor, es decir, compañías cuyo principal giro sea la comercialización y distribución de bienes y servicios. Esta empresa cuenta con un sistema denominado Sistema Inteligente Administrativo, el cual es una solución informática de tipo ERP (Enterprise Resource Planning) compuesta por un conjunto de módulos independientes y a su vez relacionados entre sí, lo cual permite la interacción bien planeada y estratégica de los mismos, los cuales se mencionan a continuación:

- **Catálogos**. Son los maestros para todo tipo de operación, permite el mantenimiento de la información administrativa de artículos, clientes, recursos humanos, entre otros.

- **Compras**. Permite la elaboración de Análisis de Compras, Generación de Ordenes de Compras, y más.

- **Almacén y Tráficos**. Control General de Inventarios, ubicación física de artículos, recepción de mercancía, Manejo de Entregas, Asignación de Mensajerías.

- **Ventas**. Análisis de Ventas, Manejo de BackOrders, Registro de Ventas perdidas, Facturación y embarques automáticos.

- **Finanzas**. Cuentas por Pagar, Cuentas por Cobrar, Tesorería.

- **Garantías**. Registro de Ordenes de Servicio, Cotizaciones de Servicio de Garantías, Administración de Centros de Servicio.

- **Contabilidad**. Registro de Pólizas Contables Manuales, Semiautomáticas y Automáticas, Catálogo de Cuentas.

- **Soporte Técnico**. Base de conocimientos para la resolución de problemas de diversa índole, así como el control de solución de casos.

- **Templates (Enciclopedia).** Generación de Hoja de Características de productos por departamento o clasificación, Manejo de información mercadológica con imágenes.

***Aspel (www.aspel.com.mx)***

Esta empresa tiene más de 15 años en el mercado y es de capital 100% mexicano. De manera general, la oferta de Aspel está compuesta por:

- **COI**. Es un sistema de contabilidad integral para Windows que captura, procesa y mantiene actualizada la información contable y fiscal de la empresa, permitiendo calcular también la depreciación y reevaluación de activos fijos.

- **SAE**. Sistema de administración de ventas que permite controlar el ciclo de todas las operaciones de compra-venta de la organización.

- **NOI**. Nómina integral que permite el control de percepciones, deducciones y declaraciones de los trabajadores, tomando en cuenta los requisitos de la legislación fiscal y laboral.

- **Banco**. Para el control de cualquier tipo de cuenta bancaria y todos sus movimientos, ya sea en moneda nacional o extranjera.

- **PROD**. Sistema de control de producción que permite la planeación y control de los procesos de fabricación, facilitando la administración de costos.

- **Caja**. Es un sistema de punto de venta para comercializar los productos y servicios de una empresa, que promete un estricto control de sus inventarios e ingresos y controles adicionales de la operación, como el cálculo de comisiones de sus vendedores.

- **S@E Comercio electrónico.** Traslada al ambiente de Internet los catálogos de inventarios, clientes y políticas comerciales registrados en SAE, construyendo una tienda virtual en unos cuantos minutos para exhibir y promocionar los productos y servicios de una compañía.

Una de las principales estrategias de ASPEL es apoyar el desarrollo del *e-commerce* y expandir sus productos habilitados para esta actividad, sobre todo si se toma en cuenta que su nicho de mercado cada día cuenta con mayores recursos de tecnología. Además, la empresa está reforzando su esquema comercial, con el que promueve entre los usuarios la adquisición de herramientas administrativas sin tener que comprar el sistema completo con tan sólo agregar el número de usuarios o empresas que necesite, pagando sólo por ello; por otro lado, permite la renta de sistemas como SAE, NOI y Banco, con lo que gana una comisión por el arrendamiento. Aspel cuenta con varios centros de servicio que dan apoyo directo al canal y que están ubicados en el DF, Guerrero, Jalisco, Coahuila, Chihuahua, Morelos, Quintana Roo, Sinaloa y Tamaulipas, además de Guatemala, El Salvador y Honduras. Actualmente la firma trabaja con los mayoristas Ingram Micro, Synnex, Centel, Maps, Calcom, Versión y Mexmal.

***Computación en Acción (Compac) (www.compac.com.mx)***

Con casi 20 años en el mercado, esta firma tapatía presentó recientemente su nueva visión como empresa y su modelo comercial de valor para el canal. Entre lo más destacable se encuentra el trabajo conjunto para que sus distribuidores, más que vender cajas, se conviertan en consultores de negocios para sus clientes y les entreguen la solución adecuada. Por ello, recientemente firmó una alianza con el Centro Regional para la Competitividad Empresarial (Crece), por medio del cual 250 de sus asociados recibirán una consultoría gratuita en una primera etapa para detectar los puntos débiles de sus organizaciones y mejorarlos, con lo cual podrán dar mejor servicio al cliente. Asimismo, Compac ha lanzado un portal (**www.contaqtame.com.mx**) para dar soporte a sus asociados con información técnica y de negocios, mediante alianzas con terceros, como el despacho Mancera, Ernest & Young.

Por el tiempo que tiene en el mercado ha desarrollado una relación estrecha con empresas que fungen como mayoristas locales, como Limac Asesoría, pero también tiene relación con Ingram Micro, Synnex y Mexmal.

Se pretende que la empresa estará dando entrenamientos bimestrales a su canal con el fin de mejorar sus habilidades y atender mejor nichos donde detecta que sus productos tienen buena aceptación. Cabe comentar que Compac mantiene un trabajo estrecho con la Business Software Alliance (BSA) y la Alianza contra la Piratería para defender la propiedad intelectual. Además, durante el 2002 fue reconocida por el CONACYT debido a su labor en el desarrollo de investigación y tecnología en el país. De manera general, su oferta está compuesta por:

- **ContPAQ**. Programa contable, financiero y fiscal que permite al usuario dominar tasas, movimientos y manejos de IVA y de cualquier impuesto hasta de 999 empresas. Además, controla e interpreta todos los gastos, costos y presupuestos de cada área del negocio.

- **AdminPAQ**. Permite administrar el proceso comercial, pues controla a detalle las operaciones de compras, ventas, inventarios, clientes y proveedores. Además maneja múltiples almacenes y monedas. Realiza reportes a la medida con su hoja electrónica.

- **NomiPAQ**. Calcula aguinaldos, primas vacacionales y finiquitos. Se pueden realizar cédulas de trabajo en Excel para hacer reportes a la medida, así como verificar paso a paso los cálculos del ISPT e IMSS mediante una hoja de cálculos.

- **CheqPAQ**. Facilita el control del flujo de efectivo, pues manipula y explota la información con su hoja electrónica; permite conocer el periodo de causación del IVA por medio del módulo de conciliación bancaria y también facilita llevar el control de las cuentas de cheques.

- **Exión**. Se trata de un sistema de administración de operaciones de compra-venta que puede trabajar sobre Excel en equipos de escritorio y *notebooks*, así como su versión Exión Móvil para *handhelds*.

***Macro Pro (www.macropro.com.mx)***

Nació en 1992 en Monterrey y cuenta con una oficina comercial en Culiacán y otra en la Ciudad de México, así como una fábrica de software en Sinaloa. Su catálogo está compuesto por productos dirigidos a empresas de cualquier giro y otras con mercados específicos:

- **Macro Pro Integral y MacroPro Comercial**. Van al nicho de ventas a menudeo, como comercializadoras, distribuidores y empresas de servicio. Integra contabilidad, ventas, nómina, compras, inventarios y punto de venta. La única diferencia es que la segunda no incluye nómina.

- **Macro Pro Fábrica**. Es Macro Pro Integral más módulos de producción, que controlan la administración de la producción.

- **Macro Pro Talleres**. Dirigida a ese tipo de establecimientos, consiste en el paquete Macro Pro más un módulo de servicio. Aplica a todo lo que involucra una orden de trabajo en talleres como los automotrices o de equipo de cómputo.

- **Macro Pro Autos**. Enfocado a agencias o empresas de compra y venta de autos; es el módulo integral, más otro de compra y venta para autos nuevos y usados.

- **Macro Pro Lite**. Orientado a microempresa, tiene módulo de compras, ventas e inventarios. Es muy básico y se adapta a cualquier micronegocio.

***Microsip (www.microsip.com.mx)***

Tuvo su origen en el Norte del país hace 16 años, pero cuenta con oficinas representativas en 27 ciudades de la República Mexicana. La familia de sus productos incluye nueve módulos: compras, inventarios, ventas, cuentas por pagar, punto de venta, cuentas por cobrar, bancos, contabilidad y nómina.

Actualmente los distribuidores que forman el canal de distribución de Microsip se clasifican en tres tipos:

- **Certificado**. Empresas con personal certificado para implantar y dar soporte a los nueve módulos que forman la familia de productos. Su actividad primordial es la consultoría y guardan una estrecha relación con el representante de la marca en su localidad.

- **Autorizado**. Combina la venta de software con la comercialización de computadoras, periféricos, consumibles y mantenimiento de equipos. Cuenta con personal capacitado, mas no certificado.

- **Referenciado**. Es un contacto que recomienda el uso de sistemas pero su actividad profesional es otra; se trata de despachos de contadores, comercios de artículos de oficina, etc.

Actualmente la empresa mantiene una alianza con OpenSys, empresa especializada en Linux y uno de los distribuidores de Suse. Gracias a esta unión el software de Microsip se habilitó para trabajar bajo servidores Linux, aprovechando las ventajas que el código abierto ofrece. Además, se encuentra en tratos con otras empresas para lanzar este año aplicaciones de cómputo móvil y CRM integradas a sus módulos. Tiene planes de incrementar el número de ciudades con representantes y lograr un crecimiento superior al 20% en el presente año.

***Sistemas Estratégicos***

Esta empresa tiene una línea llamada Control 2000, la cual abarca software administrativo, contable y fiscal. Sus productos son:

- **Contafiscal 2000 (CF2)**. Corre en Windows y está diseñado para llevar la administración de los procesos contables y fiscales de MPyMEs de manera sencilla, incluyendo las últimas reformas fiscales. Permite el cálculo fiscal del ISR, IMPAC, IVA, deducción de activos fijos y otros.

- **Bancos 2000 (B2)**. Lleva el control de los movimientos bancarios -como abonos, cargos, cheques e ingresos- y cuenta con cuatro tipos de clasificaciones: concepto, subconcepto, centro de costo y categoría. Dentro de las características más importantes se encuentra la capacidad de realizar filtros y ordenamientos múltiples a la información presentada.

- **Nómina 2000 (N2)**. Es un sistema multiempresa que permite la consulta o modificación de cualquier periodo procesado, con la actualización de acumulados. Genera las pólizas

de la nómina y gastos patronales, con opción de ligarse a Contafiscal 2000 para Windows. Hace cálculos y declaraciones de ISR, 2% impuesto estatal, impuesto sustitutivo, IMSS, SAR y SUA.

- **Administrador 2000 (A2)**. Ayuda a la empresa a controlar y agilizar sus procesos y operaciones de compra-venta y se compone de cinco módulos: facturación, inventarios, cuentas por cobrar, compras y cuentas por pagar. Es un sistema multiempresa, multiusuario, multialmacén y multimoneda. Además, contiene un reporteador que permite al usuario crear diversos tipos de reportes según sus necesidades e incluye catálogos en línea, filtro de catálogos y claves alfanuméricas.

Asimismo, Estrategia Empresarial lanzará un producto para punto de venta que podría estar listo en cuatro meses más. Para este año, la empresa espera crecer 35%.

***Grupo SP (www.sp.com.mx)***

Su origen es español y nació en 1982. Aun cuando no está en su mercado nativo, el trabajo que ha realizado en los últimos tres años en México le ha valido una posición en el mercado, tras dar pelea a competidores con más experiencia. Para mayo y septiembre se esperan nuevas versiones de ContaPlus, AdminPlus y TPVPlus. De manera general, la oferta de SP es la siguiente:

- **ContaPlus**. Entre sus bondades destacan el ser multiempresa, monopuesto y multiusuario; enlaza con AdminPlus Profesional y TPVPlus Profesional; genera nuevos tipos de ISR, tiene módulo de cheques y para análisis de impuestos; permite llevar contabilidad presupuestaria y por departamento, además de contener gráficos y enlace a Excel.

- **AdminPlus**. Trabaja con Windows 95 o superior, calcula impuestos en recibos, controla ubicaciones de artículos, máximos y mínimos, multialmacenes, así como la administración de compras mediante pedidos, remisiones y facturas de proveedores.

- **TPVPlus**. Se puede enlazar con ContaPlus Profesional, crea hasta diez propiedades definibles del producto, controla dispositivos de punto de venta, maneja gestor de tiendas, genera ventas por *tickets* con la herramienta de traslado de *ticket* a factura.

- **TiendaPlus**. Genera cheques cancelados, conciliaciones bancarias, es multidivisa, genera estados de situación financiera, estado de resultados y balanza de comprobación, saldos promedio diarios y mensuales, contiene gráficos y enlaces a Excel, entro otras características.

Cabe mencionar que estos productos tienen una versión Elite y una Profesional, que permiten que el cliente cubra sus necesidades de acuerdo a su complejidad.

***Vertisoft (www.vertisoft.com.mx)***

Esta empresa mexicana es de las más nuevas en el mercado de software administrativo, pues nació hace dos años, pero hasta ahora es que presenta un esquema de canal con el que espera integrar 30 VARs a lo largo del país en los siguientes cuatro meses. Su oferta está constituida por cuatro productos:

- **Súper Tienda**. Está dirigido a microempresas con necesidades muy básicas. El CD con el programa se da de forma gratuita para usarse durante dos meses, después de los cuales el usuario tiene la opción de comprarlo. Este producto administra distintos módulos como: banco, cliente, proveedor, punto de venta, orden de compra, facturación, reporteador, por mencionar algunos, pero no incluye nómina y contabilidad. Cuenta con dos interfases: una en Windows y otra más sencilla con comandos que ayudan a quienes no tienen mucha experiencia usando computadoras. Además, existe una alianza con NCR, pues en conjunto con sus modelos de cajas registradoras 2030 y 2000 se incluye este software para dar comprobantes fiscales.

- **Dasta**. Es un ERP que permite organizar todo tipo de tienda y almacenes por Internet. Permite llevar el control de información de artículos, proveedores, tiendas, almacén, cuentas, conceptos, bancos, mercadotecnia, así como sus diferentes departamentos.

- **Control y Administración de Inmuebles (CADI)**. Permite administrar todo tipo de inmuebles, llevando así un mejor control de sus áreas o departamentos. Este producto permite al cliente realizar cargos y abonos a nivel de local, áreas o plazas; control en la administración de catálogos de clientes, proveedores, bancos, diseño de contratos y diversos programas.

- **Sistema Administrativo para la Industria del Vestir (SAIV)**. Permite controlar toda la cadena de suministro de empresas de ese sector.

**Conocer para decidir.** Como se observa, la oferta es variada y cada fabricante ha desarrollado lo que considera como la mejor estrategia de software administrativo para MPyME, una vez que se acerque y conozca con detalle los productos y beneficios comerciales que cada empresa desarrolladora de Software ofrece. Por su parte, los fabricantes tienen la responsabilidad de evitar las guerras de precios -con las que al final pierden, al igual que sus asociados-, para conducir la lucha más por el valor de sus propuestas.

## I.3.2 TIPOS DE PRODUCCIÓN

### I.3.2.1 Producción por montaje.

Entre las industrias que trabajan por montaje se cuentan algunas de las actividades productivas de mayor relevancia para la economía actual, principalmente las mecánicas: automóviles, motores, tractores, electrodomésticos, electrónicos, etc. La producción por montaje se caracteriza por encadenar secuencias de procesos que convergen hacia una línea continua en la que se ensamblan los productos finales. Pero su primera parte agrupa operaciones de mecanizado en un sinnúmero de piezas, las que tradicionalmente han sido elaboradas en talleres manejados bajo una típica modalidad intermitente, ya sea en la propia empresa o por proveedores fuertemente vinculados a ella.

Tal organización de la producción ha originado a este tipo de industrias no pocas dificultades para el planeamiento, la programación y el control, desde que FORD estableciera los nuevos patrones de operación que reemplazaron al viejo esquema artesanal. La propia naturaleza del producto hace que este se vaya ensamblando en sucesivas etapas que convergen hacia un tronco principal: la línea de montaje final. Se configuran así verdaderas redes en las que cada punto de unión es alimentado por algunos o muchos componentes, dando lugar a una estructura con tiempos asociados. Para programar es necesario recorrer la red en sentido inverso, desde el producto hasta los orígenes, a fin de determinar que piezas fabricar y cuando hacerlo, teniendo en cuenta los problemas que esto puede llegar a acarrear.

La explosión del producto en sus partes componentes se suele representar mediante el gráfico o diagrama de Gozinto, el cual consiste en estructura matricial de la lista de materiales requeridos para el ensamble o fabricación de un producto terminado, de tal manera que resulte fácil identficar sus cantidades totales para cada nivel de componentes.

Hasta alrededor de 1960 en los países más industrializados y hasta avanzada la década de los 70 en los restantes, la programación de la producción por montaje se hacía mediante ficheros que eran atendidos por verdaderos ejércitos de empleados, donde cada fichero representaba una pieza, componente, subensamble o ensamble. Establecido un plan de producción los responsables de las fichas correspondientes a los productos finales calculaban los requerimientos de componentes y productos necesarios para fabricarlos y los comunicaban a los encargados de las fichas respectivas, y así se seguía de unos a otros, a través de la red, hasta llegar a las primeras piezas, que solían venir de proveedores. A estos se les solía comunicar un plan de requerimientos, donde generalmente los tres primeros meses eran tomados como en firme y los siguientes tres como una estimación. Como es obvio, la forma de programar era lenta, rígida, y terreno fértil para toda clase de errores. Se fueron desarrollando así prácticas tendientes a mejorar los programas, a acumular grandes inventarios de partes y las industrias de montaje adquirieron un perfil paquidérmico, tan antieconómico como inflexible. Este panorama cambió radicalmente para la industria del montaje con dos adelantos fundamentales que nacieron y se desarrollaron en las décadas del 60 y el 70: el método MRP y la producción justo a tiempo.

*Variaciones estacionales.* Estas variaciones afectan disminuyendo la demanda de algunos productos debido a cambios estacionales:

EJEMPLOS:

- Las fábricas de ventiladores disminuyen o anulan su producción en invierno, ya que este producto es inútil en esta temporada, la aumentan entrando en primavera o en verano cuando, aumenta la demanda de los ventiladores.

- Otro caso sería también el de los calefactores quienes sufrirían un caso parecido al anterior pero de manera inversa, ya que sería en invierno donde aumente la demanda de estos productos.

- Los cambios estacionales también afectan a la producción de comestibles, un ejemplo sería el de algunas galletas bañadas en chocolate, que se dejan de producir en verano debido a que el chocolate se derrite por el calor veraniego. Luego al empezar el otoño reanudan su producción.

*Control de Costos.* No cabe duda de que la mejor manera de efectuar el control cuantitativo y el costeo de la producción por montaje es a través del sistema MRP. Entre sus salidas se suelen contar informes para estos propósitos. En tanto sus archivos albergan la información acerca de lo programado y pueden ser además alimentados con los datos reales, es posible efectuar los controles de cumplimiento y eficiencia como un subproducto del mismo sistema. Por otra parte, los datos de costos de materiales, mano de obra y costos variables y fijos, permiten no solo obtener el costo por producto, sino también variaciones de cantidad de insumos y precios de costo.

Mercado Cambiante. Cuando hay mucha producción de un producto, la demanda de este disminuye. Esto produce una sobresaturación de mercadería. Se encuentran dos soluciones posibles para este problema:

- Disminuir la producción hasta que vuelva aumentar la demanda de este producto, lo que no parece factible ya que si la demanda disminuyó es poco factible que vuelva a aumentar.

- Cambiar, y producir otro producto más novedoso y así generar una nueva demanda.

### I.3.2.2 Planeamiento de la producción sobre pedido.

Un Planeamiento de la Producción sobre pedido o proyecto consiste en un conjunto de actividades de producción que:

- Tiene una identidad propia, es decir que cada producto -sea bien físico o servicio- presenta rasgos característicos distintivos con respecto a los restantes elaborados por el mismo productor; más aún, muy frecuentemente puede ser único.

- Se trata de obras de apreciable magnitud y/o importancia.

- Configura una red compleja de tareas vinculadas entre si a través de múltiples interrelaciones de precedencia.

- Su duración suele prolongarse en el tiempo (aún cuando existen diferencias considerables entre un caso y el otro) y presenta momentos o hitos definidos que marcan su comienzo y su conclusión y las instancias inmediatas de su desarrollo.

Son ejemplos típicos de proyectos, entre otros:

- La construcción de edificios, plantas industriales, caminos, puentes, diques, etc.

- La construcción de grandes buques.

- El desarrollo e implementación de sistemas computarizados.

- El desarrollo de trabajos de consultoría, habitualmente conformados por el diagnóstico de problemas organizacionales y la puesta en marcha de las recomendaciones emergentes.

- La producción de películas.

En los proyectos se presentan tres instancias sucesivas:

- En primer lugar, la decisión de realizar el proyecto, que se basa en la aprobación de un presupuesto presentado por un proveedor (en tal caso, el productor) y/o un formulario de inversión interno en la organización. En esta etapa se definen globalmente las características del proyecto, su secuencia, plazos, costos, erogaciones a efectuar y la rentabilidad o beneficios esperados (esto es, su justificación económica). Esta primera etapa es conocida comúnmente como análisis y evaluación de la inversión.

- En el segundo paso se caracteriza por la especificación pormenorizada de los trabajos a efectuar, la interrelación de los mismos, los recursos a aplicar (materiales, mano de obra, equipos, etc.), un cálculo de costos más preciso que la estimación original, el cronograma definitivo en base al cual se habrá de trabajar y el desarrollo financiero que se deriva de su realización. Todo esto se denomina *ingeniería de detalle.*

La ejecución, en la que se lleva a cabo el proyecto, emitiéndose generalmente -para el control y costeo- ordenes de producción o de trabajo (como en la producción intermitente), y

controlándose el cumplimiento de la cronología prevista, generalmente mediante el empleo de gráficos.

Los métodos y técnicas utilizables para el planeamiento y programación de proyectos son variados:

- Métodos financieros de evaluación de proyectos de inversión.
- Gráficos de Gozinto y archivos de despiece, para analizar y diseñar la estructura del producto.
- Método MRP.
- Ordenes de trabajo.
- Métodos gráficos de programación, como el de Gantt y el PERT -siglas que provienen de: Program Evaluation And Review Technique = Técnica para la Evaluación y Revisión del Programa - o método del camino crítico.

*Variaciones estacionales.* Un buen ejemplo para mostrar las variaciones estacionales es la pavimentación de caminos, ya que este proceso debe ser parado cuando las condiciones climáticas no permiten seguir con el proceso de pavimentación.

*Control de Costos.* La producción por proyectos presenta en la práctica los siguientes problemas:

- Lo más común es que, entre una y otra etapa, se produzcan desvíos en exceso tanto en los montos a erogar como en los tiempos previstos, llegándose a multiplicar los originalmente presupuestados. A veces de acuerdo con las cláusulas contractuales establecidas, eso puede desencadenar sanciones financieras para la firma responsable de ejecutar el proyecto, por lo que, cuando se trata de empresas que son oferentes habituales en el ramo, tratan de maniobrar para no verse perjudicadas y aun sacar provecho de la situación. De esta forma, los proyectos suelen terminar siendo algo muy distinto (desde el punto de vista económico-financiero, de su duración y de los beneficios esperados) de lo que en el inicio iban a ser.

- No hay un software (como el MRP para la producción por montaje) que permita desarrollar, controlar y costear adecuadamente y en forma completa los proyectos, y sobretodo dar seguimiento paso a paso de una a otra etapa para evitar los desvíos referidos. La bibliografía, en este punto, se remite al método PERT como la herramienta por excelencia, pero, aun reconociendo su utilidad, no cabe duda

de que no resulta una solución integral al problema referido. El valor del PERT, de los sistemas computarizados que lo aplican combinándolo con el GANTT, así como el empleo de órdenes para el control de trabajos específicos y, en ciertos casos, del MRP, constituyen elementos valiosos pero insuficientes para manejar un fenómeno como el de los proyectos, con tan aguda propensión a desvíos en tiempo y costo. La mejor forma de resolver este problema consiste en preparar cuadros comparativos de control mediante Excel, en los que se cotejen (tanto en costo como en plazo) los principales rubros y sub-rubros del proyecto etapa contra etapa, es decir: la ingeniería de detalle. Este control debe efectuarse con la debida frecuencia, paso a paso, teniendo especialmente en cuenta los compromisos que se contraen, pues, en caso contrario, se enfrentaran hechos consumados, verdaderas *autopsias* donde poco restara por hacer. La interrelación de estas planillas con los restantes elementos de control (tales como gráficos PERT, ordenes y familias de ordenes, MRP, etcétera) posibilitará una integración de las funcionalidades de estos con la visión global que proporcionan las planillas que resumen el control de avance, que así pasarán a erigirse en el tablero de comando que sintetiza la marcha del proyecto.

*Mercado cambiante.* En un proceso productivo, como la constitución de una red informática en una empresa es probable que durante el proceso aparezca un nuevo insumo o elemento que pueda hacer más efectiva a la red, se debe parar la producción, proyectar el agregado del nuevo elemento y luego continuar con lo proyectado.

### I.3.2.3 Modalidades de la producción de proceso continuo.

Son modalidades de la producción continua que condicionan substancialmente su planeamiento y control:

- Produce grandes volúmenes.
- Su orientación es hacia el producto, tanto desde el punto de vista del diseño de la planta, como por el hecho de que la cantidad elaborada de cada producto es muy elevada con relación a la variedad de productos.
- Cada producto es procesado a través de un método idéntico o casi idéntico.
- Los equipos son dispuestos en línea, con algunas excepciones en las etapas iniciales de preparación de los materiales. El ruteo es el mismo para cada producto procesado.
- Es de capital intensivo, por lo que el planeamiento del uso de la capacidad instalada resulta prioritario. Como es frecuente que se trabajen tres turnos durante los siete días de

la semana, se torna imposible, en tales casos, recurrir al tiempo extra cuando la demanda exige una mayor producción.

- Consecuentemente, el grado de mecanización y automatización es alto.
- Los inventarios predominantes son los de materias primas y productos elaborados, dado que los de material en proceso suelen ser mínimos.
- El planeamiento y control de la producción se basan, en gran medida, en información relativa al uso de la capacidad instalada y el flujo de los materiales de un sector a otro.
- A menudo se obtienen coproductos y subproductos, que generan complicaciones para el planeamiento, el control y el costeo.
- Las actividades logísticas de mantenimiento de planta y distribución física del producto adquieren una importancia decisiva.

Entre las industrias que se caracterizan por operar en forma continua se cuentan las que elaboran productos tales como: bebidas embotelladas, celulosa, papel, azúcar, aceite, acero, envases, etc. Dentro de un esquema conceptual de esta naturaleza, al tamaño de las corridas o lotes varía de períodos cortos hasta una operación absolutamente continua. Cabe distinguir entonces dos subtipos básicos dentro de este tipo de producción, que no dependen tanto del ramo de actividad de que se trate sino de la variedad de productos que elabore la empresa:

- *Ultracontinua:* En esta solo es necesario determinar las cantidades a producir y los insumos para períodos prolongados, por lo que carecen de relieve la programación y el lanzamiento. Desde el punto de vista del planeamiento y control de la producción, es la más sencilla.
- *Continua por lotes:* En caso de producirse por *lotes*, el tamaño de estos y su secuencia obligan al uso de algún modelo de programación que optimice tales aspectos, además de tener en cuenta las complicaciones que puedan presentarse en cada circunstancia particular.

Los modelos de planeamiento y programación mas utilizados son:

- El presupuesto, lisa y llanamente.
- La programación lineal.
- La simulación mediante computadora.
- Modelos específicos desarrollados para ciertas industrias o empresas.

*Control cuantitativo y costeo.* El control cuantitativo y el costeo en la producción continua se realizan por procesos. Debido a ello, reviste decisiva importancia la adecuada definición de los centros o módulos de control y costos (los que se corresponden con los procesos del sistema). Estos centros pueden ser: Productivos, de Servicios (generadores de electricidad, vapor, aire comprimido, etcétera), de Almacenaje (de materias primas, producción en proceso y/o productos terminados).

### I.3.2.4 Producción para Stock.

La producción intermitente comúnmente se lleva a cabo en talleres. A pesar de desarrollarse en unidades productivas de reducido tamaño, presenta un grado de complejidad y dificultades que se derivan de sus propias características.

En efecto, en ella se reciben frecuentes pedidos de los clientes que dan lugar a órdenes de producción o trabajo. Estas son generalmente de variada índole y se complementan con los recursos disponibles, que a veces resultan insuficientes y otras veces quedan en gran medida ociosos. Aún más, es común que ciertas estaciones se encuentren abarrotadas y otras con muy poca labor. Cobra especial significado la preparación o alistamiento de la maquinaria para pasar de una producción a la siguiente.

Cada pedido suele requerir una programación individual y soluciones puntuales a los problemas que trae aparejados.

Las características más destacadas de la producción intermitente son:

- Muchas órdenes de producción derivadas de los pedidos de los clientes.
- Gran diversidad de productos.
- Dificultades para pronosticar o anticipar la demanda.
- Trabajos distintos uno del otro.
- Agrupamiento de las máquinas similares en el taller.
- Necesidad de programar cada caso en particular.
- Bajo volúmen de producción por producto.
- Emisión de órdenes específicas para cada pedido.
- Mano de obra calificada.
- Necesidad de contar con recursos flexibles.

Los conceptos precedentes se refieren básicamente a la industria. Pero la producción Intermitente también se presenta en los servicios. En algunos de ellos, como en un taller de reparación de automóviles, por ejemplo. En otros, ofreciendo una visión bastante distinta en apariencia, como es el caso de un restaurante, aunque con bastantes similitudes en los aspectos esenciales de la producción. Si bien en las industrias intermitentes suelen hacerse planes anuales divididos en meses, a medida que se los va ejecutando es importante corregirlos con los datos de los pedidos anticipados. Esta dinámica hace que sea la instancia de programación a la que se asigna mayor importancia en este tipo de producción.

La programación se orienta en función de:

- Cumplimiento de plazos de los pedidos.
- Minimización de la inversión en instalaciones.
- Estabilidad de la fuerza de trabajo.
- Máximo nivel de producción.
- Atención de prioridades: grandes clientes, urgencias, etcétera.
- Flexibilidad, en general.
- Confiabilidad de los procesos críticos.
- Reserva de capacidad para pedidos especiales o urgentes.
- Minimización de los costos de producción.
- Cálculo preciso de costos para presupuestar los trabajos.
- Utilización a pleno de las fuerzas de trabajo.
- Minimización de horas extras.
- Lapso mínimo de fabricación.
- Adquisición de materiales en forma oportuna y económica.

*Control de Costos.* El control de costos de la producción de Stock se realiza por órdenes. A ellas se cargan todos los insumos aplicados para ejecutarlas. Si el costeo se efectúa por computadora, algunos cargos habrán de provenir de otro sistema; por ejemplo: el de materiales. Otros, en cambio, tendrán que ser recopilados a través de informes o partes, como ocurre comúnmente con la mano de obra. Esto resulta un problema. Este problema, hace que resulte conveniente contar con tiempos estandarizados -o, cuanto menos, predeterminados- a fin de

poder cerrar el circulo de control de la productividad y la eficiencia. En efecto, si bien estos estándares se cotejan en tiempos reales que suelen ser los declarados en los partes por los mismos involucrados, debe tenerse en cuenta que:

- Si estos aumentasen los tiempos declarados para mostrar una mayor productividad, deteriorarían la eficiencia.

- Si, por el contrario, reflejasen en los partes tiempos muy ajustados con el fin de encuadrarse dentro de los estándares, la productividad resultante seria baja.

El costeo es efectuado, obteniéndose informes -detallados por concepto de gasto- por orden, cada una de las cuales corresponde, generalmente, a un producto específico.

*Variaciones estacionales y mercado cambiante (tanto como para producción continua como para producción para stock).* Incidencia de las variaciones estacionales y los cambios de mercado en la producción continua y la producción para stock:

En ambos tipos de producción el planeamiento se orienta principalmente a buscar el modo de atender la demanda con la capacidad disponible o bien a compatibilizar ambas mediante la aplicación de métodos apropiados. Si bien las variaciones estacionales inciden en todas las actividades productivas, y deben ser consideradas en la instancia del planeamiento de la producción compatibilizando la oferta con la demanda, en la producción continua reviste mayor importancia debido a la mayor rigidez de su estructura y su consiguiente dependencia de la capacidad instalada. Para suavizar la incidencia de las variaciones estacionales en un proceso de producción continua se puede actuar ya sea sobre la demanda o ya sea sobre la oferta, siempre teniendo en cuenta los costos relevantes que derivan de la acción que se tome, los que deberán ser valorados y evaluados a fin de tomar la decisión mas óptima y oportuna.

Se puede mencionar como acciones que se pueden tomar desde el punto de vista de la demanda a las siguientes: Diferir o desechar pedidos, establecer políticas de precios o tarifas diferenciales que alienten o desalienten al consumo de los bienes o servicios ofrecidos, desarrollar productos complementarios para aprovechar las instalaciones en los periodos de baja demanda.

Respecto a la oferta: Ajustar la fuerza laboral a través de: Modificaciones en la dotación, Utilización de horas extras o de otro turno, Empleo de personal eventual, Reprogramación de periodos vacacionales, Rotación de la mano de obra en los distintos sectores productivos, Acumular o desacumular inventarios (productos elaborados), Subcontratar trabajo (tercerización), Diferir los mantenimientos productivos de maquinas e instalaciones.

Frecuentemente se toman dos o más de estas acciones en forma simultánea. La elección de estos cursos de acción enumerados se basa generalmente en un criterio económico, ya que juegan los costos derivados de ellos, los cuales deberán ser evaluados y comparados mes a mes, para que de esa manera se determine las cantidades a producir. Estos costos son:

- Costos de trabajo de horas extras.
- Costo de mantener inventario (productos elaborados inmovilizados) sobre todo los de alto valor agregado.
- Costo de personal eventual.
- Costo de subcontratar trabajo.
- Costos de oportunidad, como dejar de atender pedidos o perder clientes.

# I.4 METODOLOGÍAS DE DESARROLLO

## I.4.1 METODOLOGIA PARA EL DESARROLLO DE LA INVESTIGACION

Generalmente los tipos de estudio o investigación se dividen en tres: exploratorios, descriptivos, correlacionales y explicativos (Clasificación de Dankhe). Donde los estudios exploratorios sirven para familiarizarse con fenómenos relativamente desconocidos. Los estudios descriptivos permiten analizar cómo es y como se manifiesta un fenómeno y sus componentes. Los correlacionales pretenden ver cómo se relacionan o vinculan diversos fenómenos entre sí, o si no se relacionan. Finalmente, los explicativos buscan encontrar las razones o causas que provocan ciertos fenómenos.

Este trabajo de Tesis abarca el tipo de investigación exploratoria al describir la situación histórica y actual de los procesos de producción, y se sustenta fundamentalmente en el estudio descriptivo al llevarse a cabo una investigación bibliográfica y de campo para conocer las etapas de producción de ciertas empresas, con la intención de que el sistema se abarque en gran medida, los distintos modos de fabricación investigados.

## I.4.2 METODOLOGIA PARA EL DESARROLLO DE SOFTWARE

Una metodología es un conjunto ordenado de pasos o etapas para la obtención de un fin. Existen diversas metodologías para el desarrollo de sistemas, entre las cuales se puede mencionar MERISE, SSADM y YOURDON. Dado que se ha determinado utilizar la metodología GRAPPLE, será la única que se defina a continuación. Cabe aclarar que esta elección se debió a que dicha metodología se auxilia en demasía de UML y dado que este lenguaje de modelamiento está siendo requerido a últimas fechas en el desarrollo de sistemas informáticos dentro del ámbito laboral, resultó interesante el adentrarse en estos dos conceptos: UML y GRAPPLE (RAD3). Además, tiene cierta similitud a la metodología Yourdon (actualmente la más utilizada y por ende satisfactoriamente probada), pero con cierta tendencia a la Programación Orientada a Objetos.

### I.4.2.1 El Método Antiguo

Se caracteriza porque las etapas se deben sucederse una después de otra. De hecho las primeras metodologías de desarrollo se estructuraban de esa manera. La figura I.4 muestra un ejemplo del método antiguo el podemos llamarlo el método de “Cascada”, el cual está conformado por el análisis, diseño, codificación y distribución, tales etapas van una después de la otra, es decir sólo y sólo si la etapa anterior ha concluido, la etapa actual puede iniciar.

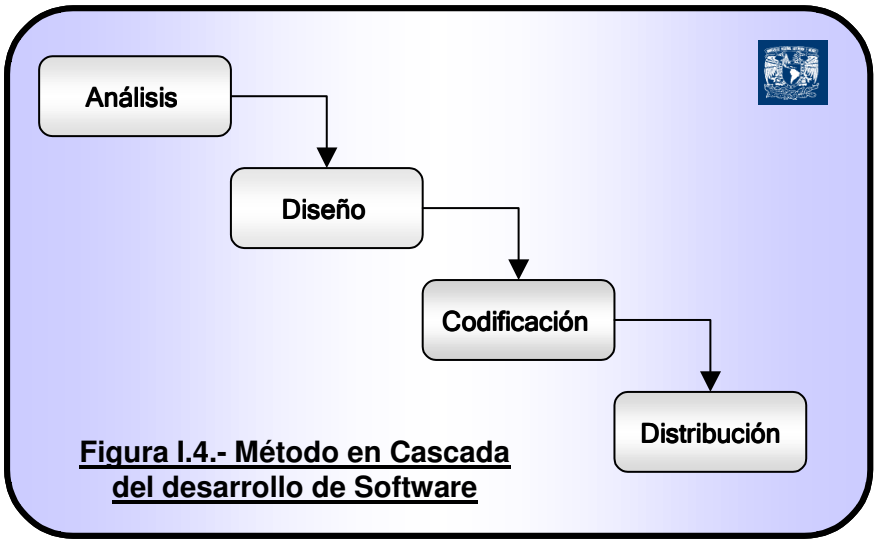



**Figura I.4.- Método en Cascada del desarrollo de Software**

Esta metodología tiene algunos inconvenientes. Por un lado, tiende a la realización de tareas individuales. Si el analista no tiene contacto con el diseñador, y este a su vez no tiene contacto con el desarrollador, ocasionalmente estos tres miembros trabajarán juntos para compartir puntos de vista, fundamentales para generar adecuadamente el sistema que cubra las necesidades de los futuros usuarios. Otro problema con este método es que reduce el impacto de la comprensión obtenida del proyecto. (Es común, que ha medida que avanza la elaboración del sistema, la comprensión e interpretación de la logística evolucione –aún cuando se haya realizado un análisis exhaustivo que garantizaba un diseño perfectamente definido-). Si el proceso no puede retroceder y volver a ver los primeros estados o etapas, es posible que las ideas desarrolladas no sean utilizadas. Intentar introducir con calzador nuevos puntos de vista durante el desarrollo es, cuando menos, bastante difícil. La revisión de un análisis y diseño –y la ulterior incorporación de una idea desarrollada- establece una mayor oportunidad de éxito.

### I.4.2.2 El Método Reciente

En contraste con el método de Cascada, la moderna ingeniería de programas tiende a la colaboración entre las fases de desarrollo. Los analistas y diseñadores hacen revisiones para desarrollar un sólido fundamento para los desarrolladores. Los cuales a su vez, interactúan con los analistas y los diseñadores para compartirles sus puntos de vista, modificar los diseños y fortalecer su código. La ventaja es que conforme crece la comprensión, el equipo incorpora nuevas ideas y genera un sistema más confiable. A pesar de ello se puede presentar una desventaja la cual consiste en que algunas personas no son muy participativas y pueden mantenerse al margen. Ahora bien, cabe aclarar que es importante no establecer barreras artificiales entre las fases ya que podría dar por resultado un sistema que no haga exactamente lo que los clientes desean. El método antiguo trae consigo otro problema: es común el caso de que el método "en cascada" implique repartir el tiempo del proyecto en la codificación. El verdadero efecto de esto es que se quita un tiempo valioso al análisis y diseño. En la figura I.5 se visualiza gráficamente lo que significa este método.

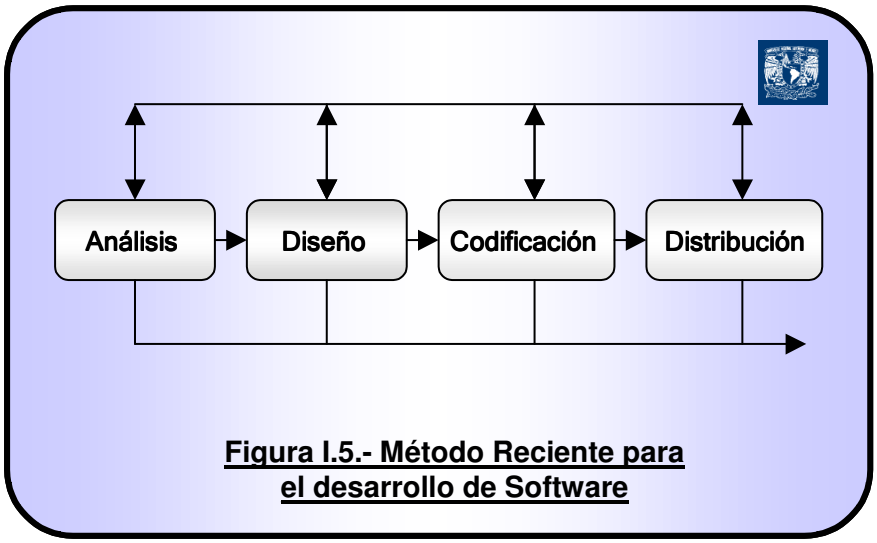



**Figura I.5.- Método Reciente para el desarrollo de Software**

### I.4.3 LO QUE DEBE HACER UN PROCESO DE DESARROLLO

En los primeros años de la programación de computadoras, una persona podía analizar un problema, otorgar una solución y escribir un programa. Actualmente es distinto ya que para desarrollar las diferentes naturalezas de sistemas complejos que demandan los negocios de hoy, es más necesario el uso de un equipo. El conocimiento se ha especializado tanto que una persona no puede conocer todas las facetas de un negocio, comprender un problema, diseñar una solución, traducirla a un programa, distribuir el programa en el hardware y asegurarse que los componentes del hardware funcionen de forma conjunta.

El equipo tiene que estar conformado por analistas para comunicarse con el cliente y comprender el problema, diseñadores para generar una solución, programadores para codificarla e ingenieros de sistemas para distribuirla. Un proceso de desarrollo tiene que tomar en cuenta todos los procesos anteriores, utilizarlos adecuadamente y asignar la cantidad de tiempo necesaria para cada fase. El proceso también debe dar por resultado diversos productos del trabajo que den indicios de progreso y conformar una estela de responsabilidad. Finalmente, el proceso deberá asegurar que sus fases no sean discontinuas. En lugar de ello, debe obtenerse información entre las fases para fomentar la creatividad y aumentar la facilidad de innovar.

Al llegar a un proceso, existe la tentación de generar una serie de fases que podrían traer una gran cantidad de papeleo. Algunas metodologías que están disponibles de forma comercial lo hacen, esto conlleva que los gerentes de proyectos llenen interminables formularios. El papeleo se complica por sí mismo. Una razón de esto es la idea errónea de que la metodología de la "unitalla" es posible; cada empresa es única. Una empresa tiene su propia cultura, normatividad, historia y personal. La metodología del desarrollo que pueda aplicarse a un consorcio internacional posiblemente fallará en una pequeña empresa y viceversa. Al intentar

meter con calzador una metodología en una empresa, se tendrá la mala impresión de que un papeleo extremo podrá ayudar.

Lo mencionado anteriormente da por resultado que un equipo de desarrollo deberá cumplir con lo siguiente:

• Asegurar que tal equipo cuente con una firme comprensión del problema que se intenta resolver.

• Poner las condiciones para que un equipo conste de una colección de responsabilidades.

• Fomentar la comunicación entre los miembros del equipo que ostentan tales responsabilidades.

• Facilitar la intercomunicación entre las fases del proceso de desarrollo.

• Desarrollar productos de trabajo que comuniquen el progreso al cliente, y eliminar el papeleo superfluo. Y lo más importante, que el proceso origine un producto terminado en un lapso corto.

Una metodología de desarrollo estructura los segmentos y actividades en un proyecto de desarrollo de sistemas. Sin una metodología habría un caos y los desarrolladores no comprenderían el problema que se supone deberían resolver, así como los sistemas no cumplirían con las necesidades de los usuarios. Las metodologías de antaño forzaban a una secuencia "en cascada" de análisis, diseño, codificación y distribución. Este tipo de metodología secuencial podía fragmentar el desarrollo, de modo que un equipo de desarrollo podría no aprovechar la mejor asimilación que se obtiene durante la vida de un proyecto. Por lo general, también distribuye la mayor parte del tiempo en la codificación, y esto resta una enorme cantidad de tiempo al análisis y diseño.

### I.4.4 GRAPPLE (DIRECTIVAS PARA EL RÁPIDO DISEÑO DE APLICACIONES)

Es un patrón simplificado de un proceso de desarrollo. Las ideas que le dan forma a GRAPPLE no son nuevas. Son una condensación de las ideas de varias otras personas. La primera palabra en las siglas de GRAPPLE, Directivas, es importante: ésta no es una férrea metodología. En vez de ello, es un conjunto de ideas adaptables y flexibles. Se puede interpretar como un patrón simplificado de un proceso de desarrollo. Se utiliza para el correcto empleo de UML dentro de un contexto. Con algunos ajustes, GRAPPLE puede aplicarse en diversas organizaciones. Da la oportunidad a un gerente de proyectos, con creatividad, de agregar sus propias ideas respecto a lo que funcionará en una organización en particular, y puede sustraer los pasos incluidos que no funcionen. Esta metodología, si bien hace un notorio énfasis en el trabajo en equipo, es entendible que para propósitos de la construcción del Sistema de Información que sustenta esta Tesis, la aplicación de GRAPPLE será unipersonal o individual.

### I.4.4.1 RAD3: la estructura de GRAPPLE

GRAPPLE consta de cinco segmentos. Cada segmento, en tumo, consta de diversas acciones. Cada acción trae consigo un producto del trabajo, y cada acción es responsabilidad de uno o varios participantes en la construcción de la herramienta de Software. En muchos casos, el gerente de proyectos puede combinar los productos de trabajo en un informe que presente al cliente. Los productos de trabajo, de hecho, tienen el mismo propósito que un avance en papel, sin sumergirse en el papeleo.

GRAPPLE se encausa a los sistemas orientados a objetos. Por ello, las acciones dentro de cada segmento se orientan a crear productos de trabajo de una naturaleza orientada a objetos.

Los segmentos son:

1. Recopilación de necesidades
2. Análisis
3. Diseño
4. Desarrollo
5. Distribución

Esto nos otorga un acrónimo RADDD o RAD3. Luego del tercer segmento, el gerente de proyectos combina los productos de trabajo en un documento de diseño para proporcionárselo al cliente y los desarrolladores. Cuando se han completado todos los segmentos RAD3, todos los productos de trabajo se combinan para conformar un documento que define al sistema.

Antes de iniciar tales segmentos, se debe asumir que el cliente ha generado un caso de negocios para el nuevo sistema. También se debe asumir que los miembros del equipo de desarrollo, particularmente los analistas, han leído tanta documentación relevante como sea posible.

A continuación se da una breve explicación de cada segmento, considerando que UML se ajusta a cada uno de estos segmentos.

### I.4.4.2 Recopilación de necesidades

Consiste en comprender lo que desea el cliente, ya que en caso contrario nunca podrá generarse el sistema adecuado. Todos los análisis de casos de uso en el mundo no ayudarán si no se entienden las bases del dominio del cliente y el problema que se desea resolver. Este segmento a su vez se divide en las siguientes actividades:

- *Descubrir los procesos de negocios.* Se debe empezar el proceso de desarrollo mediante la comprensión de los procesos de negocios del cliente, en especial aquellos que tratará de mejorar con el sistema propuesto. Para comprenderlo, un analista entrevistará al cliente o a una persona con el conocimiento necesario que sea designada por el cliente, a quien se le pregunte los pasos relevantes del proceso uno por uno. Una consecuencia importante será que el analista obtendrá un vocabulario de trabajo en un subconjunto de la terminología del cliente. El analista usará este vocabulario cuando entreviste al cliente en la siguiente acción. El producto del trabajo es un diagrama de actividades o conjunto de ellos que captan los pasos y puntos decisivos en el proceso (o procesos) del negocio.

- *Elaborar un análisis del dominio.* Puede realizarse durante la misma sesión del segmento anterior. El objetivo es comprender de la mejor manera posible el dominio del cliente. Esta acción y la anterior tratan de conceptos, no del sistema que va a generar. El analista tiene que acomodarse en el mundo del cliente, pues, a fin de cuentas, es el interlocutor entre el cliente y el equipo de desarrollo. El analista entrevista al cliente con la finalidad de comprender las principales entidades en el dominio del cliente, -es recomendable que, durante la plática entre el cliente y el analista, otro miembro del equipo tomará las notas (de forma óptima) en un equipo de cómputo portátil equipado con un procesador de textos, y un modelador de objetos generará un diagrama de clases de alto nivel-. El modelador de objetos prestará atención a los sustantivos y empezará a convertir a cada uno en una clase. Finalmente, algunos de esos sustantivos se convertirán en atributos. El modelador también prestará atención a los verbos, que se convertirán en operaciones de las clases. En este punto, una herramienta de modelado computarizada sería muy útil. El producto del trabajo es un diagrama de clases de alto nivel y un conjunto de minutas.

- *Identificación de los sistemas cooperativos.* Normalmente, los sistemas de negocios actuales no emergen de la nada, tienen que colaborar con otros. En las primeras instancias del proceso, el equipo de desarrollo verá exactamente de qué sistemas dependerá el nuevo sistema, y cuáles dependerán de él. Un diseñador de sistemas se encarga de esto, y producirá un diagrama de distribución como su producto del trabajo. El diagrama muestra a los sistemas como nodos, con líneas de comunicación entre ellos, componentes residentes y dependencias entre componentes.

- *Definir las necesidades del sistema.* Esto es muy importante, ya que en esta acción, el equipo realiza su primera sesión de JAD (Desarrollo conjunto de aplicaciones). Una sesión JAD reúne a quienes toman las decisiones en la empresa del cliente, a los usuarios potenciales y a los miembros del equipo de desarrollo. Debe haber alguien que modere la sesión; el trabajo del moderador es obtener una respuesta de quienes toman las decisiones y de los usuarios acerca de lo que esperan que haga el sistema. Al menos debe haber dos miembros del equipo que tomen notas, y el modelador de objetos deberá refinar el diagrama de clases que se obtuvo previamente. El producto del trabajo es un diagrama de paquetes. Cada paquete representa a un área de alto nivel de la funcionalidad del sistema (por ejemplo: "ayuda con el servicio a clientes").

Cada paquete agrupa un conjunto de casos de uso (por ejemplo: "obtener el historial del cliente" o "tratar con el cliente"). La complejidad del sistema será lo que determine la duración de la sesión. La empresa del cliente debe hacer el compromiso de invertir el tiempo que sea necesario.

- *Presentación de los resultados al cliente.* Cuando el equipo finaliza todas las acciones de Necesidades, el administrador de proyectos presentará los resultados al cliente. Algunas empresas pueden solicitar la aprobación del cliente en este punto, para que pueda proceder el desarrollo. Otras podrían necesitar una estimación de los costos de acuerdo con los resultados. De esta manera, el producto del trabajo podría variar de acuerdo con la empresa.

### I.4.4.3 Análisis

En este segmento, el equipo profundiza en los resultados del segmento Necesidades y aumentará su comprensión del problema. De hecho, partes de este segmento empezarán durante el segmento de Necesidades, conforme el modelador de objetos empieza a depurar el diagrama de clases durante la sesión JAD de Necesidades. Este segmento a su vez se divide en las siguientes actividades:

- *Comprensión del uso del sistema.* Consiste en un análisis de casos de uso de alto nivel. En una sesión JAD con usuarios potenciales, el equipo de desarrollo trabaja con los usuarios para descubrir a los actores que iniciarán cada caso de uso, y los actores que serán beneficiados. Un moderador interviene en la sesión, y dos miembros del equipo toman notas. Luego de algunos proyectos, el moderador de esta sesión podría convertirse en el analista de casos de uso. El equipo también intentará desarrollar nuevos casos de uso y casos de uso abstractos. El producto del trabajo será un conjunto de diagramas de casos de uso que muestren a los actores y las dependencias estereotipadas (“extender” e “incluir”) entre los casos de uso.

- *Hacer realidad los casos de uso.* Es decir, el equipo de desarrollo continúa su trabajo con los usuarios. El objetivo es analizar la secuencia de pasos en cada caso de uso. Esta sesión JAD puede ser la continuación de la sesión previa. El producto de esta actividad es una descripción textual de los pasos en cada caso de uso.

- *Depurar los diagramas de clases.* Durante las sesiones JAD, el modelador de objeto debe escuchar todos los debates y continuar con la depuración del diagrama de clases. En este punto, el modelador de objetos deberá rellenar los nombres de las asociaciones, clases abstractas, multiplicidades, generalizaciones y agregaciones. El resultado será un diagrama de clases depurado.

- *Analizar cambios de estado en los objetos.* El modelador de objetos depurará el modelo mediante la presentación de cambios de estado conforme sea necesario. El producto del trabajo es un diagrama de estados.

- *Definir la comunicación entre objetos.* Una vez que se cuente con un conjunto de diagramas de casos de uso y un diagrama depurado de clases, se debe definir la forma en que los objetos se comunican. El modelador de objetos debe desarrollar un conjunto de diagramas de secuencias y de colaboraciones para delinear la comunicación. Deben incluirse los cambios de estado.

- *Analizar la integración con diagramas de colaboraciones.* Al tiempo de realizar los pasos anteriores, el diseñador del sistema descubre los detalles específicos de la integración con los sistemas cooperativos. ¿Qué tipo de comunicación está envuelto? ¿Cuál es la arquitectura de red? Si el sistema tendrá que utilizar bases de datos, un analista de bases de datos determinará la arquitectura (física o lógica) de ellas. Con esto se obtienen los diagramas de distribución detallados y (de ser necesario) modelos de datos.

### I.4.4.4 Diseño

En este segmento, el equipo trabajará con los resultados del segmento de Análisis para diseñar la solución. En el diseño y en el análisis se harán las revisiones pertinentes hasta que el diseño se haya completado. De hecho, algunas de las metodologías combinan al Análisis y al Diseño en una sola fase.

- *Desarrollo y depuración de los diagramas de objetos.* Los programadores deben tomar el diagrama de clases y generar cualquier diagrama de objetos que sea necesario. "Aterrizarán" lo indicado en los diagramas de objetos mediante el análisis de cada operación y el desarrollo de un diagrama de actividades correspondiente. Los diagramas de actividades fungirán como la base de gran parte del código en el segmento de desarrollo. El resultado de esta actividad son los diagramas de objetos y los de actividades.

- *Desarrollo de diagramas de componentes.* Los programadores son quienes juegan un importante papel en esta acción. La tarea es visualizar los componentes que resultarán del siguiente segmento y mostrar las dependencias entre ellos. Los diagramas de componentes resultan de llevar a cabo esta actividad.

- *Planeación para la distribución.* Cuando se haya completado el diagrama de componentes, el diseñador del sistema empieza a planear la distribución e integración con sistemas cooperativos. Debe Crear un diagrama de distribución que muestre el lugar donde se encontrarán los componentes. Con ello se obtiene un diagrama que sea parte del de distribución generado con anterioridad.

- *Diseño y prototipos de la interfaz del usuario.* Esto trae consigo otra sesión JAD con los usuarios. Aunque esto es parte del Diseño, esta sesión puede ser la continuación de anteriores sesiones JAD con los usuarios —un indicio de la interacción entre el Análisis y el Diseño—. La interfaz del usuario debería permitir la consumación de

todos los casos de uso. Para ello, un analista de GUI deberá trabajar con los usuarios para desarrollar prototipos, en papel, de las pantallas que corresponderán a grupos de casos de uso. Los usuarios pegarán papeletas removibles que representen los componentes de la pantalla (botones, casillas de verificación, listas desplegables, menús y cosas así). Cuando los usuarios queden satisfechos de la posición de los componentes, los desarrolladores debe generar prototipos de las pantallas para que sean aprobados por los usuarios. Los productos del trabajo serán capturas de pantalla de los prototipos resultantes.

- *Pruebas de diseño.* Los casos de uso permiten el diseño de pruebas del software. El objetivo es evaluar si el software hace lo que se supone que debería (esto es, que hace lo que se especifica en los casos de uso). Preferentemente, un desarrollador o especialista de pruebas externo al equipo de desarrollo debe utilizar los diagramas de casos de uso para crear secuencias de comandos en herramientas automatizadas de pruebas. Tales secuencias de comandos conformarán el resultado de esta actividad.

- *Iniciar la documentación.* Nunca es demasiado pronto para empezar a documentar el sistema para los usuarios finales y gerentes de sistemas. Los especialistas en la documentación deben trabajar en conjunto con los diseñadores para empezar a generar un bosquejo de la documentación y llegar a una estructura de alto nivel para cada documento.

### I.4.4.5 Desarrollo

De este segmento se encargan los programadores. Con suficiente análisis y diseño, este segmento debería realizarse con rapidez y sin problemas.

- *Generación del código.* Con los diagramas de clases, de objetos, de actividades y de componentes a la mano, los programadores generarán el código del sistema.

- *Verificación del código.* Los especialistas en pruebas (no los desarrolladores) ejecutarán secuencias de comandos de prueba para evaluar si el código hace lo que se pretende. Esta acción alimenta a la anterior y viceversa, hasta que el código pase todos los niveles de prueba.

- *Generación de interfaces del usuario, conexión con el código y prueba.* Esta acción crea las interfaces de usuario ya aprobadas. El especialista en GUI las genera y conecta con el código. Las pruebas ulteriores aseguran que las interfaces funcionen adecuadamente.

- *Consumación de la documentación.* Durante el segmento de desarrollo, los especialistas en documentación trabajan en paralelo con los desarrolladores para asegurar la entrega oportuna de toda la documentación.

### I.4.4.6 Distribución

Cuando un sistema se ha finalizado, se distribuye en el hardware adecuado y se integra con los sistemas cooperativos. No obstante, la primera acción en este segmento puede iniciar antes de que el segmento de Desarrollo comience.

- *Planeación para copias de seguridad y recuperación.* El diseñador del sistema creará un plan que incluya los pasos a seguir en caso de que el sistema falle. El plan, producto del trabajo de esta acción, establece lo que se deberá hacer para crear una copia de seguridad del sistema y para recuperarse del error.

- *Instalación del sistema terminado en el hardware adecuado.* El diseñador del sistema, con toda la ayuda necesaria de los programadores, distribuye el sistema terminado en los equipos de cómputo adecuados. El producto del trabajo es el sistema completamente distribuido.

- *Verificación del sistema instalado.* Finalmente, el equipo de desarrollo verifica el sistema instalado. Asegurando que se ejecute como se esperaba, verificando que el plan de copias de seguridad y recuperación funcione. Los resultados de estas pruebas determinarán si se necesita hacer una depuración ulterior.

- *Celebración.* Sin mayor explicación, el equipo de desarrollo podrá inventar los productos del trabajo de esta acción.

Los movimientos de los segmentos y acciones en GRAPPLE, van de lo general a lo particular. Comienza con una interpretación conceptual del dominio, trasciende a la funcionalidad de alto nivel, profundiza en los casos de uso, depura los modelos y diseña, desarrolla y distribuye el sistema. Cabe hacer notar que hay más acciones en los segmentos de Análisis y Diseño que en el de Desarrollo ya que la idea es utilizar tanto tiempo como sea necesario en el análisis y diseño, para que la codificación se realice sin problemas. Entre más y mejor se analice, más cerca se estará del ideal. Como se describió, GRAPPLE consta de cinco segmentos: Recopilación de necesidades. Análisis, Diseño, Desarrollo y Distribución. Cada segmento consta de diversas acciones, y cada una de ellas da por resultado un producto del trabajo. Los diagramas UML constituyen productos del trabajo para varias de las acciones. Razón por la cual, a continuación se describirá este tipo de Modelado de Objetos.

# I.5 ARQUITECTURA DE TRES CAPAS

La arquitectura de una aplicación es la vista conceptual de la estructura de esta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código. La arquitectura de aplicaciones de tres-capas, se basa en componentes. El objetivo es unificar las aplicaciones para PC, las aplicaciones cliente-servidor y las aplicaciones basadas en la Web, lo cual es posible para aplicaciones de cualquier tamaño.

En nuestros días mucha información importante está almacenada en aplicaciones como sistemas de correo electrónico, y aún más recientemente en servicios de directorio. Microsoft habla sobre Universal Data Access (Acceso Universal a Datos) como una serie de manejadores e interfaces diseñadas para proveer una forma de conseguir acceder a este tipo de almacenamientos y más aún a datos como archivos de formato especiales, datos de posición geoespacial, datos científicos no estándar, etc.

Los servicios son puestos en la red y operan de manera cooperativa para dar soporte a uno o más procesos de negocios. En este modelo, una aplicación se convierte en un conjunto de servicios de usuario, negocios y datos que satisfacen las necesidades de los procesos de negocios o procesa su soporte. Como los servicios están diseñados para el uso general y siguen lineamientos de interfaz publicados, pueden ser reutilizados y compartidos entre múltiples aplicaciones.

La arquitectura de tres capas divide una aplicación en tres categorías o niveles, los cuales consisten en:

## I.5.1 LA PRESENTACIÓN (GUI)

Los servicios de presentación proporcionan la interfaz necesaria para presentar información y reunir datos. También aseguran los servicios de negocios necesarios para ofrecer las capacidades de transacciones requeridas e integrar al usuario con la aplicación para ejecutar un proceso de negocios. Los servicios de presentación generalmente son identificados con la interfaz de usuario, y normalmente residen en un programa ejecutable localizado en la estación de trabajo del usuario final. Aún así, existen oportunidades para identificar servicios que residen en componentes separados.

El cliente proporciona el contexto de presentación, que permite ver los datos remotos a través de una capa de presentación, o también una aplicación WIN32 como puede ser los formularios de Visual Basic.

Mediante el uso de componentes, se separa la programación que da acceso a los datos en las bases de datos y aplicaciones desde el diseño y otros contenidos de la aplicación. Esto ayuda a asegurar que los desarrolladores estén libres para enfocarse en escribir su lógica de negocios en

componentes sin preocuparse acerca de cómo se muestra la salida. Recíprocamente, esto da libertad a los diseñadores de usar herramientas familiares para modificar la interfaz.

La capa de servicios de presentación es responsable de:

- Obtener información del usuario.
- Enviar la información del usuario a los servicios de negocios para su procesamiento.
- Recibir los resultados del procesamiento de los servicios de negocios.
- Presentar estos resultados al usuario.

### I.5.2 LAS REGLAS DE NEGOCIOS

Los servicios de negocios son el "puente" entre un usuario y los servicios de datos. Responden a peticiones del usuario (u otros servicios de negocios) para ejecutar una tarea de este tipo. Cumplen con esto aplicando procedimientos formales y reglas de negocio a los datos relevantes. Cuando los datos necesarios residen en un servidor de bases de datos, garantizan los servicios de datos indispensables para cumplir con la tarea de negocios o aplicar su regla. Esto aísla al usuario de la interacción directa con la base de datos.

Una tarea de negocios es una operación definida por los requerimientos de la aplicación, como introducir una orden de compra o imprimir una lista de clientes. Las reglas de negocio (business rules) son políticas que controlan el flujo de las tareas. Como las reglas de negocio tienden a cambiar más frecuentemente que las tareas específicas de negocios a las que dan soporte, son candidatos ideales para encapsularlas en componentes que están lógicamente separados de la lógica de la aplicación en sí. Para ayudar a los desarrolladores a construir la lógica de negocio basado en componentes incluye un conjunto muy poderoso de servicios que se encargan de la comunicación en una aplicación de tres capas. Estos servicios están altamente integrados unos con otros bajo un sistema operativo y expuestos de forma única a través de COM. El nivel de servicios de negocios es responsable de:

- Recibir la entrada del nivel de presentación.
- Interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a automatizar (por ejemplo, la preparación de impuestos por ingresos, el procesamiento de ordenes y así sucesivamente).
- Enviar el resultado procesado al nivel de presentación.

Algunos de los servicios para la capa de Negocios son los siguientes:

- Transacciones y Servicios de Componentes.
- Servicios Asíncronos.

## I.5.3 LOS DATOS

El nivel de servicios de datos es responsable de:

- Almacenar los datos.
- Recuperar los datos.
- Mantener los datos.
- La integridad de los datos.

Los servicios de datos tienen una variedad de formas y tamaños, incluyendo los sistemas de administración de bases de datos relacionales (SABDs) como Microsoft SQL Server o Sybase Adaptive Server y sistemas de archivos tales como el Sistema de Archivos NTFS. Lo que propone Windows DNA para el acceso a datos es el Universal Data Access (UDA). UDA es un marco de trabajo basado en COM y estándares abiertos de la industria. En vez de requerir que todos los datos se encuentren en un almacenamiento común, UDA provee una interfaz programable común a virtualmente cualquier tipo de almacenamiento, ya sea estructurado o no. UDA especifica interfaces a nivel de sistema llamadas OLE DB. Proveedores de datos como ser SQL Server y Microsoft Exchange implementan interfaces OLE DB para proveer acceso a sus almacenamientos específicos. UDA también especifica una interfaz de programación de alto nivel llamada ActiveX Data Objects (ADO) que usa OLE DB para acceder a los datos, el cual representa un modelo de programación más fácil para los desarrolladores.

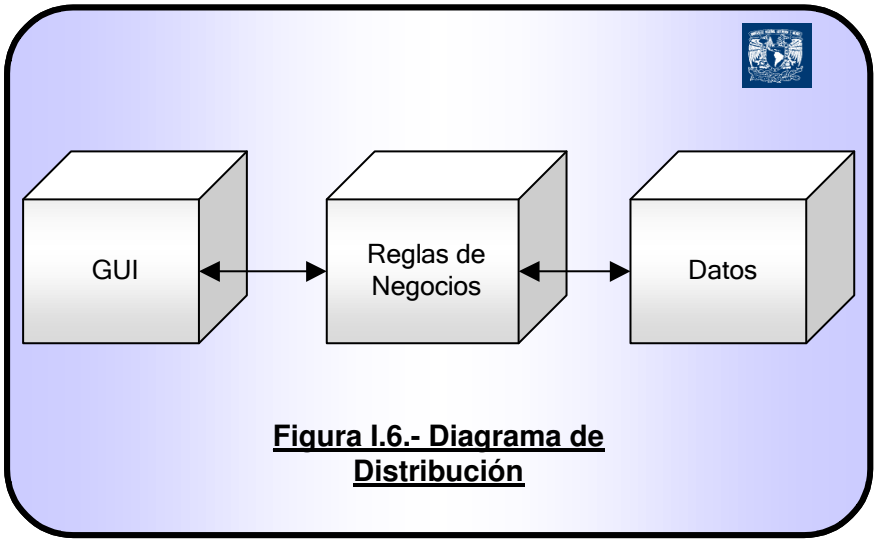



**Figura I.6.- Diagrama de Distribución**

La figura I.6 muestra la arquitectura de tres capas. Con una arquitectura de tres capas, pero separando las reglas de negocios de las empresas, no se requerirá invertir tiempo de más para entender las reglas de negocios, si se realizan modificaciones o cambios al sistema, herramientas de desarrollo del sistema o RDMBS (Sistema Administrador de Bases de Datos).

Esta arquitectura también resuelve los problemas de reuso y escalabilidad, y además como permite diseñar las reglas de negocios como una capa separada, puede usarse cualquier aplicación o RDBMS como se muestra en la figura I.7. Con PowerBuilder se pueden diseñar las reglas de negocios como componentes, tal como Objetos de Usuario No Visuales, Controles ActiveX, Applets de Java, JavaBeans, Objetos en C++ y varios más.

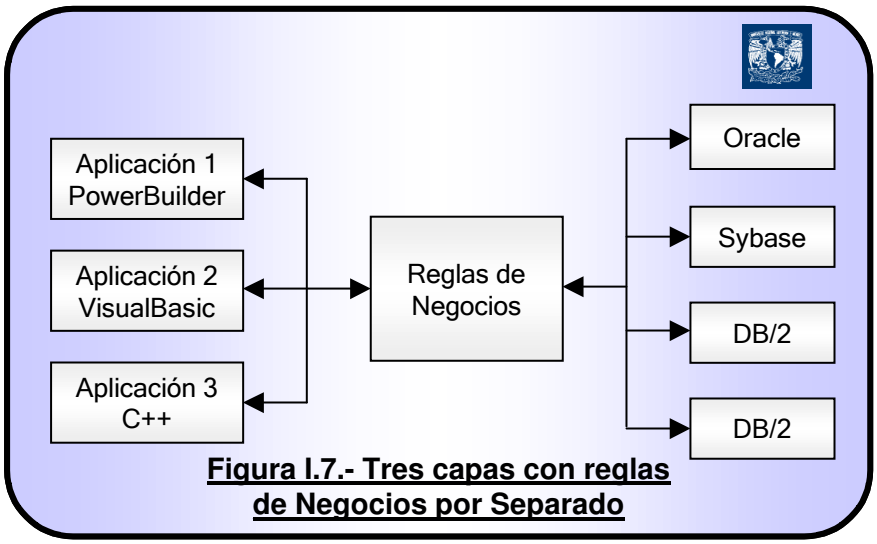



**Figura I.7.- Tres capas con reglas de Negocios por Separado**

# I.6 UML

Dado que en la actualidad es importante reducir los tiempos para el desarrollo de sistemas, es fundamental contar con un diseño sólido. Esto dio la pauta para que Grady Booch, James Rumbaugh e Ivar Jacobson crearan una notación de diseño que los analistas, desarrolladores y clientes aceptaran como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica). Resultado de esto es UML, el cual surgió en la década de los 90´s luego de la búsqueda de un lenguaje de modelamiento que unificara a la industria, que siguió a la "guerra de métodos" de los 70´s y 80´s. A pesar de que UML evolucionó primeramente de varios métodos orientados al objeto de segunda generación (en nivel de notación), UML no es simplemente un lenguaje para modelamiento orientado al objeto de tercera generación. Su alcance extiende su uso más allá de sus predecesores. Y es la experiencia, experimentación y una gradual adopción del estándar lo que revela su verdadero potencial y posibilita a las organizaciones darse cuenta de sus beneficios.

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language), es un lenguaje gráfico para el desarrollo de sistemas, ya que permite visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML proporciona una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. En general permite generar material de apoyo que permita poder definir diagramas propios. Es decir, es un lenguaje de modelamiento para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Dentro de un proceso de sistema intensivo, un método es aplicado para llegar o evolucionar un sistema. Como un lenguaje, es usado para la comunicación. Se puede interpretar como un medio para capturar el conocimiento (semánticas) respecto a un tema y expresar el conocimiento (sintaxis) resguardando el tema propósito de la comunicación. El tema es el sistema en estudio. Además, se enfoca en la comprensión de un tema a través de la formulación de un modelo del tema (y su contexto respectivo). El modelo abarca el conocimiento cuidando del tema, y la apropiada aplicación de este conocimiento constituye inteligencia. Cuidando la unificación, integra las mejores prácticas de la ingeniería de la industria tecnológica y sistemas de información pasando por todos los tipos de sistemas (software y no - software), dominios (negocios vs software) y los procesos de ciclo de vida. En cuanto a cómo se aplica para especificar sistemas, puede ser usado para comunicar "qué" se requiere de un sistema y "cómo" un sistema puede ser realizado. En cuanto a cómo se aplica para visualizar sistemas, puede ser usado para describir visualmente un sistema antes de ser realizado. En cuanto a cómo se aplica para construir sistemas, puede ser usado para guiar la realización de un sistema similar a los "planos". En cuanto a cómo se aplica para documentar sistemas, puede ser usado para capturar conocimiento respecto a un sistema a lo largo de todo el proceso de su ciclo de vida.

UML no es un lenguaje de programación visual, sino un lenguaje de modelamiento visual, tampoco es una herramienta o depósito de especificación, sino un lenguaje para modelamiento de especificación. No es un proceso, sino que habilita procesos. En resumen, UML está relacionado con la captura, comunicación y nivelación (disgregación en niveles) de conocimientos.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de tales diagramas es presentar diversas perspectivas de un sistema los cuales se les conoce como MODELO el cual describe lo que se pretende realice un sistema, sin embargo, nunca va a decir como implementar dicho sistema. Cabe hacer énfasis en que debido a que UML es un lenguaje, cuenta con reglas para combinar tales elementos. A continuación se citan los principales elementos gráficos:

- Diagrama de Clases.
- Diagrama de Objetos.
- Diagrama de Caso de Uso.
- Diagramas de Estados.
- Diagrama de Secuencias.
- Diagrama de Actividades.
- Diagrama de Colaboraciones.
- Diagrama de Componentes.
- Diagrama de Distribución.

Además existe un par de características adicionales representadas por dos elementos gráficos más: Paquetes y Estereotipos.

### I.6.1 DIAGRAMA DE CLASES

Un diagrama de clases esta compuesto por los siguientes elementos: Clase (atributos, métodos y visibilidad) y Relaciones (herencia, composición, agregación, asociación y uso). *Clase* es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.). Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento. En UML, una clase es representada por un rectángulo que posee tres divisiones, tal y como se muestra en la figura I.8:

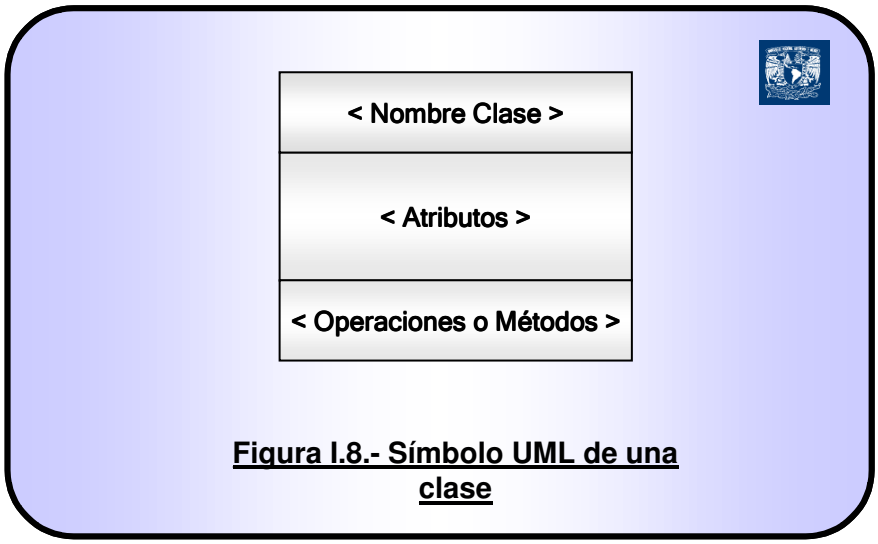



**Figura I.8.- Símbolo UML de una clase**

En donde:

- ***División Superior****:* Contiene el nombre de la Clase.
- *División Intermedia:* Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser privada, protegida o pública).
- *División Inferior:* Contiene los métodos, operaciones o acciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: pueden ser privada, protegida o pública).

**Atributos:** Son las características de una Clase, las cuales dependiendo del grado de comunicación y visibilidad de ellas con el entorno pueden ser de tres tipos:

- *public:* El atributo es visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- *private:* El atributo sólo es accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).
- *protected:* El atributo no es accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

**Métodos:** Son las operaciones de una clase, es decir, son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

- *public:* El método es visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- *private:* El método sólo es accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).

- *protected:* El método no es accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven.

## I.6.2 DIAGRAMA DE OBJETOS

Un objeto es una instancia de una clase (una entidad que tiene valores específicos de los atributos y acciones). El nombre del objeto se encuentra a la izquierda de dos puntos (:), y el nombre de la clase a la derecha, tal y como se muestra en la figura I.9.

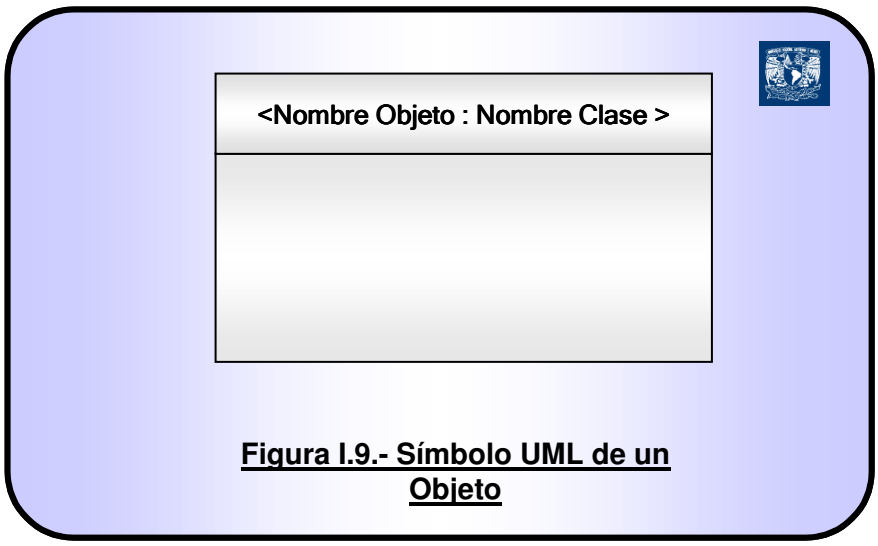



**Figura I.9.- Símbolo UML de un Objeto**

## I.6.3 DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso representa la forma en como un Cliente (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso). Tal y como se muestra en la figura I.10, el diagrama de casos de uso consta de los siguientes elementos:

En donde:

- *Actor:* Es el papel que el usuario juega con respecto al sistema. Es importante destacar que esto indica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

- *Casos de Uso.* Es una tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

- *Relaciones de Uso, Herencia y Comunicación.* Forma en que las clases y actores se conectan entre sí. Los tipos de relación son: Asociación, Dependencia o Instanciación y Generalización.

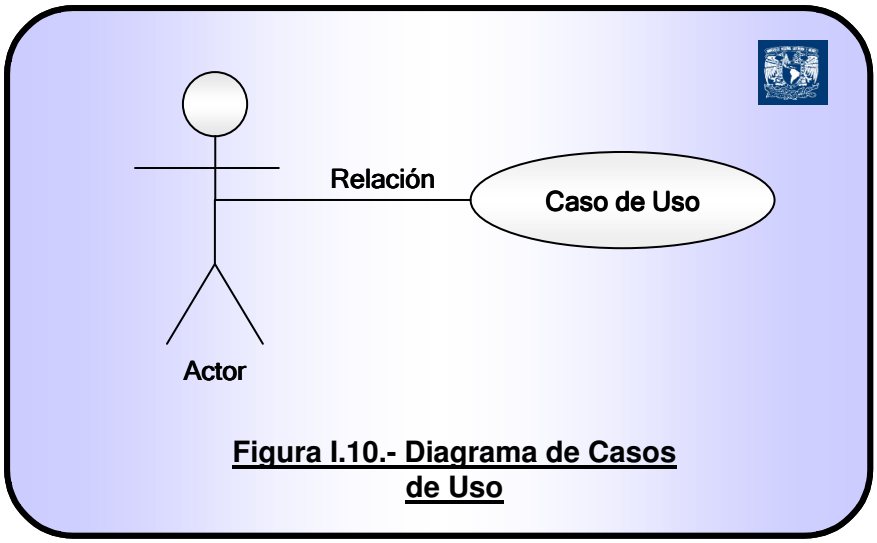



**Figura I.10.- Diagrama de Casos de Uso**

## I.6.4 DIAGRAMAS DE ESTADOS.

Permite caracterizar un cambio en un sistema. Presenta los estados en los que puede encontrarse un objeto junto con las transacciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado. También se le denomina Motor de Estado. Su objetivo es mostrar las condiciones de un solo objeto. Un ejemplo de este tipo es el mostrado en la figura I.11.

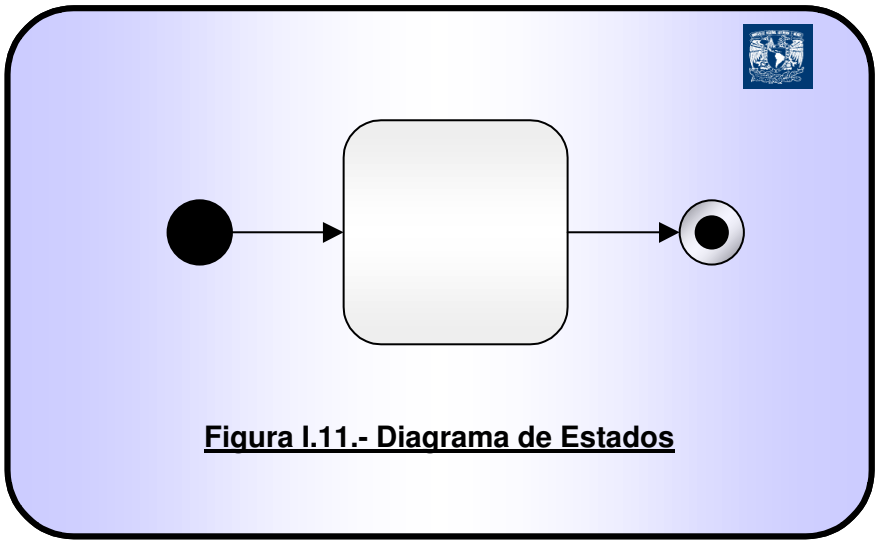


**Figura I.11.- Diagrama de Estados**

## I.6.5 DIAGRAMA DE SECUENCIAS.

Establece el siguiente paso y muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo. La figura I.12 muestra la conformación básica de este tipo de diagrama.

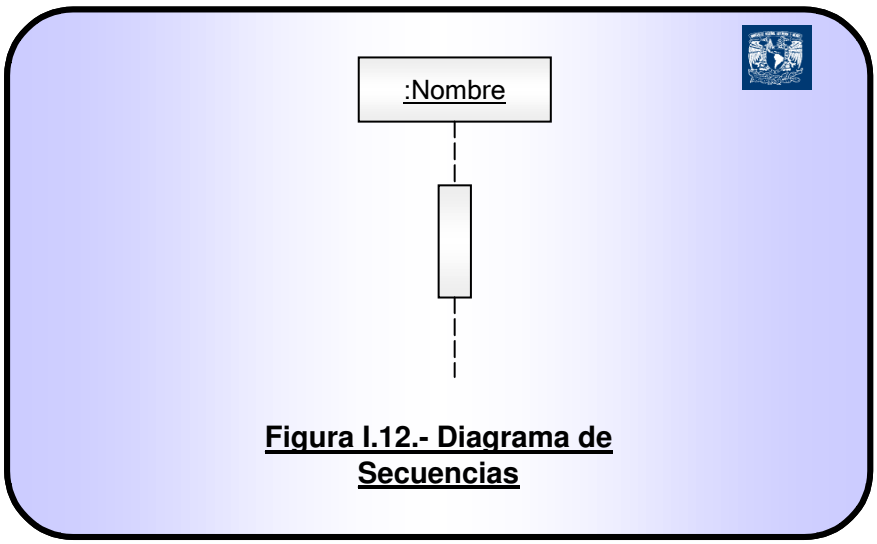



**Figura I.12.- Diagrama de Secuencias**

## I.6.6 DIAGRAMA DE ACTIVIDADES.

Su objetivo es mostrar los pasos en una operación o proceso. Es similar al diagrama de flujo tal y como se observa en la figura I.13.

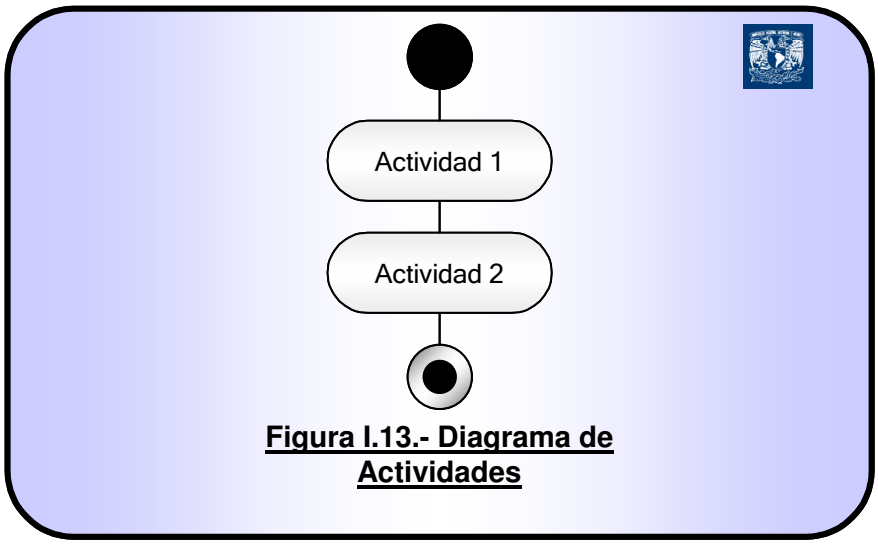



**Figura I.13.- Diagrama de Actividades**

## I.6.7 DIAGRAMA DE COLABORACIONES.

Este diagrama muestra la forma en que los objetos colaboran entre sí, tal como sucede con un diagrama de secuencias, pero además de las relaciones entre objetos, este diagrama muestra los mensajes que se envían los objetos entre sí. En la figura I.14 se muestra su estructura básica.

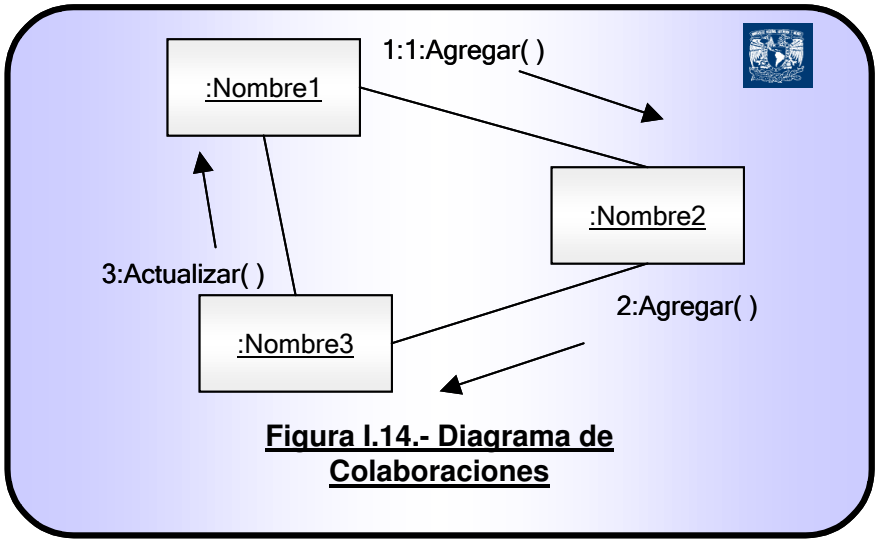



**Figura I.14.- Diagrama de Colaboraciones**

## I.6.8 DIAGRAMA DE COMPONENTES.

La intención es representar un Componente de Software. Por lo que contiene componentes, interfaces y relaciones. También pueden aparecer otros tipos de símbolos. Ver figura I.15.

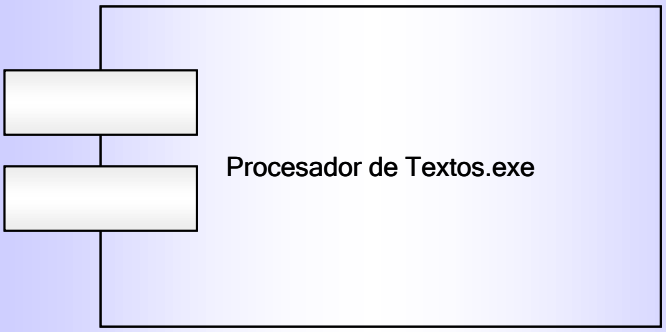



**Figura I.15.- Diagrama de Componentes**

## I.6.9 DIAGRAMA DE DISTRIBUCIÓN.

El propósito de este diagrama es MOSTRAR la arquitectura de distribución, los nodos de hardware (procesadores y/o dispositivos) y las conexiones físicas entre los nodos. Ver figura I.16.

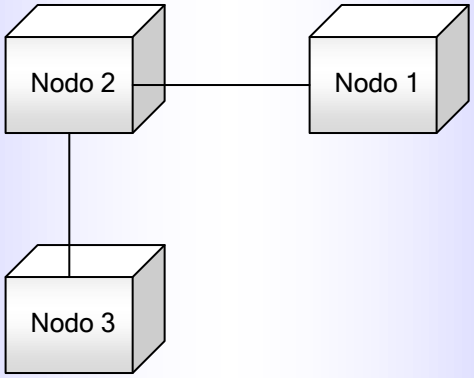



**Figura I.16.- Diagrama de Distribución**

Los diagramas antes mencionados serán la base para desarrollar el análisis del sistema. Ahora bien, debido a que UML evolucionó primeramente de varios métodos orientados al objeto de segunda generación (en cuanto a nivel de notación), la mayoría de analistas, programadores y demás personal involucrado en la creación de sistemas que se auxilian del UML, creen que sólo es relativo a sistemas de software orientados al objeto, cuando actualmente no es simplemente un lenguaje para el diseño orientado al objeto de tercera generación, sino un "lenguaje para modelamiento unificado" relativo a sistemas en general. El éxito de UML es medido por su apropiado uso por lo que, es importante considerar que su utilización no garantiza el éxito, sino que permite a quienes lo utilizan, enfocarse en la distribución de valor.

# I.7 BASES DE DATOS

Una base de datos es una colección de datos que es gestionada y organizada por un software específico, el DBMS (DataBase Management System, Sistema de Gestión de Bases de Datos). Un DBMS es sustancialmente un software que se coloca entre el usuario y los datos como tales. Gracias a este estrato intermedio el usuario y las aplicaciones no acceden a los datos tal y como se memorizan efectivamente, es decir a su representación física, sino que se ve sólo una representación lógica. Esto permite un grado elevado de independencia entre las aplicaciones y la memorización física de los datos. El administrador de la base de datos, si lo necesita, puede decidir memorizar los datos de un modo diferente o incluso cambiar el DBMS sin que las aplicaciones, es decir los usuarios, se resientan. Lo importante es que no cambie la representación lógica de esos datos, que es la única cosa que los usuarios conocen. Esta representación lógica se conoce como 'Esquema de la base de datos' y es la forma de representación de los datos de más bajo nivel a la que un usuario de la base de datos puede acceder.

La característica principal según la cual los DBMS se clasifican es la representación lógica de los datos que muestran a sus usuarios. Con el paso de los años, se han adoptado numerosos modelos para los datos, al frente de los cuales existen diversos tipos de bases de datos.

## I.7.1 BASES DE DATOS JERÁRQUICOS

Los datos se organizan en grupos unidos entre ellos por relaciones de "posesión", en las que un conjunto de datos puede tener otros conjuntos de datos, pero un conjunto puede pertenecer sólo a otro conjunto. La estructura resultante es un árbol de conjuntos de datos.

## I.7.2 BASES DE DATOS RETICULARES

El modelo reticular es muy parecido al jerárquico, y de hecho nace como una extensión de este último. También en este modelo los conjuntos de datos están unidos por relaciones de posesión, pero cada conjunto de datos puede pertenecer a uno o más conjuntos. La estructura resultante es una red de conjuntos de datos.

## I.7.3 BASES DE DATOS RELACIONALES

Las bases de datos que pertenecen a esta categoría se basan en el modelo relacional, cuya estructura principal es la relación, es decir una tabla bidimensional compuesta por líneas y columnas. Cada línea, que en terminología relacional se llama tupla, representa una entidad que se pretende almacenar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos. Entidades con características comunes, es decir descritas por el mismo conjunto de atributos, formarán parte de la misma relación.

Las bases de datos relacionales son el tipo de bases de datos actualmente más difundido. Los motivos de este éxito son fundamentalmente dos:

- Ofrecen sistemas simples y eficaces para representar y manipular los datos.
- Se basan en un modelo, el relacional, con sólidas bases teóricas.

El modelo relacional fue propuesto por E.F. Codd. Gracias a su coherencia y facilidad de uso, el modelo se convirtió a partir de los años 80 en el más usado para la producción de DBMS.

La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tuplas) y columnas (atributos) tal y como se muestra en la figura I.17. Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

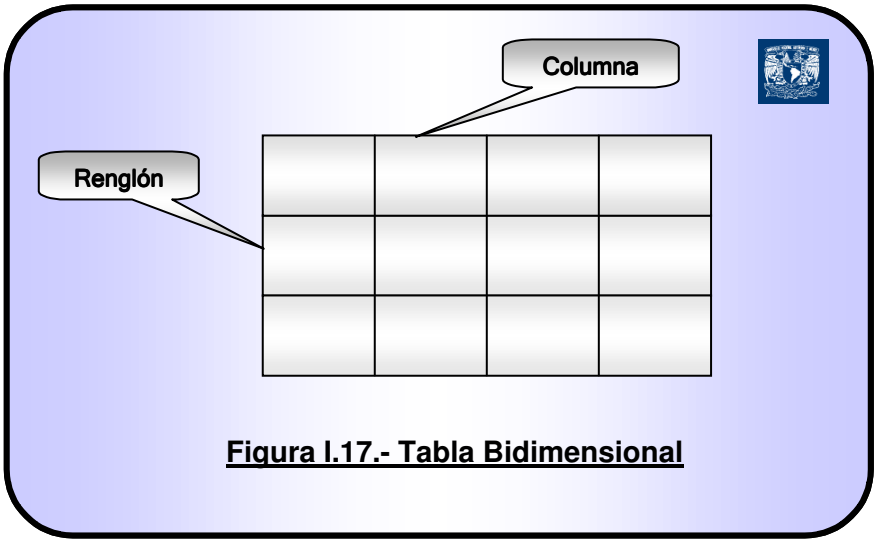



**Figura I.17.- Tabla Bidimensional**

Las bases de datos relacionales efectúan todas las operaciones en las tablas usando el álgebra relacional, aunque normalmente no le permiten al usuario usarla. El usuario interacciona con la base de datos a través de una interfaz diferente el lenguaje SQL, un lenguaje declarativo que permite escribir conjuntos de datos. Las instrucciones SQL vienen descompuestas por el DBMS en una serie de operaciones relacionales.

## I.7.4 BASE DE DATOS POR OBJETOS (OBJECT-ORIENTED)

El esquema de una base de datos por objetos está representado por un conjunto de clases que definen las características y el comportamiento de los objetos que poblarán la base de datos. La diferencia principal respecto a los modelos examinados hasta ahora es la no positividad de los datos. En efecto, con una base de datos tradicional (entendiendo con este término cualquier base de datos no por objetos), las operaciones que se tienen que efectuar en los datos se les piden a las aplicaciones que los usan. Con una base de datos orientada a objetos, al contrario, los objetos memorizados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. En cierto sentido, se podrá pensar en los objetos como en datos a los que se les ha puesto una inyección de inteligencia que les permite saber cómo comportarse, sin tener que apoyarse en aplicaciones externas.

Los primeros dos tipos de bases de datos, los jerárquicos y reticulares, hoy ya casi pertenecen a la historia de la informática. La mayor parte de las bases de datos que hoy se usan pertenece a la categoría de las bases de datos relacionales. Los motivos de este éxito (también comercial) hay que buscarlos en el rigor matemático y en la potencialidad expresiva del modelo relacional en que se basan, en su facilidad de uso y, último pero no menos importante, en la disponibilidad de un lenguaje de interrogación estándar, el SQL, que, al menos potencialmente, permite que se desarrollen aplicaciones independientes del DBMS concreto relacional que se use.

Las bases de datos por objetos son la nueva frontera en la investigación sobre las bases de datos; efectivamente, sus características de extendibilidad, que se derivan de la posibilidad de definir nuevos tipos de datos y comportamientos, las hacen particularmente apetecibles para todas las aplicaciones que usan datos complejos, como por ejemplo imágenes, sonidos o ambos coordinados. Por desgracia, la falta de un modelo universalmente aceptado para los objetos, así como que no exista un lenguaje de interrogación estándar, hace que cada productor implemente la propia visión específica, a menudo absolutamente incompatible con las otras. Recientemente, han aparecido en el mercado algunas bases de datos definidas como object-relational, que intentan introducir en el modelo relacional las características de extendibilidad propias de las bases de datos orientadas a objetos.

Independientemente del tipo de base de datos, las funciones principales que se pueden esperar de un DBMS son:

- Permitir el acceso a los datos a través de un esquema conceptual, en vez de hacerlo a través de un esquema físico.
- Compartir e integrar los datos entre aplicaciones diferentes.
- Controlar el acceso compartido a los datos.
- Garantizar la seguridad e integridad de los datos.

Gracias a estas características, las aplicaciones que se desarrollan pueden contar con una fuente de datos segura, fiable y generalmente escalable. Estas propiedades son deseables para aplicaciones que usan la red Internet como infraestructura y que por tanto tienen evidentes problemas de seguridad y de escala.

El sistema de esta Tesis, utilizará el modelo de bases de datos relacional, ya que en base a la experiencia resulta ser la de más fácil aplicación y mayor seguridad, además del reducido tiempo que implica su creación, en comparación con otros tipos de bases de datos.

### I.7.5 SQL

En una base de datos relacional se efectúan todas las operaciones en las tablas usando el álgebra relacional, aunque normalmente no le permiten al usuario usarla. Se interacciona con la base de datos a través de una interfaz diferente, mejor conocida como lenguaje SQL, un lenguaje declarativo que permite escribir conjuntos de datos. Las instrucciones SQL vienen descompuestas por el DBMS en una serie de operaciones relacionales.

### I.7.6 HISTORIA DE SQL

Empieza en 1974 con la definición, por parte de Donald Chamberlin y de otras personas que trabajaban en los laboratorios de investigación de IBM, de un lenguaje para la especificación de las características de las bases de datos que adoptaban el modelo relacional. Este lenguaje se llamaba SEQUEL (Structured English Query Language) y se implementó en un prototipo llamado SEQUEL-XRM entre 1974 y 1975. Las experimentaciones con ese prototipo condujeron, entre 1976 y 1977, a una revisión del lenguaje (SEQUEL/2), que a partir de ese momento cambió de nombre por motivos legales, convirtiéndose en SQL. El prototipo (System R), basado en este lenguaje, se adoptó y utilizó internamente en IBM y lo adoptaron algunos de sus clientes elegidos. Gracias al éxito de este sistema, que no estaba todavía comercializado, también otras compañías empezaron a desarrollar sus productos relacionales basados en SQL. A partir de 1981, IBM comenzó a entregar sus productos relacionales y en 1983 empezó a vender DB2. Durante la década de los años ochenta, numerosas compañías (Oracle, Sybase, entre otras) comercializaron productos basados en SQL, que se convierte en el estándar industrial para las bases de datos relacionales. En 1986, el ANSI adoptó SQL (sustancialmente adoptó el dialecto SQL de IBM) como estándar para los lenguajes relacionales y en 1987 se transformó en estándar ISO. Esta versión del estándar va con el nombre de SQL/86. En años posteriores éste sufrió diversas revisiones que condujeron primero a la versión SQL/89 y, posteriormente, a la actual SQL/92.

El hecho de tener un estándar definido por un lenguaje para bases de datos relacionales abre potencialmente el camino a la intercomunicabilidad entre todos los productos que se basan en él. Desde el punto de vista práctico, por desgracia las cosas fueron de otro modo. Efectivamente, en general cada productor adopta e implementa en la propia base de datos sólo el corazón del lenguaje SQL (el así llamado Entry level y a lo mucho, el Intermediate level),

extendiéndolo de manera individual según la propia visión que cada cual tenga del mundo de las bases de datos.

Actualmente, está en marcha un proceso de revisión del lenguaje por parte de los comités ANSI e ISO, que debería terminar en la definición de lo que en este momento se conoce como SQL3. Las características principales de esta nueva encarnación de SQL deberían ser su transformación en un lenguaje stand-alone (mientras ahora se usa como lenguaje hospedado en otros lenguajes) y la introducción de nuevos tipos de datos más complejos.

## I.7.7 OPERACIONES BÁSICAS CON SQL

### I.7.7.1 Sentencias de Selección o Consultas

Las consultas son las operaciones fundamentales en lenguaje SQL. La sentencia SELECT, que se utiliza para expresar consultas de información, es la más potente y compleja de las sentencias SQL. La sentencia SELECT recupera datos de una base de datos y los devuelve en forma de resultados de la consulta. Consta de seis cláusulas: las dos primeras (SELECT y FROM) obligatorias y las otras cuatro opcionales. La forma de la sentencia SELECT es:

SELECT [DISTINCT] {* | nombre_de_columna, ...}

FROM nombre_de_tabla [alias_tabla] ...

[WHERE expresión1 operador expresion2]

[GROUP BY {expresión_columna, ...} ]

[HAVING {condición} ]

[UNION [ALL] (SELECT ...)]

[ORDER BY {expresión_orden [DESC | ASC], ... ]

SELECT lista los datos a recuperar por la sentencia SELECT. Los elementos o datos a seleccionar pueden ser columnas de la base de datos o columnas a calcular por SQL cuando efectúa la consulta o también el asterisco (*) para recuperar todos los campos de una tabla. El operador DISTINCT, si se incluye, debe preceder la primera expresión de columna. Este operador elimina las filas o registros duplicados del resultado de la consulta. Las funciones de agrupamiento pueden ser también parte de una cláusula SELECT. Devuelven un único valor de un conjunto de registros. Pueden usarse con un nombre de campo, o en combinación con una expresión de columna más compleja. La expresión de columna puede ir precedida por el operador DISTINCT. El operador DISTINCT eliminará los valores repetidos de una expresión

de agrupamiento. SUM Devuelve la suma total de los valores de una expresión de columna o campo numérica. AVG Devuelve la media de los valores de una expresión de columna. COUNT Devuelve el número de valores en una expresión de columna. Un caso especial es COUNT(*), que nos devuelve el número de registros incluyendo aquellos registros con valores nulos. MAX Devuelve el valor más alto de los contenidos en una expresión de columna. MIN Devuelve el valor más bajo de los contenidos en una expresión de columna. FROM lista las tablas o ficheros que contienen los datos a recuperar por la consulta. WHERE dice a SQL que incluya solo ciertas filas o registros de datos en los resultados de la consulta, es decir, que tienen que cumplir los registros que se desean ver. La cláusula WHERE contiene condiciones en la forma: WHERE expresión1 operador expresion2. GROUP BY especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la base de datos, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria de resultados para cada grupo. Seguido de la cláusula GROUP BY se especifican los nombres de uno o más campos cuyos resultados se desean agrupados. HAVING dice a SQL que incluya solo ciertos grupos producidos por la cláusula GROUP BY en los resultados de la consulta. Al igual que la cláusula WHERE, utiliza una condición de búsqueda para especificar los grupos deseados. En otras palabras, especifica la condición que deben de cumplir los grupos. Sólo es válida si previamente se ha especificado la cláusula GROUP BY. UNION combina el resultado de dos sentencias SELECT en un único resultado. Este resultado se compone de todos los registros devueltos en ambas sentencias. Por defecto, los registros repetidos se omiten. Para no quitarlos se empleará la palabra ALL. Cuando se utilice el operador UNION, la lista de selección para cada sentencia SELECT debe tener el mismo número de expresiones de columnas con el mismo tipo de datos y en el mismo orden. ORDER BY ordena los resultados de la consulta en base a los datos de una o más columnas. Si se omite, los resultados saldrán ordenados por el primer campo que sea clave en el índice que se haya utilizado. Por tanto, indica como deben clasificarse los registros que se seleccionen.

### I.7.7.2 Sentencias para crear o destruir Tablas

La sentencia para crear una tabla es CREATE y para destruirla DROP.

*Creación de una tabla.* La sentencia para crear una tabla tiene la forma:

CREATE TABLE nombre_de_tabla (definición_columna, ...)

donde: *definición_columna* esta compuesto por el nombre de la columna o campo, seguida del tipo de dato de dicha columna. Los tipos de datos más comunes son : DECIMAL, NUMERIC, CHAR, DATE, DATETIME, INTEGER

*Destrucción de una tabla.* La sentencia para destruir una tabla tiene la forma:

DROP TABLE nombre_de_tabla

### I.7.7.3 Inserción de Información en Tablas

La sentencia de INSERT se utiliza para añadir registros a las tablas de la base de datos. El formato de la sentencia es:

INSERT INTO nombre_de_tabla [(nombre_de_columna, ...)] VALUES (expr, ...)

donde *expr* es una lista de expresiones o valores constantes, separados por comas, para dar valor a los distintos campos del registro que se añadirá a la tabla.

### I.7.7.4 Actualización de Información en Tablas

La sentencia UPDATE se utiliza para cambiar el contenido de los registros de una tabla de la base de datos. Su formato es:

UPDATE nombre_de_tabla SET nombre_columna = expr, ...

[WHERE { condición }]

donde *expr* es el nuevo valor que se desea asignar al campo que le precede. La expresión puede ser un valor constante o una subconsulta. La cláusula WHERE sigue el mismo formato que la vista en la sentencia SELECT y determina que registros se modificarán.

### I.7.7.5 Eliminación de Registros de Información de Tablas

La sentencia DELETE se utiliza para borrar registros de una tabla de la base de datos. El formato de la sentencia es:

DELETE FROM nombre_de_tabla

[WHERE { condición }]

La cláusula WHERE sigue el mismo formato que la vista en la sentencia SELECT y determina que registros se borrarán.

### I.7.7.6 Transacciones

Una transacción es una serie de cambios en la base de datos que deben ser tratadas como una sola. En otras palabras, que se realicen todos o que no se haga ninguno, pues de lo contrario se podrían producir inconsistencias en la base de datos. Cuando no se tiene activada una

transacción el gestor de base de datos ejecuta inmediatamente cada sentencia INSERT, UPDATE o DELETE que se le encomiende, sin posibilidad de deshacer los cambio en caso de ocurrir cualquier percance. Cuando se activa una transacción los cambios que se van realizando quedan en un estado de provisionalidad hasta que se realiza un COMMIT, el cual hará definitivos los cambios o hasta realizar un ROLLBACK eliminará todos los cambios producidos desde que se inició la transacción.

Como podrá notarse sólo se ha mencionado la sintaxis que se ha de llevar a cabo para las operaciones básicas en SQL, la utilización de tales operaciones se citarán una vez que se construyan todos los objetos de Bases de Datos que el sistema requiera. Esto no pretende ser un tutorial de SQL pero si mencionar de manera general la redacción de las instrucciones SQL para el manejo de información. De ahí que no se profundice en mencionar la estructura o sintaxis básica de sentencias SQL más complejas.

# *TEMA II: Herramientas de Software para el Desarrollo del Sistema*

OBJETIVO: Enunciar en qué consiste PowerDesigner el cual permitirá llevar cabo el diseño conceptual y físico. Dar a conocer los elementos que conforman a PowerBuilder, y finalmente, hacer una breve referencia a SQL-Server de tal manera que se identifique la importancia de esta herramienta en el desarrollo del Sistema.

# II.1 POWERDESIGNER

Sybase PowerDesigner es una herramienta de desarrollo de aplicaciones de fácil uso. Permite a los diseñadores y desarrolladores mejorar la productividad del ciclo de desarrollo desde el análisis al diseño generando un esquema de base de datos y de objetos de negocio.

Sybase PowerDesigner es la solución de modelamiento y diseño "todo en uno" para empresas que requieren construir o aplicar reingeniería a sus aplicaciones de negocio, rápidamente, a bajo costo, y de manera consistente. PowerDesigner elimina los obstáculos que se interponen en el camino del desarrollo efectivo de procesos: distintos conjuntos de habilidades, múltiples plataformas, y la multiplicidad de lenguajes de desarrollo que existen en la mayor parte de empresas.

Las últimas versiones, presentan nuevas características de modelamiento de procesos, modelamiento mejorado basado en UML, y soporta las técnicas de modelamiento tradicionales y emergentes dentro de un ambiente altamente gráfico. Esto recorta tiempo y complejidad a los proyectos de desarrollo que cubren múltiples plataformas y tipos de código. PowerDesigner también presenta un verdadero repositorio empresarial para almacenar y administrar toda la información de modelamiento y desarrollo de la empresa; esto minimiza inconsistencias y mejora dramáticamente la productividad del desarrollador.

## II.1.1 CARACTERÍSTICAS

PowerDesigner es un entorno verdaderamente integrado para el análisis y diseño de aplicaciones empresariales, con completas capacidades para:

- Modelamiento de Procesos de Negocio: PowerDesigner brinda poder a los usuarios no técnicos para diseñar y modelar procesos de negocio en términos reales del negocio, usando un modelo simple, fácil de usar, altamente gráfico, y no técnico. Incluye soporte a la generación e ingeniería reversa de código XML.

- Modelamiento de Datos: PowerDesigner diseña y genera el esquema de la base de datos a través de un verdadero modelamiento de bases de datos relacionales de dos niveles (conceptual y físico) - basado en métodos probados. PowerDesigner también soporta técnicas específicas de modelamiento para data-warehouse.

- Modelamiento de Objetos: PowerDesigner completa el análisis y el diseño usando técnicas UML estándar. A partir de un diagrama de clase, PowerDesigner automáticamente genera y realiza ingeniería de reversa de ambientes populares como Java (incluyendo EJB 2.0), XML, Servicios Web, C++, PowerBuilder, Visual Basic y más, a través de un generador personalizable.

- Repositorio Empresarial: La versión Enterprise de PowerDesigner agrega el valor de un repositorio de clase empresarial. El repositorio permite fácilmente visualizar y compartir modelos y otra información entre todos los miembros del equipo de desarrollo. El repositorio es altamente escalable y soporta seguridad basada en roles, control de versiones, búsqueda y generación de reportes.

Los equipos de proyectos comparten un diccionario central construido sobre una base de datos SQL. Es posible asegurar la consistencia de los datos accediendo al modelado de información con una hojeada al poderoso diccionario browser. Para seguridad e integridad de los datos, permite también administrar los derechos de usuarios y bloquear el acceso.

### II.1.2 APLICACIONES

PowerDesigner es una suite de aplicaciones de Sybase para la construcción, diseño y modelado de datos a través de diversas aplicaciones. Esta suite cuenta con los siguientes productos:

- PowerDesigner ProcessAnalyst. Permite analizar el flujo de datos de toda la empresa, a través de los departamentos hasta el usuario final.

- PowerDesigner DataArchitect. Provee a los diseñadores de las bases de datos una manera eficiente para la creación inteligente, depuración e ingeniería de reversa del modelado tanto conceptual como físico de los datos.

- PowerDesigner AppModeler. Permite el diseño y ajuste de los componentes de objetos y datos en aplicaciones de uso común como PowerBuilder, Power++, Visual Basic y Delphi ajustando el modelo de base de datos. Junto con la aplicación de servidor PowerDynamo (incluido) se pueden publicar las bases de datos en Internet/Intranet directamente del modelo de base de datos.

- PowerDesigner WarehouseArchitect. Provee un poderoso datawarehousing para el diseño e implementación de una base de datos. Cuenta con soporte para bases de datos tradicionales DBMS y bases de datos en plataformas de sistemas analíticos usando modelados dimensionales, esquemas de "estrella" y "nieve", particionamiento y agregación. También cuenta con un alto desempeño en el indexamiento de esquemas.

- PowerDesigner MetaWorks. Permite fácilmente ver y compartir la información del modelado de datos con una definición constante de objetos. También puede comparar y mezclar dos modelos de datos paso a paso.

- PowerDesinger Viewer. Crea reportes de los modelos físicos, conceptuales y procesos del modelado de la base de datos. También permite generar reportes para Internet en HTML. Este producto cuenta con demos directos del sitio de Sybase en Internet para su evaluación.

La familia PowerDesigner ofrece una solución de modelación completa que se puede adaptar a las necesidades de los DBA, analistas o diseñadores. Su estructura modular permite a las empresas aplicar las soluciones que necesita de acuerdo con la talla y los objetivos de sus proyectos.

Hasta la fecha, las soluciones de modelamiento de procesos de negocio no han solucionado los problemas de falta de comunicación entre los usuarios finales, los analistas de negocio y los desarrolladores de aplicaciones. PowerDesigner permite modelar y construir componentes reutilizables que eliminan esos fallos de comunicación y acortan el tiempo de desarrollo ofreciendo la posibilidad de ver todos los modelos de forma global.

Ahora, los directores de negocio pueden beneficiarse de las mismas ventajas que tienen los directores técnicos, ya que permite a todos los usuarios la posibilidad de modelar y diseñar procesos de negocio maximizando la colaboración con los departamentos técnicos.

Por lo que a titulo personal se puede decir que es una excelente herramienta para el Análisis y Diseño de Sistemas de Información. Ya que es un conjunto de herramientas que permiten la creación de DFD, modelos lógicos, físicos y la generación de Base de Datos físicamente. A su vez permite el control de todos los modelos y la generación automática de Aplicaciones en herramientas de 4GL.

Para el desarrollo del Sistema de este trabajo de Tesis, PowerDesigner permite que el diseño del sistema se desarrolle en dos niveles, los cuales se citan a continuación.

### II.1.3 NIVELES DE DISEÑO CON POWERDESIGNER

- Nivel Conceptual: Entidades, relaciones, dominios, tipos de datos conceptuales, identificadores, y reglas de negocios. Basado en la notación de ingeniería de Martin.

- Nivel Físico: Tablas, columnas, dominios, llaves primarias, llaves foráneas, llaves alternadas, índices, constraints de integridad referencial declarativa, vistas, parámetros de almacenamiento físico, reglas de negocios, triggers y procedimientos almacenados.

Finalmente cabe mencionar que Sybase PowerDesigner, soporte Modelos Funcionales y Notaciones de Diagramas de Flujo, Modelo Funcional de Objeto (OMT) Yourdon/DeMarco, Gane & Sarson, y SSADM (Análisis de Sistema Estructurado y Metodología de Diseño, Structured System Analyst & Design Methodology), entre otros.

## II.2 POWERBUILDER

PowerBuilder es una herramienta gráfica de desarrollo extremadamente flexible. Utilizando PowerBuilder es posible desarrollar poderosas aplicaciones gráficas con acceso a bases de datos. PowerBuilder proporciona todas las herramientas necesarias para la construcción de aplicaciones sólidas, tales como sistemas de facturación, de contabilidad y sistemas de manufactura.

Las aplicaciones construidas con PowerBuilder consisten de ventanas con las cuales los usuarios interactúan, se pueden utilizar todos los controles estándar tales como botones de comando, checkbox, cuadros de lista, controles de edición, etcétera; así como controles especiales propios de PowerBuilder que hacen las aplicaciones fáciles de desarrollar y fáciles de usar. PowerBuilder permite desarrollar y desplegar aplicaciones en la arquitectura adecuada para todas las necesidades, y si ésta cambia, se puede redesplegar fácilmente en una arquitectura diferente. PowerBuilder es un entorno orientado a objetos lo que significa que dispone de todas las características importantes que se encuentran en esos entornos, los cuales incluyen herencia y encapsulación. Con PowerBuilder se pueden construir componentes gracias a la gran variedad de herramientas que proporciona. PowerBuilder incluye herramientas que permiten construir aplicaciones basadas en la web que permiten extender las aplicaciones a internet.

*Plataforma de desarrollo combinada.* PowerBuilder soporta una plataforma de desarrollo y otra de despliegue. Por ejemplo se puede desarrollar utilizando PowerBuilder bajo ambiente Windows (Windows 95, Windows XP, Windows NT, etcétera) y desplegar la misma aplicación sin cambios en una máquina en un ambiente Windows 3.1, Macintosh, o UNIX. Se puede inclusive tener a un equipo de desarrolladores trabajando en una misma aplicación en una plataforma combinada al mismo tiempo, esto es, algunos utilizando Windows y otros Macintosh, ya que se pueden compartir libremente los objetos que se utilizan en la aplicación, gracias a que los objetos son los mismos a través de las diferentes plataformas de desarrollo de PowerBuilder. La mayoría de las aplicaciones desarrolladas en PowerBuilder corren bajo Windows pero la interfaz gráfica de usuario se ve y trabaja igual en todas las plataformas que soporta.

Arquitecturas que soporta PowerBuilder:

- Cliente/servidor.
- Una capa.
- De dos capas (tradicional cliente/servidor).
- Multicapa (Aplicaciones distribuidas).
- Internet.
- Web.

- PowerBuilder window plug-in.
- PowerBuilder window ActiveX.
- DataWindow plug-in.
- ActiveX automation.

PowerBuilder aún no es tan popular en México como otras herramientas debido a su precio, es mucho más fácil encontrar un libro sobre Visual Basic o Delphi que sobre PB. Sin embargo para aplicaciones Cliente-Servidor es una herramienta más completa. PowerBuilder es una herramienta de desarrollo de aplicaciones para el ambiente Windows, por lo tanto, como tal utiliza las características de este ambiente.

Haciendo énfasis en el concepto Cliente-Servidor, cabe mencionar que esta pone en comunicación una estación de trabajo con un servidor de bases de datos central. Con la disponibilidad generalizada de redes y estaciones de trabajo baratas, las aplicaciones Cliente-Servidor se hacen cada vez más populares.

Una aplicación PowerBuilder es un programa que se ejecuta en una computadora. El programa utiliza una interfaz gráfica de usuario, por ejemplo, Windows. También se puede conectar, a través de una red, a algunas fuentes de datos, por ejemplo una base de datos. El usuario interacciona con el programa a través de la interfaz gráfica de usuario, bien utilizando el teclado, o bien utilizando un dispositivo de selección tal como un ratón. Por ejemplo, el usuario comienza una aplicación de captura de datos haciendo doble clic sobre un ícono. A continuación abre una captura de datos. Esto se puede realizar a través de un menú, del teclado o de un botón de la pantalla. A partir de este momento el usuario interacciona con la aplicación a través de la ventana que se acaba de abrir. Una ventana puede tener controles tales como un botón de salida o un botón de recuperación de datos. El usuario puede pulsar con el ratón un botón de salida para cerrar una ventana, o un botón de recuperación de datos para obtener los datos a presentar. También podrá introducir datos por medio de los campos de entrada contenidos en la ventana. La aplicación puede presentar los resultados en forma de tabla, imagen o gráfico. PowerBuilder tiene todas las herramientas para realizar estas aplicaciones.

PowerBuilder es un entorno de programación que está compuesto de diferentes herramientas para el desarrollo rápido que se manejan con el ratón, y un lenguaje de programación. Está pensada especialmente para el entorno cliente servidor, a través de odbc o de controladores nativos para las bases de datos más importantes del mercado. El lenguaje de programación se llama PowerScript, y posee una implementación de la programación orientada a objetos muy completa. Con PowerBuilder es posible crear objetos genéricos, crear herencias de éstos y generar objetos más especializados, implementar la herencia múltiple a través de los objetos de servicio, y finalmente crear aplicaciones reutilizando el código al máximo.

## II.2.1 MODELO CLIENTE-SERVIDOR

Un modelo cliente-servidor tiene tres componentes: un servidor, una red y clientes. Tal y como se puede apreciar en la figura II.1.

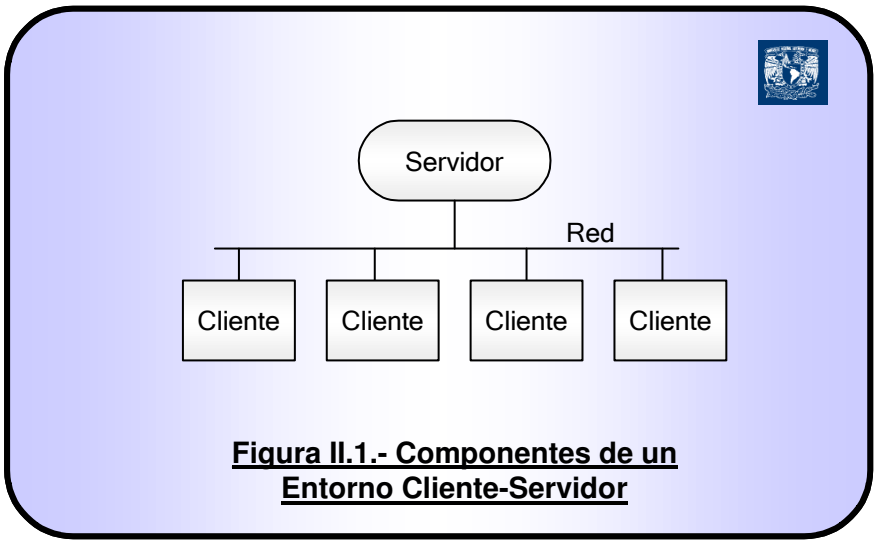



**Figura II.1.- Componentes de un Entorno Cliente-Servidor**

### II.2.1.1 Servidor

Es una computadora que almacena y gestiona datos. Por ejemplo, en el servidor se puede ejecutar un sistema de gestión de bases de datos. Este contiene datos de alguna empresa, por ejemplo, puede contener todas las facturas de la misma. El servidor es responsable de suministrar cualquier dato requerido por el usuario. El servidor obtiene los datos en función de la petición realizada y los envía al cliente. Es así mismo responsable del control de la concurrencia, del mantenimiento de la integridad de la base de datos y de la gestión de las transacciones. El servidor se responsabiliza de la seguridad de los datos y del control de accesos. Finalmente, también suministra el espacio de almacenamiento necesario para los datos.

Una máquina cliente solicita datos del servidor. La computadora cliente ejecuta un programa localmente. Este programa está preparado para solicitar adecuadamente los datos al servidor. Así mismo, es responsable de la gestión de la presentación de los datos, de la interacción con el usuario y de la operativa de recuperación de datos. La computadora cliente valida los datos introducidos por el usuario y genera las solicitudes de datos necesarias al servidor.

### II.2.1.2 Red de Computadoras

Conecta los clientes al servidor. Dos de las redes más comunes son Novell y TCP/IP. Una red Novell o TCP/IP puede conectar todas las estaciones de trabajo de una oficina al servidor de la base de datos. Esta interconexión de equipos comúnmente se lleva a cabo a través de segmentos de red definidos por una dirección ip. En específico para el caso en particular del sistema de esta tesis se usará el protocolo TCP/IP y, en situaciones muy excepcionales, tan sólo se definirían grupos de trabajo, dentro de un entorno de red.

### II.2.1.3 Cliente

Envía una solicitud de datos a un servidor. El servidor devuelve los datos seleccionados en respuesta a la solicitud. El cliente puede realizar cambios en los datos y devolverlos al servidor para incluirlos en la base de datos. La computadora cliente puede realizar la mayor parte de la manipulación o procesamiento de los datos. Por ejemplo, si el usuario está introduciendo datos, todas las validaciones y correcciones tienen lugar en el cliente. Cuando los datos son correctos, se envían a la correspondiente base de datos del servidor. El cliente puede solicitar también cambios que se realizan directamente en los datos almacenados en el servidor. Por ejemplo, el cliente puede enviar una solicitud al servidor para modificar o borrar todos los registros que están ligados a un determinado padre.

En el sistema de facturación de una compañía, la computadora central (un servidor de red) mantiene una base de datos con toda la información relativa al sistema. Este servidor de base de datos está conectado a una red. Varias computadoras personales y estaciones de trabajo repartidos por toda la compañía están también conectados a la misma red. Cada una de las estaciones de trabajo o computadoras personales son un cliente. La máquina que contiene la base de datos actúa ahora como un servidor de base de datos. Las máquinas cliente envían las solicitudes de datos a través de la red. El servidor obtiene los datos solicitados y los devuelve al cliente que los solicitó a través de la red. Una máquina cliente puede gestionar todas las interacciones con el usuario. Esto descargará la mayor parte del proceso desde el servidor al cliente.

Un programa en PowerBuilder que se está ejecutando en una máquina cliente puede presentar un determinado formato de pantalla. El usuario puede introducir datos por medio de ese formato y el programa los utiliza para hacer una solicitud de datos al servidor.

Cualquier dato utilizado por la aplicación cliente está almacenado en la base de datos del servidor. Si el empleado que introduce los pedidos quiere información sobre ellos, dicha información proviene de la base de datos del servidor y va al correspondiente cliente a través de la red. Cualquier cambio realizado en las órdenes se reenvía de nuevo utilizando la red a la base de datos del servidor.

La aplicación cliente puede presentar los datos recuperados, por ejemplo, con un determinado formato de pantalla. También puede transformar los datos y presentarlos por pantalla. La aplicación podría utilizar los datos recuperados del servidor para realizar un informe, u obtener una información resumen y presentarla con un formato normal o utilizando gráficos.

#### II.2.1.4 Datos distribuidos

En un entorno cliente-servidor, los datos pueden estar distribuidos en diferentes máquinas de una red. La facturación, las ventas y los envíos podrían estar en diferentes servidores, conteniendo cada uno de ellos la correspondiente porción de información de la empresa. Esta información puede encontrarse toda en un servidor y en una base de datos, en un servidor y en más de una base de datos, o en diferentes servidores y en diferentes bases de datos.

PowerBuilder permite construir aplicaciones que acceden a los datos desde cualquier lugar de la empresa. Múltiples conexiones concurrentes a diversos sistemas de gestión de bases de datos pueden acceder a datos en cualquier máquina. Se pueden desarrollar aplicaciones que acceden a datos remotos almacenados en mainframes, mini-computadoras o estaciones de trabajo, o acceder a datos almacenados en bases de datos locales.

#### II.2.1.5 Datos locales

Por supuesto, una aplicación PowerBuilder también puede ejecutarse en la misma máquina que contiene la base de datos. En el caso más simple, el cliente y el servidor se encuentran en la misma máquina, y por lo tanto no se necesita ninguna red. PowerBuilder se suministra con el sistema de gestión de base de datos relacional Watcom. Este puede ejecutarse en una computadora personal con Microsoft Windows. Una aplicación que se ejecuta en Microsoft Windows puede acceder a la base de datos Watcom que también se está ejecutando en Microsoft Windows.

### II.2.2 PROGRAMACIÓN ORIENTADA A OBJETOS

Este concepto es sencillo de entender si se analiza con un ejemplo, una silla es un objeto; los elementos que la caracterizan son los atributos (color, altura, etcétera). Todos estos atributos (características del objeto) pueden tomar valores dentro de un dominio definido por la característica en si (color no puede tomar el valor 1,2 metros, la altura no puede ser amarillo, etcétera).

Analizando su relación con los objetos en la programación, es de notarse que un objeto en programación es un elemento que posee características, pero más aún posee métodos (funciones) que han sido definidas para interactuar en operaciones comunes con dicho objeto. Ejemplo: buscar un registro en una lista.

Una instancia particular de un objeto se convierte en una ‘variable’, para todos los fines de la programación, sus atributos se convierten en ‘campos’ de esta.

### II.2.2.1 Tipos de Objetos

Existen dos tipos de objetos:

- VISUALES: Son aquellos que pueden ser vistos por el usuario en el monitor de su computadora. Ejemplos: Botones, ventanas, etcétera.

- NO VISUALES: Son aquellos que aunque no pueden ser vistos por el usuario, poseen todas las características de estos. Ejemplos: Errores, Objetos de Transacción (SQL).

En una aplicación desarrollada en PowerBuilder los objetos 'No visuales' son muy necesarios para que los objetos 'Visuales' interactúen con otros elementos ajenos al Sistema computacional en si.

### II.2.2.2 Atributos, Eventos, Métodos (Funciones)

Como ya se mencionó previamente un objeto esta constituido por "Atributos"; estos lo caracterizan. Existen atributos que pueden modificarse y otros no. También existen atributos que son modificables tanto durante el diseño de la aplicación en si, como durante la ejecución, y otras que solo pueden ser modificadas durante el diseño.

Los "métodos", son funciones destinadas a manipular elementos que son definidos en conjunto con el objeto, esto implica que un objeto solo puede manipular elementos que estén contenidos en él (efecto "caja negra"), por ejemplo en un objeto lista, la función buscar sólo lo hace en los registros del objeto lista. Este principio se denomina Encapsulamiento, y permite una autonomía de cada objeto con su entorno.

Los "Eventos", son cada acción que se puede realizar sobre el objeto; ejemplo: Hacer un click sobre un objeto botón, presionar una tecla al escribir en un objeto caja de texto, etcétera. Cada una de estas acciones es independiente una de otras, pero no necesariamente son excluyentes. Ejemplo: al presionar el botón se realizan los Eventos "Cerrar" de la ventana, y el evento "Destruir" también de la ventana; el primero se realiza al sacar la ventana de la pantalla y el segundo se realiza cuando se saca la ventana de la memoria principal.

También es necesario indicar el principio de la "Herencia", como en el ser humano un hijo "hereda" algunos rasgos de sus progenitores, en la programación orientada a objetos, un objeto puede "heredar" los valores dados a un atributo, junto con todas las características dadas al objeto origen (código, otros objetos contenidos dentro de este, etcétera); pero con la libertad de deshacerse de estos o añadir nuevos sin alterar al objeto original.

## II.2.3 LA INTERFAZ GRÁFICA DE POWERBUILDER

Una vez que se ha puesto en marcha el ambiente de desarrollo PowerBuilder lo primero que se visualiza es una ventana con un menú y una barra de herramientas. Con PowerBuilder es obligatorio tener seleccionada una aplicación. Antes de empezar a crear una aplicación se debe conocer como se organiza la interfaz de usuario de PowerBuilder. Por lo que a continuación se describe esa organización.

### II.2.3.1 PowerBuilder Libraries

Mejor conocida como Pbl, es el contenedor básico de objetos PowerBuilder. Cualquier objeto que se construya deberá estar contenido en una pbl o librería. Una aplicación puede estar compuesta por una o varias pbl. Lo normal es empezar con una e ir distribuyendo después los objetos en diferentes librerías a medida que estos van creciendo en número. La organización depende de las preferencias personales, se puede dividir los objetos por su clase o por su utilidad, se puede meter una aplicación entera en una pbl o crear una pbl por cada opción de menú. La práctica indicará la mejor manera de hacerlo según el tipo de desarrollo que se realice (una librería de clases, una enorme aplicación de gestión, etcétera). Una pbl incluye el código fuente del objeto y la versión compilada. De una pbl se debe generar una pbd, que es lo mismo que una pbl pero sin el código fuente, o una dll típica de windows.

### II.2.3.2 Painters

Cada vez que se requiera crear o modificar algo, se deberá abrir el painter (pintor) correspondiente. La barra de botones o herramientas (toolbar en la terminología de PowerBuilder) que aparece al arrancar PowerBuilder muestra todos los painters que pueden ejecutarse. Cada vez que pulsamos un botón de la barra de herramientas, ésta y el menú de la ventana cambiarán ofreciendo las opciones particulares del painter. Desde los painters se diseña cada uno de los componentes de una aplicación PowerBuilder, Los más importantes son:

- *Aplicación (Application):* Editar el objeto aplicación. También se especifica aquí la lista de las pbl que van a componer la aplicación.

- *Proyecto (Project):* Genera los ejecutables una vez que la aplicación está terminada.

- *Ventana (Window):* Diseño visual de las ventanas y donde se incluye la parte más importante del código de la aplicación.

- *Objeto de Usuario (User Object):* Diseño de los objetos reutilizables.

- *Menú:* Generación de menús que después se pueden asociar a una o más ventanas.

- *Estructura (Structure):* Estructuras de datos (del tipo utilizado en el lenguaje C) globales a la aplicación.

- *Función (Function):* Funciones globales a la aplicación.

- *Ventana de Datos (Datawindow):* Diseño de los formularios asociados a una consulta SQL en sus diversos formatos (Formato libre, tabular, hoja de cálculo, etcétera). Junto con el objeto Window son los más utilizados de PowerBuilder.

- *Consulta SQL (Query):* Consultas Sql que guardamos para usar varias veces (generalmente poco utilizado).

- *Tubería (Pipeline):* Traspaso de datos desde una tabla a otra en la misma base de datos o en dos diferentes. Muy útil, ya que ahorra teclear sentencias SQL demasiado extensas. Podemos utilizarlos manualmente para hacer traspasos de datos de una aplicación antigua a otra nueva o llamarlo desde un programa PowerBuilder que realice traspasos automáticos.

- *Base de Datos (Database):* Diseño de la base de datos. Se puede prescindir de este painter y utilizar las herramientas propias de la base de datos.

- *Librería (Library):* Crear librerías, mover objetos de una a otra, borrar objetos, recompilar el código cuando ha habido modificaciones en herencias intermedias.

Crear una aplicación en PowerBuilder consiste en crear cada una de sus piezas con el painter adecuado, y después codificar las interacciones entre ellas. El orden lógico de esta creación generalmente es la siguiente:

1º. Objeto Aplicación.

2º. Menús.

3º. Ventana Marco Mdi Principal.

4º. Subventanas Mdi.

5º. Datawindows.

## II.2.4 POWERSCRIPT

Para comenzar a programar en PowerBuilder, se requiere conocer o aprender el PowerScript, el cual es muy extenso y por ende resulta difícil recordar todas las instrucciones necesarias para poder realizar programas más o menos complejos. El archivo de ayuda no está organizado para esa fase del aprendizaje, aunque más adelante su uso nos será de mucha utilidad. El PowerScript maneja principalmente funciones, por lo que a continuación se citan las más importantes.

### II.2.4.1 Pronombres

- *This:* Referencia al objeto actual.
- *Super:* Referencia al objeto del que hereda el actual.
- *Parent:* Referencia al objeto que contiene el actual.
- *ParentWindow:* Referencia a la ventana que contiene el menú (solo aplica para menús).

### II.2.4.2 Variables y Tipos de Datos

Existen muchísimos tipos de datos en PowerBuilder, uno por cada objeto. Así que se puede declarar una variable del tipo de un objeto PowerBuilder (window, datawindow, application, etcétera). También existen los tipos básicos que son los típicos de cualquier lenguaje de programación: String, integer, long, decimal, real y double son los más utilizados. Existen diversos tipos de variables según el alcance de su visibilidad.

La visibilidad a nivel de clase se debe declarar escribiendo la palabra que la define (protected, private o public) seguida de dos puntos. Por defecto todas las variables son públicas.

Las variables locales a un script (las que solo son visibles dentro de un evento o función) se declaran directamente en el código del script y no utilizan modificador de visibilidad a nivel de clase ya que están sujetas a la visibilidad del script que las contienen.

## II.2.5 VENTANA

Las funciones más usadas en este tipo de objeto son:

- *Open:* Abrir una ventana.
- *OpenSheet:* Abrir una ventana dentro de otra ventana del tipo mdi.

- *OpenWithParm:* Abrir una ventana enviándole un parámetro. Para recoger los parámetros en una ventana que fue abierta con OpenWithParm(parámetro), se utiliza el objeto Message de la siguiente manera:

  *Variable = Message.doubleparm:* Valores numéricos.

  *Variable = Message.StringParm:* Cadenas de caracteres.

  *Variable = Message.PowerObjectParm:* Cualquier objeto de PowerBuilder, incluyendo los UserObjects. Útil para enviar estructuras complejas como parámetro.

  Esta debe ser la primera instrucción que se codifique en el evento open de la ventana que debe recoger el parámetro.

- *OpenSheetWithParm:* Abrir una ventana dentro de otra del tipo mdi enviándole un parámetro.

- *Close:* Cerrar una ventana. Close(ParentWindow) y Close(Parent) son válidas.

## II.2.6 VENTANA DE DATOS

Una Ventana de Datos o DataWindow es un objeto que contiene (entre otras cosas) una instrucción Select de SQL (comúnmente), y una representación visual de los datos que trata la misma. El programador proporciona al DataWindow la instrucción select, y el objeto es capaz de generar las instrucciones insert, update y delete por si mismo, cuando sea oportuno. Es necesario conocer el sistema de buffers que utiliza la datawindow para comprender como se ha de utilizar:

Cuando ejecutamos un Retrieve(), se lanza el select a la base de datos y las filas devueltas se almacenan en un buffer de memoria que es lo que el usuario visualiza y modifica. A diferencia de otras herramientas windows (como Microsoft Access o Borland Paradox), por mucho que el usuario manipule los datos, las modificaciones no se graban en la base de datos hasta que el programa no ejecute una instrucción update (mediante la función update()). Es entonces cuando el DataWindow genera las instrucciones insert, update y delete según lo que el usuario haya hecho. Ya que se trabaja con una copia de los datos en memoria y no con SQL. El DataWindow mantiene varios buffers de forma paralela y algunas funciones pueden recibir un parámetro que indica sobre que buffer deseamos operar.

### II.2.6.1 Buffers del DataWindow

- *Primary!:* Es el buffer por defecto. El que el usuario modifica con el teclado.

- *Original!:* Es el buffer que guarda los datos tal y como se recuperaron de la base de datos.

- *Deleted!:* Guarda las filas borradas.

- *Filtered!:* Cuando aplicamos un filtro al DataWindow, aquí se almacenan las filas que no se ven, y que han desaparecido del buffer Primary!.

La utilidad de estos buffers, estriba en que se pueden mover filas de uno a otro, y acceder y modificar a los datos de cada uno de ellos (con ciertas restricciones). Ejemplo: Cuando un usuario borra una fila, esta no desaparece de la memoria, sino que simplemente se mueve del buffer Primary al Deleted, Así que si se desea se puede recuperar sin acceder a la base de datos.

Por otra parte también está el estado o estatus de las filas/columnas del DataWindow. Este estado permite a PowerBuilder conocer que instrucciones ha de generar cuando se invoque un Update(). En un principio después de ejecutar un Retrieve todas las filas tendrán el estado NotModified. Si se modifica alguna pasará al estado DataModified. Si se borra alguna fila pasará a ser Deleted. Al insertar una nueva su estado será New, y al introducir un valor en la nueva fila se convertirá en NewModified. Una fila con estado New, no se grabará en la base de datos. Todos estos estados son muy útiles cuando se quiere validar los datos antes de grabar y se requiere saber distinguir entre los datos nuevos y los que se han modificado. PowerBuilder es una herramienta muy potente, lo malo es que esa potencia lo convierte en complejo y a veces el que no conoce esta herramienta de desarrollo puede pensar que muchas cosas no se pueden hacer. Normalmente se puede hacer de todo, pero a veces de forma algo diferente a como se ha hecho con otras herramientas.

El último buffer a tener en cuenta es el buffer "Text". Este buffer es de tipo string y contiene la cadena del campo que está siendo modificado en ese momento. Cuando el usuario cambia de campo con el ratón o con la tecla Tab, el campo se valida y pasa a la respectiva columna del DataWindow, a partir de entonces ese valor toma el tipo de la columna (date, number, etcétera). Sin embargo hasta ese momento es de tipo string. Se debe tener cuidado al acceder a los datos de la DataWindow, si el usuario no pulsó la tecla de Tabulador, el dato todavía no está en la columna.

### II.2.6.2 Funciones del DataWindow

Las Funciones más usadas en el DataWindow son:

- *AcceptText:* Dispara la validación de los campos o columnas independientemente de si el usuario pulsó o no la tecla de Tabulador.

- *DBCancel:* Provoca la cancelación de la función Retrieve() en curso.

- *DeletedCount:* Obtiene el número de filas que el usuario ha borrado.

- *DeleteRow:* Borra una fila.

- *Describe:* Obtiene información muy detallada sobre la estructura del DataWindow.

- *Filter:* Ejecuta el filtro actual.
- *GetChild:* Obtiene un apuntador a un DataWindow contenido dentro de otro.
- *GetColumn:* Obtiene el número de columna actual.
- *GetColumnName:* Obtiene el nombre de la columna actual.
- *GetItemxxx:* xxx representa un tipo de dato. Devuelve el valor contenido en una columna.
- *GetItemStatus:* Obtiene el status de una fila o columna.
- *GetRow:* Obtiene el número de fila actual.
- *GetSQLSelect:* Obtiene el select del DataWindow.
- *GetText:* Obtiene una cadena de caracteres que contiene el valor de una columna que esta siendo editada antes de que el usuario la confirme cambiando de campo.
- *GetValue:* Obtiene una cadena de caracteres conteniendo el valor de una columna.
- *InserRow:* Inserta una fila en blanco.
- *Modify:* Modifica la estructura interna de un DataWindow.
- *Print:* Imprime un DataWindow.
- *Retrieve:* Ejecuta un Select contra la base de datos y muestra los datos en el DataWindow.
- *RowCount:* Cuenta las filas en un DataWindow.
- *SaveAs:* Almacena el contenido de un DataWindow en un archivo. Soporta diversos formatos incluyendo excel, texto, rtf y html.
- *ScrollToRow:* Desplaza el foco y el control del DataWindow hasta una fila.
- *SetColumn:* Cambia el foco a una columna en concreto.
- *SetFilter:* Modifica las condiciones de filtro de un DataWindow.
- *SetFocus:* Coloca el foco en un DataWindow.
- *SetItem:* Modifica el valor de una columna.

- *SetItemStatus:* Modifica el status de una fila o columna.
- *SetRow:* Hace que la fila especificada sea la actual.
- *SetSort:* Modifica las condiciones de ordenamiento.
- *SetTransObject:* Enlaza un DataWindow con el objeto transacción para la conexión con la Base de Datos.
- *ShareData:* Hace que un DataWindow comparta datos con otro.
- *Sort:* Ordena un DataWindow con las condiciones especificadas con SetSort().
- *Update:* Graba las modificaciones en la base de datos.

Hay muchas más funciones pero estas son las básicas para el manejo fundamental de los DataWindows.

### II.2.6.3 Eventos del DataWindow

Los eventos son como funciones predefinidas que se disparan automáticamente cuando ocurre el suceso que se relaciona con el mismo. En los eventos se coloca el código que se desea se ejecute cuando ocurra la acción que define el evento. Se pueden crear eventos de usuario, pero la llamada se debe realizar a través de código, pues los eventos predefinidos son llamados por si mismos. Los eventos reciben parámetros. Estos parámetros proporcionan la información relevante sobre las condiciones en que se produjo el evento. Ejemplo: En el evento ItemChanged, que ocurre cuando un campo cambia de valor por acción del usuario, el parámetro row proporciona el número de fila donde ocurrió el cambio. También los eventos pueden retornar valores que en general sirven a PowerBuilder para saber si el resultado del evento fue correcto. Ejemplo: Si en el evento itemchanged finaliza este con un return 2, PowerBuilder impide que el campo actual cambie de valor. Esto es útil cuando se quiere validar un campo introducido por el usuario. Los eventos más importantes del objeto DataWindow son: Clicked, Constructor, DbError, Destructor, EditChanged, Error, ItemChanged, ItemError, ItemFocusChanged, RetrieveEnd, RetrieveRow, RetrieveStart, RowFocusChanged, SqlPreview, UpdateEnd y UpdateStart.

## II.2.7 PFC POWER BUILDER FOUNDATION CLASS

Mediante la utilización de PowerBuilder se puede identificar tres arquitecturas de programación: Orientada a Eventos, Orientada a Objetos sin manejo de PFC's y Orientada a

Objetos con PFC's. Dado que el sistema de esta tesis se desarrollará mediante la utilización de esta última modalidad, sólo se hará referencia a la misma.

Se dice que la arquitectura de las PFC, o clases fundamentales de PowerBuilder, está basada en los servicios (Service-Based Architecture, SBA). Como las clases de Powersoft están escritas en PowerBuilder y además el usuario recibe el código fuente, cada uno puede personalizarlas y mejorarlas a su gusto fácilmente. Usar las PFC reporta muchos beneficios a la hora de construir aplicaciones PowerBuilder. Estos son algunos de ellos:

- Las aplicaciones se desarrollan más rápidamente.
- Pueden emplearse muchos componentes construidos de antemano en lugar de tener que reinventarlos cada vez.
- Aumenta la cantidad de código reutilizable, al emplear componentes preconstruidos y verificados.
- Convención de nombres consistente y estandarización del código.
- El mantenimiento de las aplicaciones es más barato y más sencillo.
- Metodología estandarizada de desarrollo.

Las PFC se distribuyen como un conjunto de clases agrupadas en cuatro bibliotecas PowerBuilder. Acompaña a estas clases un entorno de trabajo compuesto por ventanas, menús, objetos de usuario y objetos DataWindow. Además se incluye también una relación de clases de servicios implementadas como objetos de usuario de clases personalizadas no visuales.

PFC es considerado como el mayor avance en el uso de la orientación a Objetos. Una PFC proporciona a los desarrolladores de PowerBuilder, una librería de clases robusta que utiliza las últimas características y arquitecturas disponibles y que le permite construir Sistemas de Clase Internacional en menor tiempo y con mayor calidad. Las PFC's ofrecen al desarrollador una forma efectiva de utilizar y extender la funcionalidad contenida en la librería de clases. Los objetivos de estas librerías son:

- Familiarizar en los conceptos y características de las PFC's.
- Optimizar el reuso con la utilización de las clases de las PFC's.
- Mejorar dramáticamente la calidad del desarrollo de Sistemas.
- Mejorar la productividad de los desarrolladores y reducir costos.
- Construir aplicaciones usando PFC´s.

- Personalizar los objetos de las PFC's para cumplir con los requerimientos de su aplicación.

### II.2.7.1 Introducción a la Arquitectura basada en los Servicios (SBA)

Con una arquitectura basada en los servicios, el diseño y desarrollo de los complejos comportamientos de los componentes necesarios en una aplicación se lleva a cabo de distinta manera. Para entender más claramente la arquitectura basada en los servicios puede ser útil aprender un poco más sobre las arquitecturas que PowerBuilder ha venido utilizando desde su aparición en 1991. La arquitectura basada en los servicios es simplemente el último paso de esta evolución.

PowerBuilder se ha basado desde sus primeros días en una arquitectura interna orientada a objetos. Sin embargo, los usuarios de la primera versión de PowerBuilder no pudieron disfrutar aún de características tales como la herencia, el polimorfismo y la encapsulación. Las aplicaciones de PowerBuilder 1.0 se codificaban casi por completo desde el pintor de ventanas. Muchos usuarios de PowerBuilder 1.0 confiaban en exceso en las técnicas de copiar, pegar, copiar código y ventanas enteras para incrementar la productividad.

Desde el punto de vista del autor, muchos de los hábitos de programación incorrectos tienen su origen en estos primeros días. Estos fueron los comienzos de la arquitectura de «ventanas gordas».

Con PowerBuilder 2.0 aparecieron la herencia y la encapsulación. Por primera vez una ventana podía heredar el comportamiento de su antecesor. Gracias a la herencia, los programadores podían crear nuevas ventanas mucho más rápidamente. Echando la vista atrás, es posible acordarse de los esfuerzos que se hacían para incluir en la clase antecesora toda la cantidad de código posible para alcanzar la máxima reutilización y hacer extensible el comportamiento heredado a las ventanas descendientes. Los primeros pasos que se daban en la construcción de ventanas antecesoras requerían muchos refinamientos, ya que cada ajuste incidía en el resto de ventanas descendientes. Comenzaron a aparecer entonces las bibliotecas de clases compuestas principalmente de plantillas para ventanas. La arquitectura de «ventanas gordas» sobrevivió un año más.

Lo que caracterizó a PowerBuilder 3.0 fue la aparición de los objetos de usuario. Los objetos de usuario visuales y no visuales hicieron que muchos programadores revisaran las bases de la arquitectura básica de sus aplicaciones. Transfiriendo parte del código de las ventanas a objetos de usuario visuales se podía comenzar a simplificar la intrincada jerarquía de las ventanas. Según las ventanas antecesoras fueron desapareciendo, los controles DataWindow antecesores se fueron convirtiendo en objetos más grandes y complejos. La arquitectura de "ventanas gordas" dio paso a la de "DataWindow gordas".

La aparición de los objetos no visuales en PowerBuilder 3.0, denominados comúnmente NVOs, hizo posible la aparición de la arquitectura basada en los servicios. En PowerBuilder 4.0, los objetos de usuario no visuales pasaron a denominarse clases personalizadas. Dicho sencillamente, una arquitectura basada en los servicios implica transferir algunos de los

comportamientos de un objeto, tales como una ventana o un control DataWindow, a un objeto de una clase personalizada asociado. La clase personalizada pasa a ser el proveedor de servicios del objeto, mientras que la ventana o la DataWindow se convierten en el solicitante del servicio.

### II.2.7.2 El Objetivo de las arquitecturas basadas en los Servicios

La característica más importante de la arquitectura basada en los servicios es la división del comportamiento de los objetos de PowerBuilder en objetos de usuario de clases personalizadas no visuales. Esta división permite reducir el tamaño de los objetos de PowerBuilder tanto en el desarrollo como en la ejecución del programa. Las pruebas de rendimiento demuestran que un objeto de gran tamaño puede influir significativamente en el rendimiento del programa.

Los comportamientos relacionados se implementan en un mismo proveedor de servicios o bien en una familia de clases personalizadas de proveedores de servicios. Los servicios que no están relacionados entre sí se organizan en proveedores de servicio independientes. Esta nueva forma de dividir u ordenar el código permite entender más fácilmente cómo está implementado un servicio, ya que sus distintos componentes están agrupados en un solo objeto proveedor de servicios.

Los proveedores de servicios aumentan la cantidad de código reutilizable y reducen el mantenimiento innecesario de código. Cualquier modificación en un servicio está limitado a un solo objeto y es menos probable que se produzcan errores en otros proveedores de servicios.

El tiempo de desarrollo debe verse reducido, ya que las verificaciones están circunscritas a un objeto específico. Ya no es necesario pasar el depurador por todo el código generado en el solicitante del servicio.

La arquitectura basada en los servicios se fundamenta más en los componentes que la mayoría de las arquitecturas utilizadas en las bibliotecas de clases de PowerBuilder creadas por terceros. Es posible seleccionar e implementar servicios individuales según el modelo plug-and-play. Puede prescindirse de los servicios que no se quieren emplear.

La arquitectura basada en los servicios constituye el último paso en la evolución del desarrollo orientado a objetos de PowerBuilder. Entender las bases de la arquitectura basada en los servicios es esencial para comprender la arquitectura subyacente de la biblioteca PowerBuilder Foundation Classes. En un futuro próximo se podrá comprobar cómo aumenta el uso de las arquitecturas basadas en los servicios.

Como puede notarse PowerBuilder tiene todas las herramientas necesarias para construir aplicaciones cliente-servidor. Permite realizar aplicaciones que se ejecutan en un computador cliente, comunicándose entre sí utilizando protocolos de red estándar. Las aplicaciones pueden acceder a datos locales o remotos provenientes de una serie de fuentes. En los servidores se pueden ejecutar la mayor parte de los sistemas de bases de datos más importantes. Los servidores pueden ser computadores personales con Novell y Microsoft SQL, estaciones de trabajo o mini-computadores con Oracle, Informix o Sybase o Mainframes con DB2.

# II.3 SQL-SERVER

Si bien, la intención de esta tesis, no es el exponer o citar un amplio panorama de un Manejador de Bases de Datos (RDBMS), si es importante mencionar que para la correcta creación, administración y mantenimiento de las Tablas, Disparadores, Procedimientos Almacenados, etcétera, se ha de utilizar SQL-Server, específicamente el Administrador de Entorno de Base de Datos (Enterprise Manager), así como el Analizador de Sentencias SQL (Query Analizer). No obstante, a continuación se da una breve descripción de las bondades que se obtienen al utilizar el RDBMS SQL-Server sobretodo en su última versión.

## II.3.1 SQL-SERVER 2000

Es el último lanzamiento de los productos de bases de datos de Microsoft, que aprovecha la sólida base establecida por SQL-Server 6.5 y SQL-Server 7. Es el RDBMS ideal para un amplio espectro de clientes corporativos y productores independientes de software (ISV) inmersa en la creación de aplicaciones empresariales. Las necesidades y requisitos del cliente han dado lugar a innovaciones significativas en el producto SQL-Server versión 2000, entre las que se incluyen la facilidad de uso, escalabilidad y fiabilidad, y almacenamiento de datos.

## II.3.2 OBJETIVOS DE DISEÑO DE SQL-SERVER

### II.3.2.1 Liderazgo e Innovación

Las innovaciones permiten a SQL-Server 2000 liderar algunas de las categorías de aplicaciones de más rápido crecimiento dentro del sector de las bases de datos. Entre estas categorías se pueden mencionar el comercio electrónico, informática móvil, automatización de sucursales, aplicaciones de líneas de negocio y depósitos de datos. Entre las importantes áreas de liderazgo e innovación de SQL-Server 2000 cabe citar:

- Primera base de datos que se amplía desde los portátiles a la empresa mediante el mismo código base y que ofrece una compatibilidad del código al 100%.
- Primera base de datos que soporta la configuración automática y la auto-optimización.
- Primera base de datos con un servidor OLAP integrado.
- Primera base de datos con los servicios de transformación de datos (Data Transformation Services, DTS) integrados.

- El marco de almacenamiento de datos de Microsoft (Data Warehousing Framework) constituye el primer planteamiento de amplia cobertura para la resolución de los problemas que plantea la utilización de metadatos.

- La primera base de datos que ofrece administración multiservidor para un gran número de servidores.

- Una gran variedad de opciones de duplicación de cualquier base de datos.

- La mejor integración con la familia Windows NT Server, Microsoft Office y BackOffice.

- Acceso universal a los datos (Universal Data Access), la estrategia de Microsoft para permitir el acceso de alto rendimiento a una gran cantidad de fuentes de información.

### II.3.2.2 Facilidad de uso

Los clientes buscan soluciones a los problemas de la empresa. La mayor parte de las soluciones para bases de datos simplemente implican nuevos costos y complejidad añadida. La estrategia de Microsoft estriba en convertir a SQL-Server en la base de datos que permita llevar a cabo la creación, administración y distribución de las aplicaciones empresariales de la forma más sencilla. Esto significa proporcionar a los desarrolladores un modelo de programación simple y rápido, eliminar la necesidad de administrar la base de datos en las operaciones habituales y proporcionar herramientas sofisticadas para acometer las operaciones más complejas.

SQL-Server 2000 reduce el costo total de propiedad mediante opciones tales como la administración de varios servidores con una única consola; ejecución de trabajos basados en eventos y generación de alertas; seguridad integrada y procedimientos de comandos para realizar tareas administrativas. Esta versión también deja vía libre al administrador de la base de datos para llevar a cabo trabajos más sofisticados al automatizar las tareas rutinarias. Mediante la combinación de estas potentes utilidades para la administración con las nuevas opciones de configuración automática, SQL-Server 2000 constituye la opción ideal para las aplicaciones de automatización de sucursales y de bases de datos incrustadas.

### II.3.2.3 Ampliable y fiable

Los clientes invierten en sistemas de administración de bases de datos en forma de aplicaciones escritas para sus bases de datos y también en la formación que conlleva su administración y despliegue. Esta inversión debe estar protegida: a medida que el negocio crece, la base de datos debe crecer para tratar más datos, transacciones y usuarios. Los clientes también

desean proteger su inversión cuando llevan las aplicaciones de base de datos a equipos portátiles o a sucursales.

Para satisfacer estas necesidades, Microsoft ofrece un único motor de base de datos ampliable desde un equipo portátil que ejecuta el sistema operativo Windows 95 o Windows 98, hasta clusters multiprocesador simétricos de varios terabytes de información y que ejecutan Windows 2000 Server Enterprise Edition. Todos estos sistemas mantienen la seguridad y fiabilidad que exigen los sistemas empresariales críticos.

Una novedad de la versión 2000 es su diseño para cubrir las necesidades cada vez mayores del mercado de la informática móvil, con nuevas e innovadoras funcionalidades como un pequeño espacio físico para la memoria, ajuste automático y duplicación en varias instalaciones.

### II.3.2.4 Almacenes de datos

Los sistemas de proceso de transacciones siguen siendo un componente fundamental de las infraestructuras de bases de datos corporativas. Las empresas también realizan grandes inversiones en mejorar el conocimiento de sus datos. La estrategia de Microsoft consiste en reducir el costo y la complejidad del almacenamiento de datos al tiempo que pone la tecnología al alcance de un mayor número de personas.

Microsoft ha establecido un planteamiento de amplia cobertura para el proceso completo del almacenamiento de datos. El objetivo es facilitar aún más la creación y el diseño de soluciones de soluciones económicas de almacenamiento de datos mediante la combinación de tecnologías, servicios y alianzas entre fabricantes.

La Microsoft Alliance for Data Warehousing es una coalición que reúne a los líderes del sector en el almacenamiento de datos y aplicaciones. El marco de almacenamiento de datos de Microsoft (Microsoft Data Warehousing Framework) es un conjunto de interfaces de programación que ha sido diseñado para simplificar la integración y administración de soluciones de almacenamiento de datos.

Entre las innovaciones que se incluyen en SQL-Server 2000 destinadas a mejorar el proceso de almacenamiento de grandes cantidades de datos, se encuentran:

- "Plato", un componente primordial para las soluciones empresariales que requieran proceso analítico en línea (Online Analytical Processing, OLAP), desde la generación de informes y análisis corporativos hasta el modelado de datos y el soporte en la toma de decisiones.

- Data Transformation Services (Servicios de transformación de datos) para importar, exportar y transformar datos.

- Mejoras en el tratamiento de las consultas complejas y bases de datos de gran tamaño (VLDB).
- Microsoft Repository (Depósito de Microsoft), una infraestructura común para compartir la información.
- Herramientas visuales de diseño para crear y mantener los diagramas de bases de datos.
- Duplicación integrada, que incluye la actualización en varias instalaciones, para mantener almacenes de datos dependientes.
- Integración de soluciones de terceros.

### II.3.3 ENTERPRISE MANAGER

Es la principal herramienta administrativa de SQL-Server y provee una consola de Administración (Microsoft Management Console, MMC). Es una interfaz de usuario amigable que permite a los usuarios realizar lo siguiente:

- Definir grupos de servidores que corren con SQL-Server.
- Registrar servidores individuales en un grupo.
- Configurar todas las opciones de SQL-Server para cada servidor registrado.
- Crear un administrador para todas las Bases de Datos en SQL-Server, objetos, conexiones, usuarios y permisos para cada servidor.
- Definir y ejecutar todas las tareas administrativas de SQL-Server en cada uno de los servidores registrados.
- Diseñar y verificar el correcto funcionamiento de sentencias o declaraciones SQL, procesos por lotes y rutinas interactivas a través de la herramienta Analizadora de Consultas (Query Analyzer).
- Invoca los distintos asistentes definidos para SQL-Server.

Para el caso del este trabajo de tesis, se utilizará para la correcta administración de los objetos de base de datos del sistema, (conocido en el medio informático como Backend).

## II.3.4 QUERY ANALYZER

El analizador de consultas de SQL, es una herramienta gráfica que habilita un administrador o desarrollador de Base de Datos para escribir sentencias o declaraciones SQL, ejecuta múltiples sentencias (querys) simultáneamente, visualiza resultados, analiza la sintáxis de la sentencias y recibe asistencia para mejorar el funcionamiento o performance de la sentencia o query.

Si bien es cierto, que no se profundiza sobre el funcionamiento de las dos herramientas mencionadas con anterioridad, si cabe mencionar que sólo servirán como complemento para el desarrollo del sistema, de ahí que se realice un breve comentario sobre la importancia de estas dado el propósito de su existencia.

# *TEMA III: Análisis y Diseño*

OBJETIVO: Identificar los requerimientos y con estos llevar a cabo el análisis y diseño del Sistema.

# III.1 EL PROCESO GENERAL DE PRODUCCIÓN

Si bien es cierto que este trabajo de Tesis no se basa exclusivamente en la logística de una empresa específica en cuanto a su proceso de fabricación o ensamblado, sí se han recabado experiencias de aquí y de allá para darle forma a un seguimiento de producción lo más básico posible, el cual, obviamente sería imposible ajustarlo al 100% con la forma de trabajar de todas las empresas que elaboren bienes materiales, no obstante se pretende cubrir o abarcar los aspectos más puntuales en todo proceso de producción, tal como la lista de materiales, orden de producción, plan de producción, análisis de producción, entre otros.

## III.1.1 FLUJO GENERAL

A grandes rasgos se puede definir como los pasos básicos de cualquier industria de ensamble o fabricación de bienes, a los citados en la figura III.1.

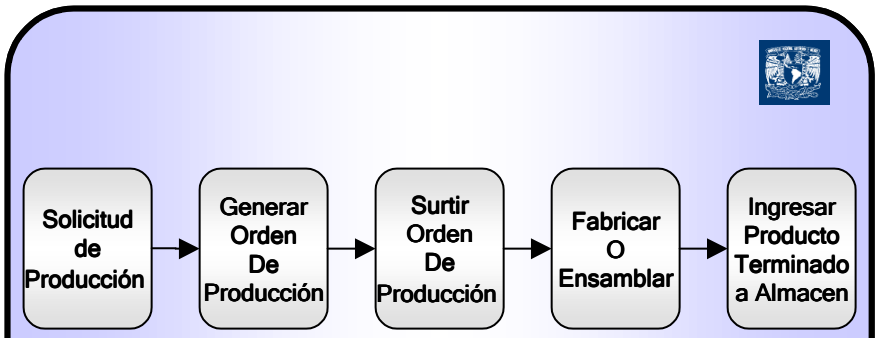



**Figura III.1.- Proceso General de Producción**

Si bien es cierto que a una solicitud de producción le puede anteceder un plan de producción, las etapas mencionadas en la figura anterior, son las que conforman un proceso estándar de producción para su control de acuerdo al objetivo de esta Tesis.

## III.1.2 FLUJO INSUMO - PRODUCTO

Dado que un sistema es un ente que recibe algo, lo procesa y produce algo, un sistema de producción se puede representar de acuerdo a lo mostrado en la figura III.2.

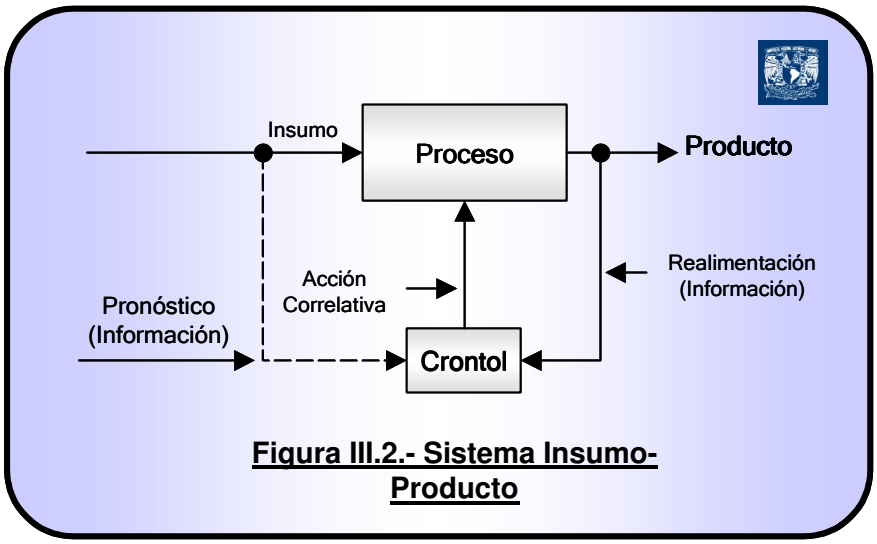



**Figura III.2.- Sistema Insumo-Producto**

Donde Pronóstico y Realimentación son las dos funciones que dan información adecuada para compararlas con los patrones de comportamiento preestablecidos, los cuales permiten tomar las decisiones correctivas necesarias. Por ejemplo, si en una empresa la demanda de su producto aumenta en forma apreciable, la función de pronóstico debe proveer la información para detectar este fenómeno oportunamente y para tomar la decisión de aumentar la producción convenientemente. Por otra parte, la realimentación nos da información, tanto de la cantidad que se está produciendo, como de la calidad y del servicio a clientes; con lo cual se pueden tomar las decisiones correctivas necesarias.

Un sistema a su vez puede estar compuesto por subsistemas, los cuales tienen el mismo flujo estándar mostrado en la figura III.2.

# III.2 RECOPILACIÓN DE NECESIDADES ESPECÍFICAS

Se requiere desarrollar un sistema de producción que permita llevar a cabo el control del proceso de producción, así como el control de insumos y costos, en base a entrevistas realizadas en empresas que se dedican a fabricar bienes, los requerimientos fundamentales que cubren sus necesidades de administración de producción. A continuación se citarán dichos requerimientos, no obstante cabe aclarar que se han definido los más básicos, por lo que en la medida que se avance en el desarrollo del sistema pueden surgir nuevos requerimientos o datos adicionales que originalmente no se consideraron, obviamente la premisa es que todo esté considerado en el Análisis y Diseño del sistema.

## III.2.1 LISTA MAESTRA DE COMPONENTES O EXPLOSIÓN DE MATERIALES

Llamada también estructura del producto o lista de partes para ensamble. Describe cómo se hace un producto a partir de sus partes y ensambles. En la forma modular más común, la lista de materiales de un artículo de inventario (llamado *Padre*) indica precisamente los componentes *inmediatos* requeridos para producir una unidad padre tal como se muestra en la figura III.3.

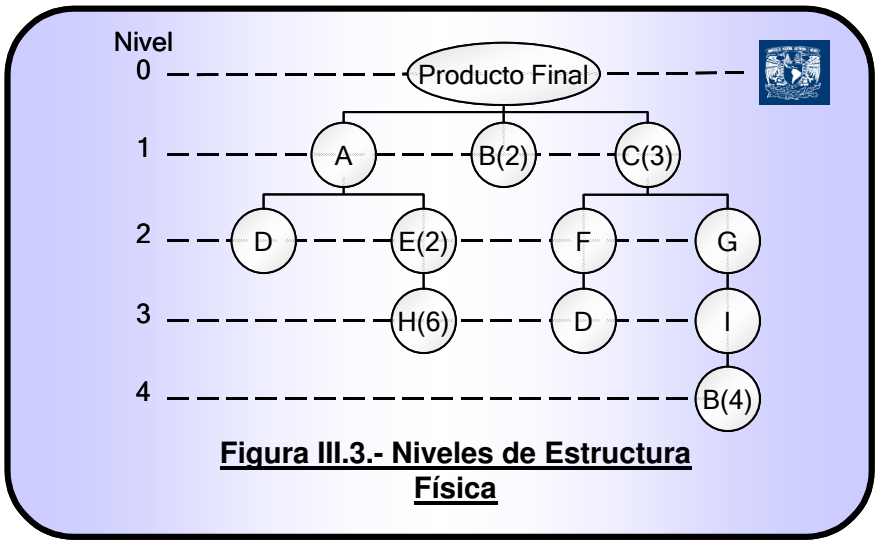



**Figura III.3.- Niveles de Estructura Física**

Por ejemplo, un producto terminado podría ser el resultado del ensamble de una unidad del componente A, dos unidades de B y tres unidades de C. El producto terminado es en este caso el padre. Al nivel siguiente, el componente A sería el padre y su lista modular de materiales indicaría los componentes necesarios para producir una unidad A. En la figura III.3 y III.4 el componente A requiere una unidad del subcomponente D y dos unidades del subcomponente E.

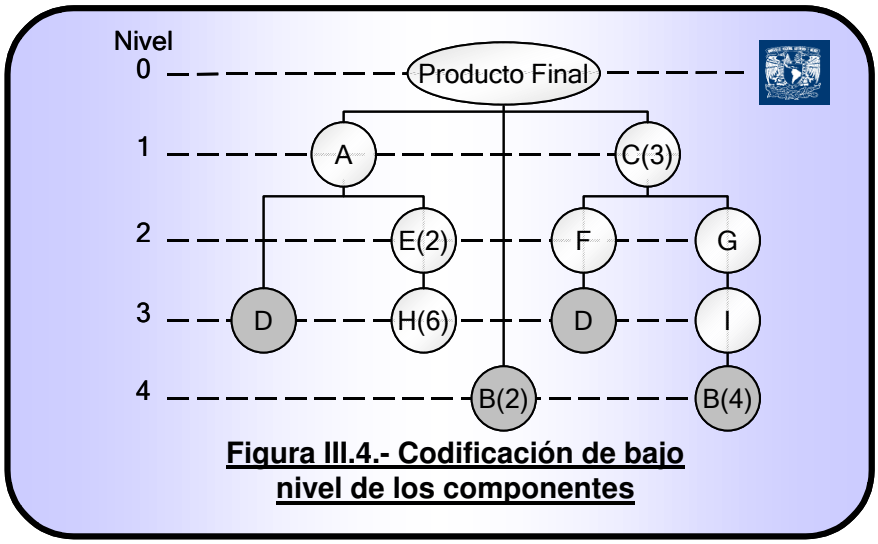



**Figura III.4.- Codificación de bajo nivel de los componentes**

### III.2.1.1 Clave de niveles

A cada lista maestra de Materiales se le asigna una clave de nivel de acuerdo con la situación respecto del producto final:

*Nivel 0.* Un producto terminado, que no se usa como componente de ningún otro.

*Nivel 1.* Un componente de un padre de nivel 0. Si un componente es al mismo tiempo un producto terminado que se vende a los clientes y un componente de otro producto terminado, se le clasifica al nivel más bajo. A este procedimiento se le llama *codificación de bajo nivel* y sirve para aumentar la eficiencia del procesamiento de datos.

*Nivel n.* Una parte a nivel n es un componente de un padre a nivel n-1. Cuando la parte existe a dos niveles se le clasifica al nivel más bajo de conformidad con el código de bajo nivel.

### III.2.1.2 Requerimiento en Sistema

Se debe elaborar un programa para el mantenimiento de las listas maestras o explosión de materiales, el cual permita generar nuevas listas maestras, realizar modificaciones a nivel de componentes, para cambio o eliminación de los mismos, o modificación de cantidades de componentes a utilizar.

Los datos que debe considerar la Lista Maestra son:

- Clave y Descripción del Componente.
- Tipo de Material.
- Indicar si es un componente principal u opcional.

- Cantidad.
- Unidad de Medida.
- Indicar si se considerará su Merma.
- Indicar si será Serializable.
- Estatus del registro.

### III.2.2 ORDEN DE PRODUCCIÓN

Es el registro del artículo a producir, donde se indica los datos más importantes para proceder a su elaboración, tales como nombre y clave del producto terminado, fecha de generación, número de Plano de Diseño (opcional), número de revisión, entre otros datos. Además de estos datos generales también contiene un listado de los componentes o partes que conformarán el artículo, la información de este listado es la clave y descripción del componente, unidades a utilizar del componente, indicativo si es fabricada o comprada (opcional). La Orden de Producción deberá consistir de los Siguientes Estatus o Etapas:

- Generada. (Nueva Orden de Producción).
- Cancelada. (Cancelada como una Orden de Producción No Válida).
- Surtida. (Componentes e Insumos suministrados por el Almacén de Materia Prima).
- En Producción. (La Cantidad de Producto Final indicado en la Orden de Producción se está fabricando actualmente). A su vez se registrará el Avance fase a fase del Proceso definido en la Generación de la Orden de Producción).
- Terminada. (Cuando la cantidad de productos terminados indicados en la Orden de Producción, han terminado de fabricarse, o bien cuando personal autorizado determina como Terminada a la Orden de Producción aún cuando no se ha completado la cantidad de productos a fabricar).

### III.2.2.1 Requerimiento en Sistema

Se debe elaborar un programa que permita generar la orden de producción a partir de la lista maestra de componentes (explosión de materiales) para el producto indicado, considerando como datos generales los siguientes:

- Folio de la Orden Producción.
- Estatus.
- Fecha de Generación.
- Fecha de Fabricación.
- Clave y descripción del producto terminado o a fabricar.
- Turno.
- Línea de Ensamble.
- Tipo de Producción.
- Cantidad a producir.
- Costo Inicial.
- Costo Final.
- Indicar si se considerará su Merma al Costo Final.
- Indicar si será Serializable.
- Número de Lote.
- Estatus.

Y como datos específicos, la lista maestra de componentes "explotada", de tal manera que permita visualizar:

- Clave del componente.
- Cantidad Requerida y Cantidad Máxima permitida.
- Cantidad utilizada en Producción.
- Costo Unitario.

- Indicativo de Fabricación.

Implícitamente deberá mostrar un presupuesto de una probable orden de Producción, así como mostrar el monto total del costo que implicará la generación del producto final.

Si el sistema detecta la existencia de un plan de producción habilitado para el día actual, deberá generar automáticamente la Orden de Producción correspondiente.

Detectar cuando un componente o insumo ha llegado a su cantidad mínima en el inventario.

Realizar la descarga automática de almacén para movimientos de materiales o refacciones utilizados por las órdenes de trabajo, una vez que esta es surtida.

Una vez que se cierre o se dé por concluida una Orden de Producción se deberá obtener información fehaciente de las labores realizadas, costos reales de los recursos utilizados incluidos los derivados por merma y desperdicio si así lo considera el responsable de la generación de la Orden de Producción. Además de ello se obtendrá el tiempo efectivo de ejecución y los tiempos muertos, así como los números de Lote y/o Serie. También se incluirán incidencias de Máquinas u Operarios, importantes para el análisis de efectividad de los sistemas productivos.

Al generar la Orden de Producción no importará si hay disponibilidad suficiente, sin embargo está no podrá ser surtida hasta que las cantidades requeridas sean totalmente cubiertas, en caso contrario el usuario deberá modificar la Orden Producción hasta cuadrar la cantidad a producir con la cantidad disponible.

Se deberá tener la opción de anexar documentos auxiliares tales como planos, especificaciones, procedimientos e instrucciones, etcétera, asegurando la actualización y consistencia de la información.

### III.2.3 HOJA DE ITINERARIO O DEFINICIÓN DE PROCESOS Y FASES DE PRODUCCIÓN

Es la parte de la función de planeación que se encarga de concentrar la información relativa a la secuencia de procesos y operaciones, así como las máquinas, herramientas y equipo auxiliar que se emplearán para la producción de una pieza, producto o lote dados. La información básica que debe incluirse es el nombre y clave del producto terminado o a fabricar, operaciones a ejecutar, tamaño del lote, departamento en los que se desarrolla el trabajo, tipo de máquina o estación de trabajo para cada operación, secuencia de operaciones, tiempo que requiera cada operación, entre otros.

### III.2.3.1 Requerimiento en Sistema

Se debe elaborar un programa para la generación de la hoja de itinerario que conllevará la ejecución de una Orden de Producción, de tal manera que permita analizar y apreciar el tiempo que se invertirá para su ejecución. Deberá considerarse para cada proceso ciertas fases (en caso requerido). Mediante estos datos, deberá ser posible monitorear paso a paso el proceso de fabricación o ensamble.

## III.2.4 PLANIFICACION DE PRODUCCIÓN

Para lograr una buena planeación es necesario considerar el factor que lleva implícito ésta misma. A este factor se le llama programación, y programar significa establecer un horario destinado a las actividades que requieren las instalaciones productivas. En este caso interesa elaborar horarios para órdenes de fabricación, utilizando los medios de producción disponibles en la fábrica.

### III.2.4.1 Requerimiento en Sistema

Se debe elaborar un control de los procesos que se lleven a cabo para la ejecución de una Orden de Producción considerando tiempos estimados y tiempos reales, así como citar las causantes que derivaron en demoras e inconvenientes para la óptima ejecución de una Orden de Producción, de tal manera que permita monitorear el avance de la producción. Deberá considerarse para cada proceso ciertas fases. Deberá permitir conocer el estado actual de Orden de Producción, de tal manera que sea posible detectar retrasos en la fecha de finalización o cierre.

Para ello se deberá considerar una terminal por cada proceso y si es preciso por cada etapa, que recopile los datos que se susciten en tiempo real.

La planeación de Órdenes de Producción permite definir las tareas y los recursos que se requieren para obtener el producto final. Una orden de Producción puede o no estar dividida en tareas, las cuales son unidades de trabajo que deben poder ser programadas y realizadas independientemente.

## III.2.5 PLAN DIARIO Y MENSUAL DE PRODUCCIÓN

Para lograr una buena planeación de la cantidad de productos terminados que se desea obtener de todas y cada una de las líneas de producción deberá programarse previamente la cantidad a producir, preferentemente con un mes de anticipación o más tardar el primer día del mes.

### III.2.5.1 Requerimiento en Sistema

Se debe elaborar un programa que permita generar un Plan Diario de un Mes de Producción de tal manera que se disponga de datos que sirvan para generar en automático las Órdenes de Producción. Las estimaciones necesarias para determinar dichas cifras deberán ser bajo criterio del responsable del departamento de Producción.

Este plan deberá contar con la clave del artículo a producir así como su cantidad correspondiente. Más ambicioso resultará si aunado a este plan, se muestra de manera paralela la cantidad producida por día; información que permitirá analizar rápidamente, si el plan de producción está cumpliendo con su propósito.

## III.2.6 CATÁLOGOS

Tanto la Explosión de Materiales como la Generación de Ordenes de Producción conllevan la previa generación de Catálogos o Maestros que permitan administrar la información básica y comúnmente solicitada. Los catálogos identificados para este propósito son:

- Unidades de Medida.
- Turnos.
- Líneas de Ensamble.
- Motivos de Tiempos Muertos.
- Tipos de Producción.
- Estatus de la Orden de Producción.
- Fases de Producción.
- Procesos de Producción.
- Artículo versus Proceso.

### III.2.6.1 Requerimiento en Sistema

Se debe elaborar un conjunto de Catálogos o Maestros, los cuales además de asignar un folio o identificador a la descripción registrada, deberá almacenar el Usuario y la Fecha de Alta, así como el Usuario y Fecha de Baja, en caso de que el registro sea Deshabilitado.

## III.2.7 ANÁLISIS DE PRODUCTIVIDAD

Productividad es la cualidad o condición de ser productivos. Es un concepto que guía la administración de un sistema de producción y mide su éxito. Es la cualidad que indica qué tan bien se están utilizando la mano de obra, el capital, los materiales y la energía. Es la relación entre producción e insumos. La producción creciente puede o no mejorar la productividad, dependiendo de los insumos utilizados para lograr ese aumento. La productividad indica la eficiencia de las operaciones y sugiere, por lo tanto, su rentabilidad; pero las operaciones ineficientes pueden en ocasiones ser rentables si el producto disfruta de una acogida favorable en el mercado.

La productividad se puede medir por la siguiente relación básica:

$$\text{Pr}oductividad = \frac{\text{Pr}oducción}{Insumos}$$

Como medida de la eficiencia de producción, la relación toma comúnmente la forma de producción por horas de trabajo, siendo el importe o las unidades de producción la dimensión del numerador; pero la relación se puede adaptar para calificar la mayoría de las funciones de producción.

Se produce un incremento de la productividad cuando la relación producción a insumos aumenta de un período al siguiente. Un aumento de la producción no da lugar necesariamente a un aumento de la productividad. Por ejemplo, la productividad sube cuando la relación de producción / insumos pasa de

$$\frac{120\ (\text{Pr}oducción)}{100\ (Insumos)} \text{ a } \frac{150\ (\text{Pr}oducción)}{100\ (Insumos)}$$

Si la segunda relación hubiera sido 150 / 125, la productividad habría permanecido igual al registrarse un aumento de la producción. De manera que la meta de aumentar la productividad se alcanza mediante un excedente proporcional siempre mayor de los valores de producción sobre los insumos.

En vista de que las organizaciones usan las medidas de productividad para diversos fines, no hay una formula estándar para calcular un índice de productividad. Se han propuesto innumerables variantes para realizar el propósito de la relación básica producción / insumos.

Para ello es recomendable utilizar un Índice de Productividad Total el cual es una cifra única que expresa la eficiencia de toda una organización. Su formulación incluye una exposición general del valor de producto o servicio y un valor resumido de todos sus insumos. Típicamente, se usan dimensiones monetarias para el numerador y el denominador, a fin de permitir que los diversos productos y recursos sean expresados en términos equivalentes. La conocida relación producción/horas de trabajo sólo se aproxima a un índice de productividad total cuando la

organización depende mucho de la mano de obra. Una medición de la productividad total se obtiene a partir de

$$IPT = \frac{\Pr oducto + Servicio}{MO + Materiales + Energía + Capital}$$

donde:

IPT = Indice de Productividad Total.

MO = Mano de Obra.

A partir de esta expresión básica se pueden hacer adaptaciones para representar con más exactitud las funciones de una organización en particular. Al particularizar el índice, la intención es reflejar los objetivos de la entidad. Por consiguiente, se han elaborado muchas versiones. Por ejemplo una empresa puede opinar que las compras de materias primas representan un elemento de productividad que no corresponde a su tipo de negocio y, por lo tanto deben ser excluidas de sus insumos. Otras empresas que utilizan grandes cantidades de materiales dirían que la exclusión no está justificada. Otra más puede tener un consumo pequeño y constante de energía y dirá que el insumo energía se puede pasar por alto en el modelo. Un productor de aluminio, por supuesto, pensará de otro modo. Uno de los muchos modelos posibles de productividad total tiene la fórmula siguiente:

### III.2.7.1 Requerimiento en Sistema

Se debe elaborar un programa para generar el análisis de productividad, el cual permita obtener estadísticas de Rendimiento y efectividad, mermas por línea de producción. Deberá permitir analizar costos y horas de trabajo estimados contra reales.

# III.3 REQUERIMIENTOS GENERALES

## III.3.1 PERMISOS RESTRINGIDOS

Restricciones de acceso, de tal manera que usuarios existan diferentes niveles de seguridad.

## III.3.2 REPORTES

- Reporte de proyecciones de consumo sobre la base del plan de producción.

- Reportes sobre costos incurridos y por incurrir según Ordenes de Producción ya ejecutadas o planificadas.

- Generación de gráficas de diversa índole.

- Formato General de Orden Producción.

- Reportes Tabulares de la información existente en los Catálogos generados.

## III.3.3 CONSIDERACIONES ESPECÍFICAS Y ESTÁNDARES

La Generación o Cancelación de cada registro, entiéndase por ello Alta o Baja de información en Catálogos, Explosión de Materiales y Generación de Orden de Producción implicará el almacenamiento del nombre del usuario así como la Fecha en que éste lleva a cabo tal operación. Asimismo los Catálogos deberán conservar un estándar entre sí, pues al tratarse de Operaciones comunes de Alta, Baja y Cambio, resulta lógico considerar la misma forma de funcionamiento entre ellos, de tal manera que resulte más fácil al usuario el manipular la información contenida en estos programas.

# III.4 MODELO DE TRES CAPAS

Las tres capas definidas para el desarrollo del Sistema de Información para el Control de Procesos de Producción son las que se muestran en la figura III.5.

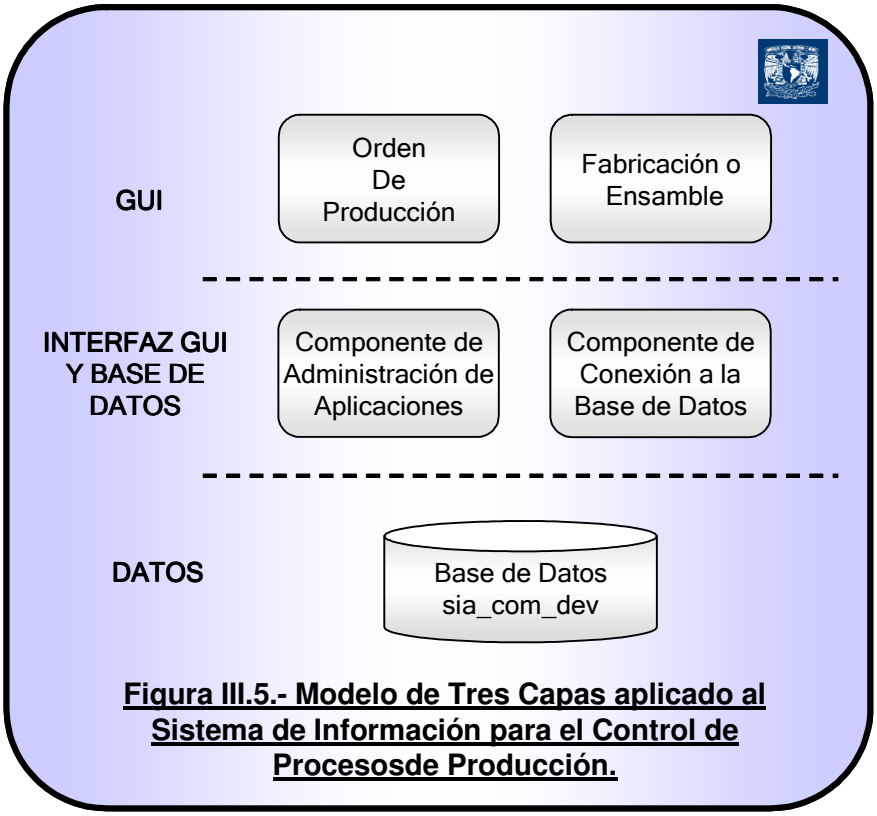



**Figura III.5.- Modelo de Tres Capas aplicado al Sistema de Información para el Control de Procesosde Producción.**

# III.5 CODIFICACIÓN UML

## III.5.1 CONCEPTOS DE LA EMPRESA

A grandes rasgos las entidades o principales conceptos que intervienen en el proceso de Producción son los que se muestran en la figura III.6.

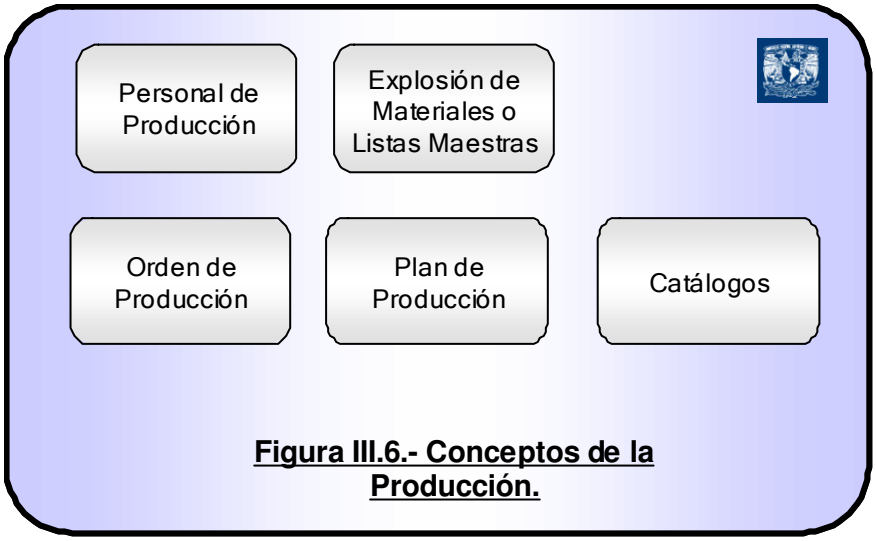



**Figura III.6.- Conceptos de la Producción.**

## III.5.2 CASOS DE USO

### III.5.2.1 Explosión de Materiales

El Ing. Industrial o el personal responsable administra las listas maestras, de tal manera que determina cuáles y que cantidad de componentes integran un producto terminado o subcomponente, cambia componentes y cantidades, o bien los elimina, es decir tiene al día las listas de materiales. Esto se demuestra con el correspondiente caso de Uso de la figura III.7.

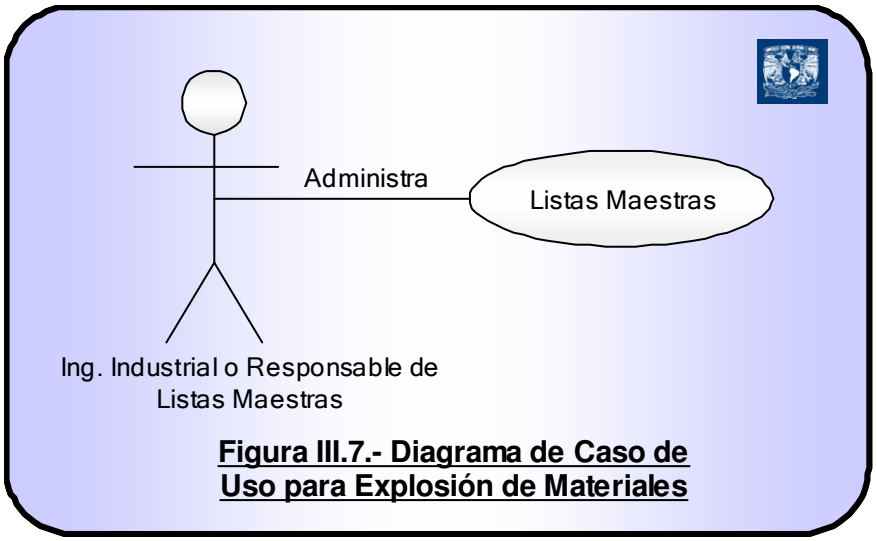



**Figura III.7.- Diagrama de Caso de Uso para Explosión de Materiales**

### III.5.2.2 Orden de Producción

El área de Logística y Producción es la responsable de determinar lo que ha producirse día con día, de ahí que sea la encargada de generar las Ordenes de Producción que corresponda con las cantidades requeridas por el área de ventas o planeación de ventas, por lo establecido en el Plan de Producción o en caso extremo, por las requisiciones urgentes o inmediatas que el almacén de producto terminado determine. Esto se demuestra con el correspondiente caso de Uso de la figura III.8.

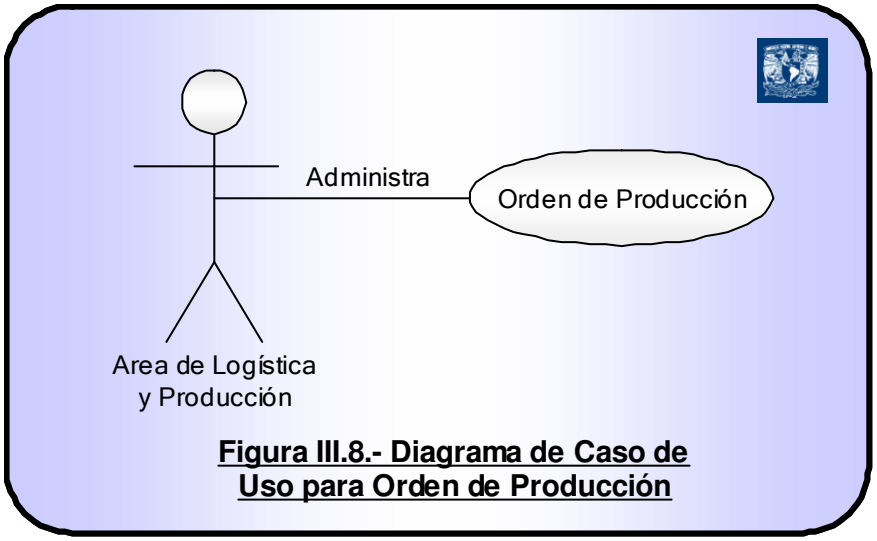



**Figura III.8.- Diagrama de Caso de Uso para Orden de Producción**

### III.5.2.3 Plan de Producción

El área de Logística y Planeación de la Producción genera y administra el Plan Mensual de Producción por lo que es aquí donde se establece la cantidad diaria a producir. La estructuración de este plan se conformará de las cantidades solicitadas por el almacén de Producto Terminado, o por un Plan de Ventas, definido por el área correspondiente. Esto se demuestra con el correspondiente caso de Uso de la figura III.9.

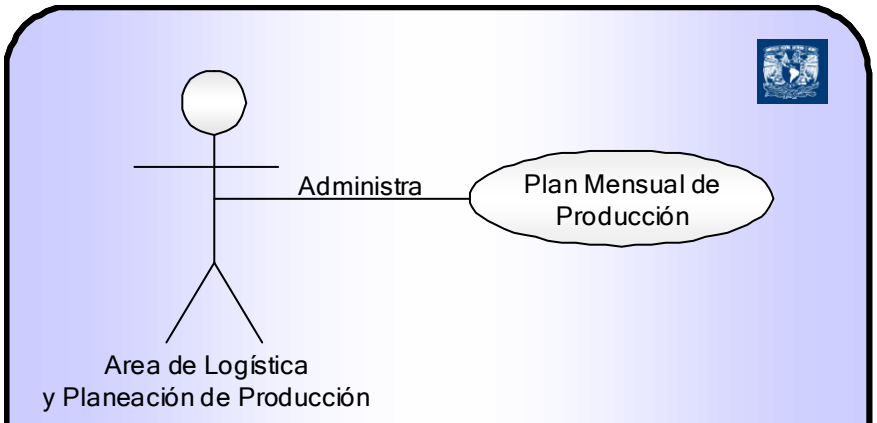



**Figura III.9.- Diagrama de Caso de Uso para el Plan de Producción**

### III.5.2.4 Bitácora de Avance de Orden de Producción

El personal asignado a la línea de Producción será el responsable de ir registrando las cantidades que él mismo va completando, por lo que el supervisor en Turno deberá monitorear el Avance que se va suscitando en la Orden de Producción durante esta etapa y registrarlo en su bitácora. Esto se demuestra con el correspondiente caso de Uso de la figura III.10.

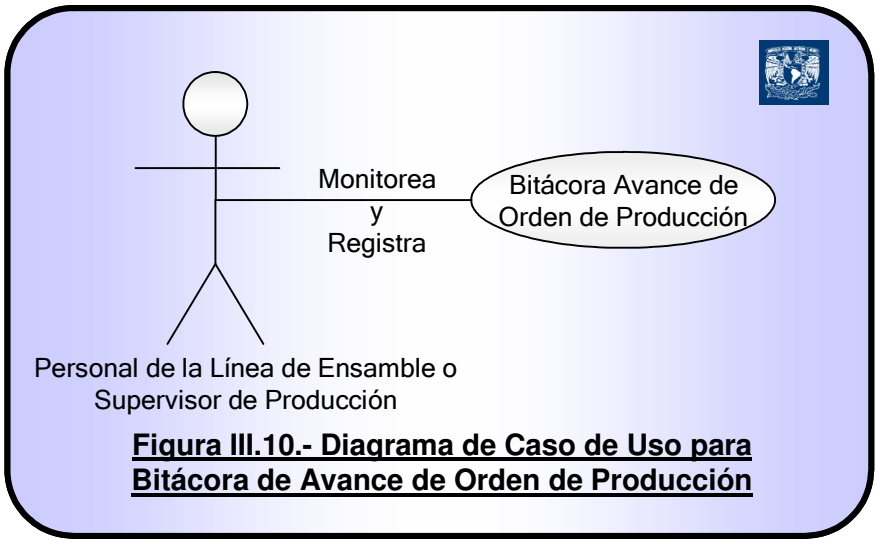



**Figura III.10.- Diagrama de Caso de Uso para Bitácora de Avance de Orden de Producción**

## III.5.3 DIAGRAMAS DE COLABORACIÓN

### III.5.3.1 Explosión de Materiales

El Ing. Industrial o el personal responsable se ha definido de forma genérica como Usuario Final. Esto se demuestra con el correspondiente Diagrama de Colaboración de la figura III.11.

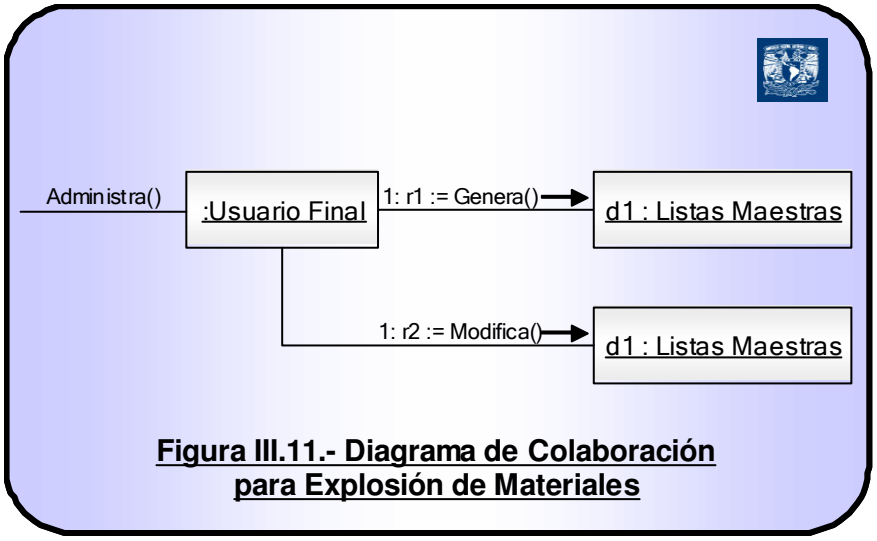



**Figura III.11.- Diagrama de Colaboración para Explosión de Materiales**

### III.5.3.2 Orden de Producción

El área de Logística y Producción se ha definido de forma genérica como Usuario Final. Esto se demuestra con el correspondiente Diagrama de Colaboración de la figura III.12.

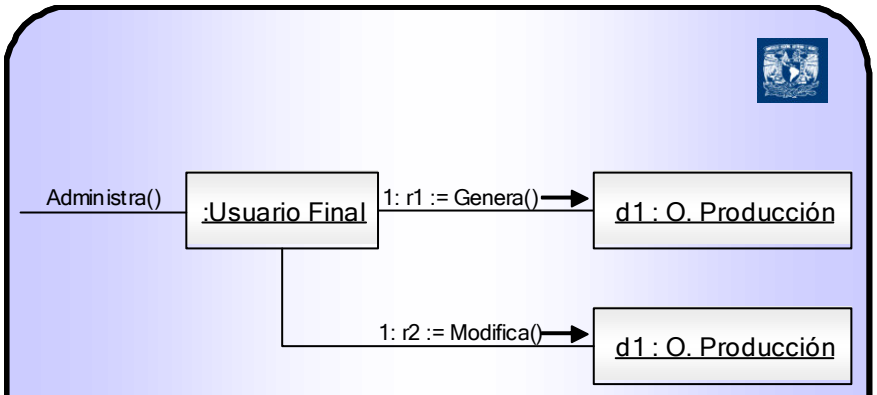



**Figura III.12.- Diagrama de Colaboración para Orden de Producción**

### III.5.3.3 Plan de Producción

El área de Logística y Planeación de Producción se ha definido de forma genérica como Usuario Final. Esto se demuestra con el correspondiente Diagrama de Colaboración de la figura III.13.

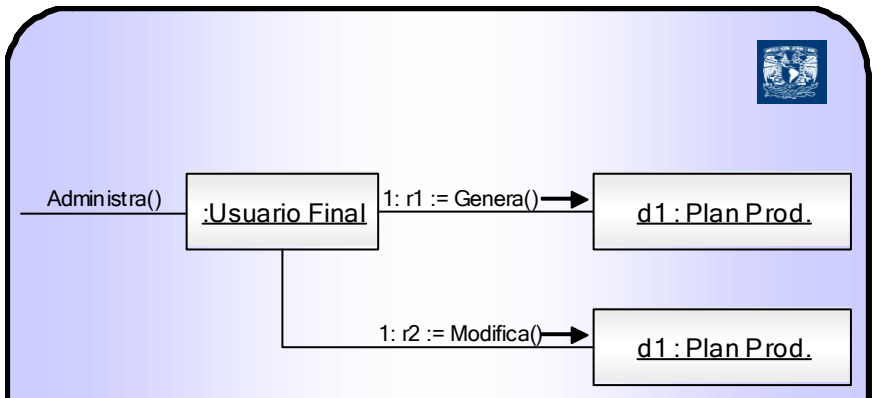



**Figura III.13.- Diagrama de Colaboración para Plan Mensual de Producción**

### III.5.3.4 Bitácora de Avance de Orden de Producción

El área de Logística y Planeación de Producción se ha definido de forma genérica como Usuario Final. Esto se demuestra con el correspondiente Diagrama de Colaboración de la figura III.14.

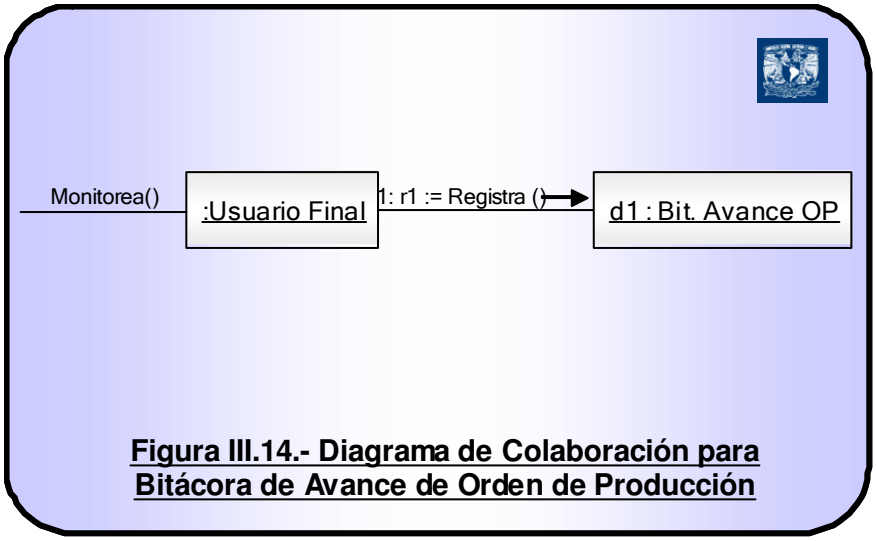



**Figura III.14.- Diagrama de Colaboración para Bitácora de Avance de Orden de Producción**

## III.5.4 DIAGRAMAS DE SECUENCIA

### III.5.4.1 Explosión de Materiales

En el Diagrama de Colaboración de la figura III.15, se muestra de manera genérica la Secuencia Fundamental de la forma en que el usuario final estructura la Explosión de Materiales.

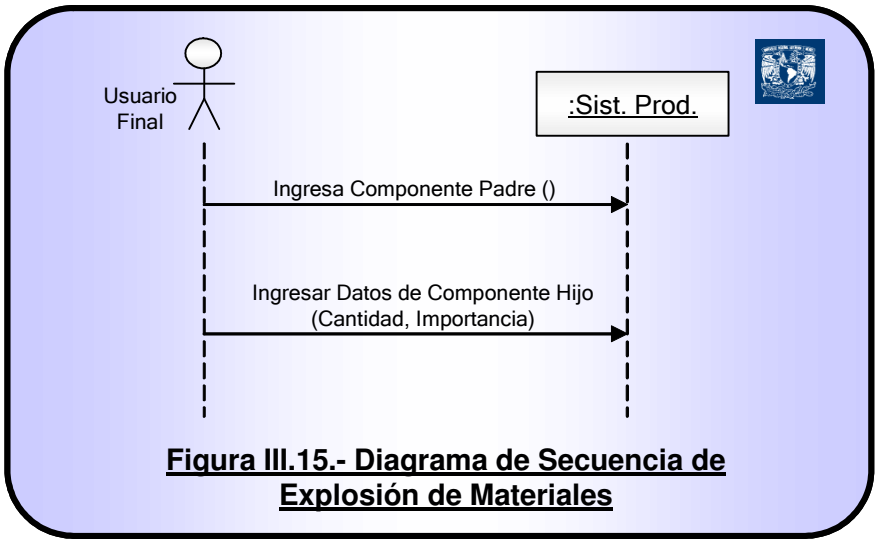



**Figura III.15.- Diagrama de Secuencia de Explosión de Materiales**

### III.5.4.2 Orden de Producción

En el Diagrama de Colaboración de la figura III.16, se muestra la secuencia para generar una Nueva Orden de Producción.

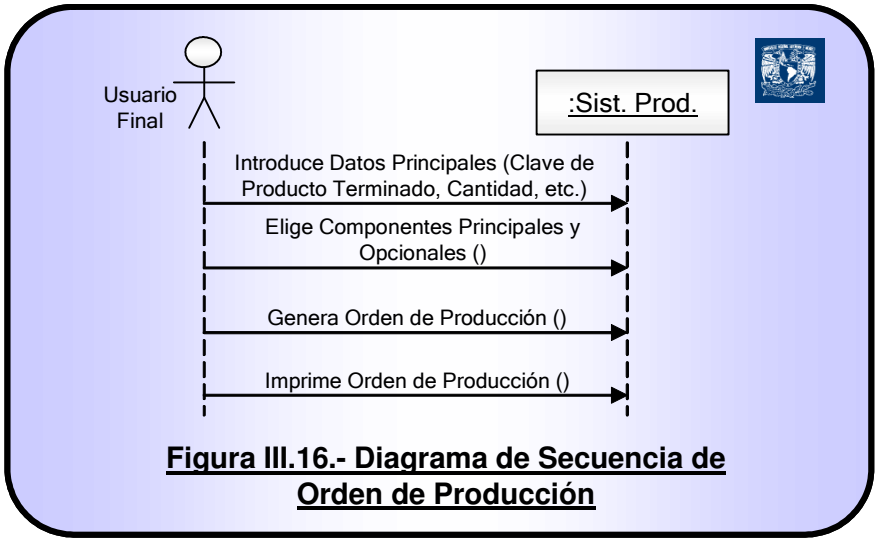



**Figura III.16.- Diagrama de Secuencia de Orden de Producción**

### III.5.4.3 Plan Mensual de Producción

El Diagrama de Colaboración de la figura III.17, muestra la sucesión de pasos que el usuario final lleva a cabo para definir el Plan Mensual de Producción, día a día, para un Producto Terminado cualquiera.

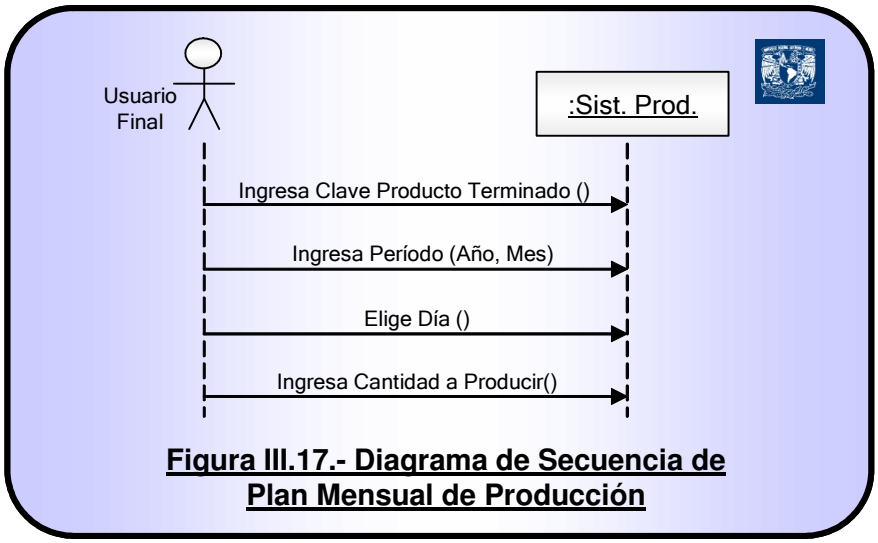



**Figura III.17.- Diagrama de Secuencia de Plan Mensual de Producción**

### III.5.4.4 Bitácora de Avance de Orden de Producción

Este Diagrama de Colaboración (figura III.18), define la serie de pasos que el usuario final lleva a cabo para registrar la bitácora del Avance de la Orden de Producción, cuando está última se encuentra en el estatus "En Producción", la cantidad de Registros en esta bitácora dependerá del Proceso y Fases asociadas a la Orden de Producción, las cuales se definieron durante su generación.

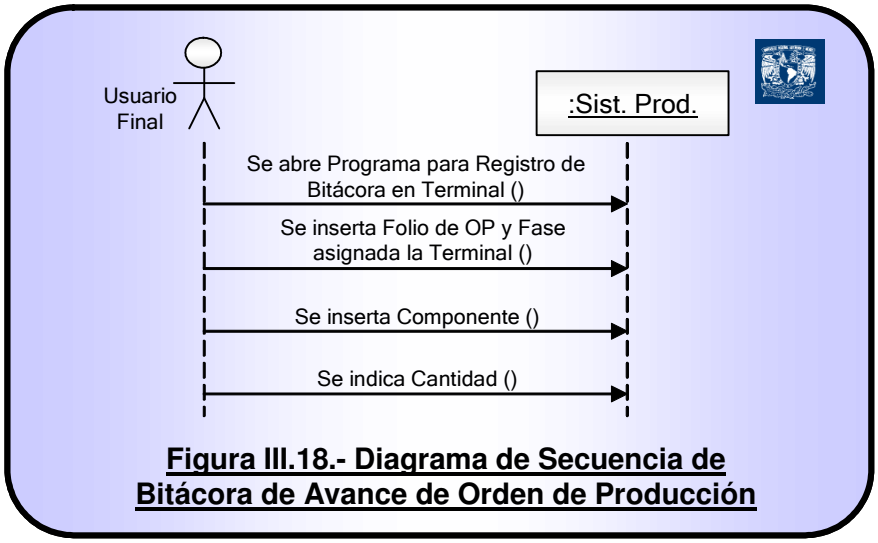



**Figura III.18.- Diagrama de Secuencia de Bitácora de Avance de Orden de Producción**

## III.5.5 DIAGRAMAS DE CLASE

### III.5.5.1 Clase Principal

A continuación se muestra la Super Clase de los Principales Objetos del Sistema, esta contiene sus características y métodos generales, tal y como se puede ver en la figura III.19.

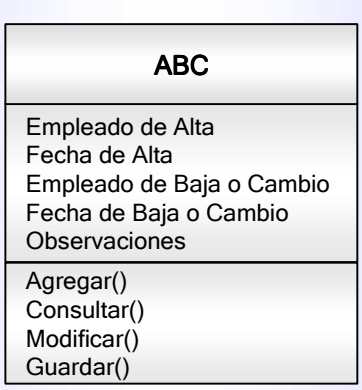
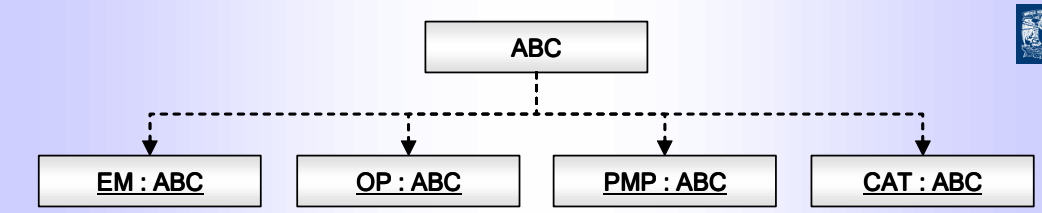



**Figura III.19.- Clase de Altas, Bajas y Cambios**

### III.5.5.2 Clase con Herencia para Objetos Principales del Sistema

Los Objetos de las principales operaciones del sistema se mencionan en la figura III.20.




Donde:
EM = Explosión de Materiales.
OP = Orden de Producción.
PMP = Plan Mensual de Producción.
CAT = Catálogo.

**Figura III.20.- Diagrama de Clases con Herencia para los Los Principales Objetos del Sistema**

### III.5.5.3 Clase con Herencia para Catálogos del Sistema

El objeto CAT (Catálogo) es a la vez la Clase Padre de todos los Catálogos del Sistema. Esto se muestra en la figura III.21.

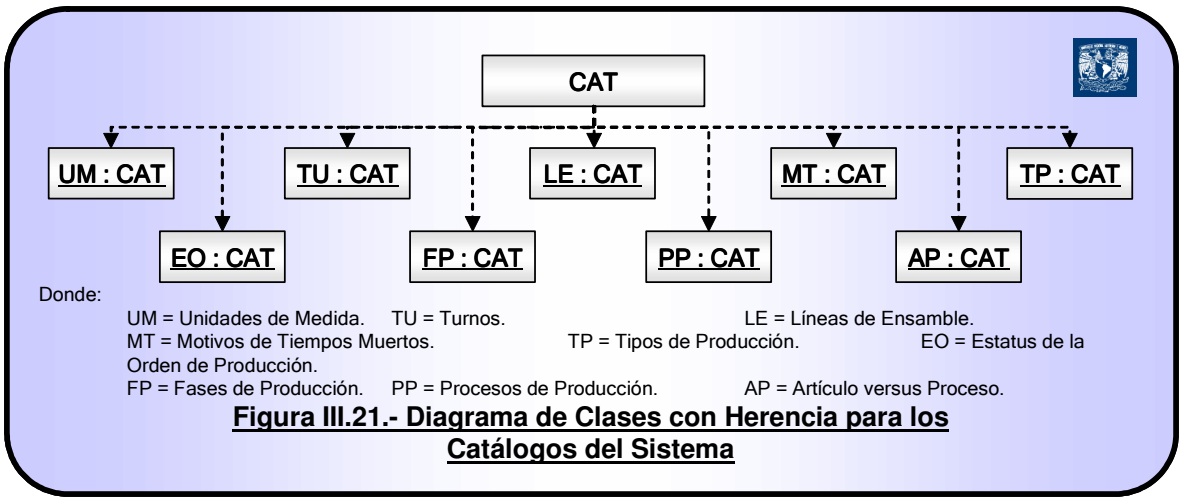



**Figura III.21.- Diagrama de Clases con Herencia para los Catálogos del Sistema**

## III.5.6 DIAGRAMA DE FLUJO DE UNA ORDEN DE PRODUCCIÓN

En la figura III.22, se muestra el camino que recorre una Orden de Producción desde su Generación hasta su Terminación.

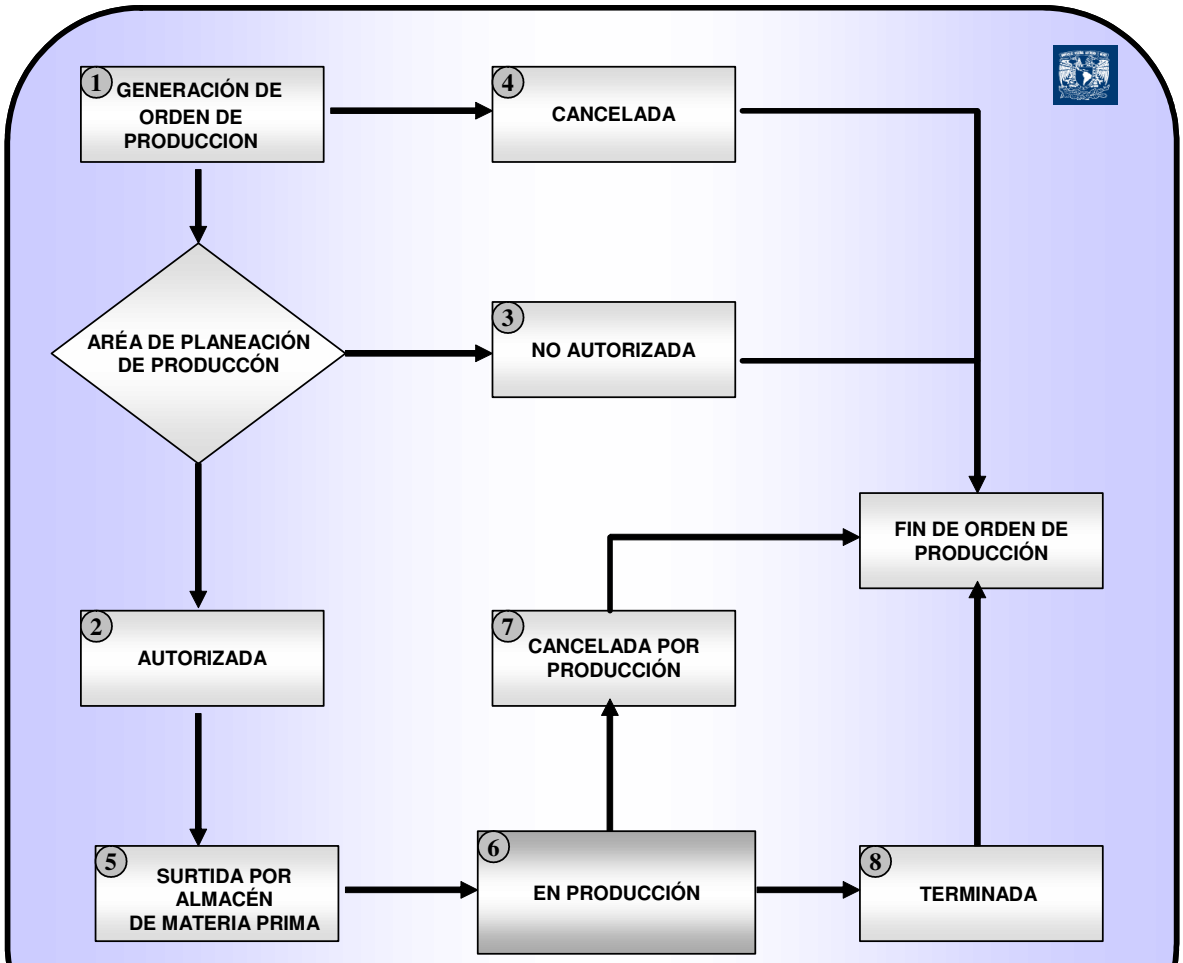



**Figura III.22.- Flujo de una Orden de Producción**

## III.5.7 DIAGRAMA DE ACTIVIDADES DE LOS ESTATUS DE LA ORDEN DE PRODUCCIÓN

Los Estatus que comprende la secuencia de una Orden de Producción se visualiza en la figura III.23.

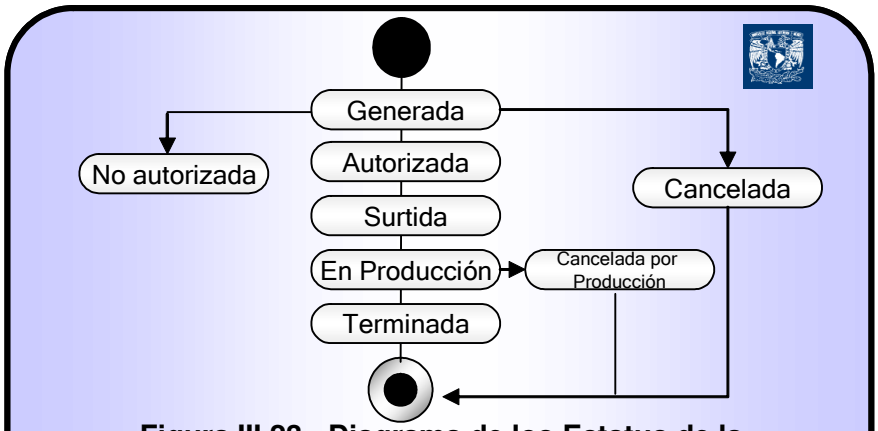



**Figura III.23.- Diagrama de los Estatus de la Orden de Producción**

## III.5.8 DIAGRAMA DE ACTIVIDADES DE LAS FASES DEL PROCESO DE PRODUCCCIÓN CUANDO EL ESTATUS DE LA ORDEN DE PRODUCCIÓN ES "EN PRODUCCIÓN".

La parte fundamental de este sistema se muestra en la figura III.24, donde se percibe la sucesión de fases, para su correcto monitoreo.

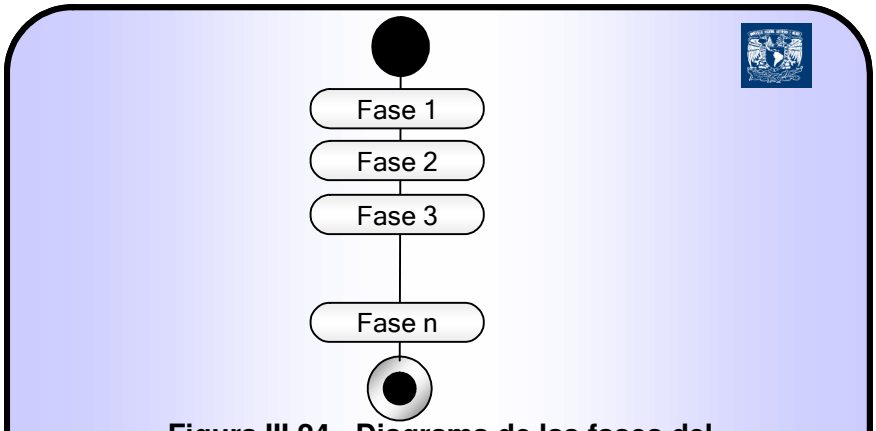



**Figura III.24.- Diagrama de las fases del Proceso de Producción cuando la OP está en el estatus "En Producción".**

# III.6 PROTOTIPOS DE LA IINTERFAZ DEL USUARIO

## III.6.1 OBJETOS PRINCIPALES DEL SISTEMA

### III.6.1.1 Acceso

El propósito de este objeto será permitir el acceso al Sistema, siempre y cuando el usuario final proporcione una clave de usuario y su respectiva contraseña. Este par de datos deberá cubrir una serie de requisitos y validaciones internas que se llevan a cabo, entre ellas que ser un empleado "activo", disponer de permisos suficientes para operar con este Sistema, entre otros. El prototipo correspondiente se muestra en la figura III.25

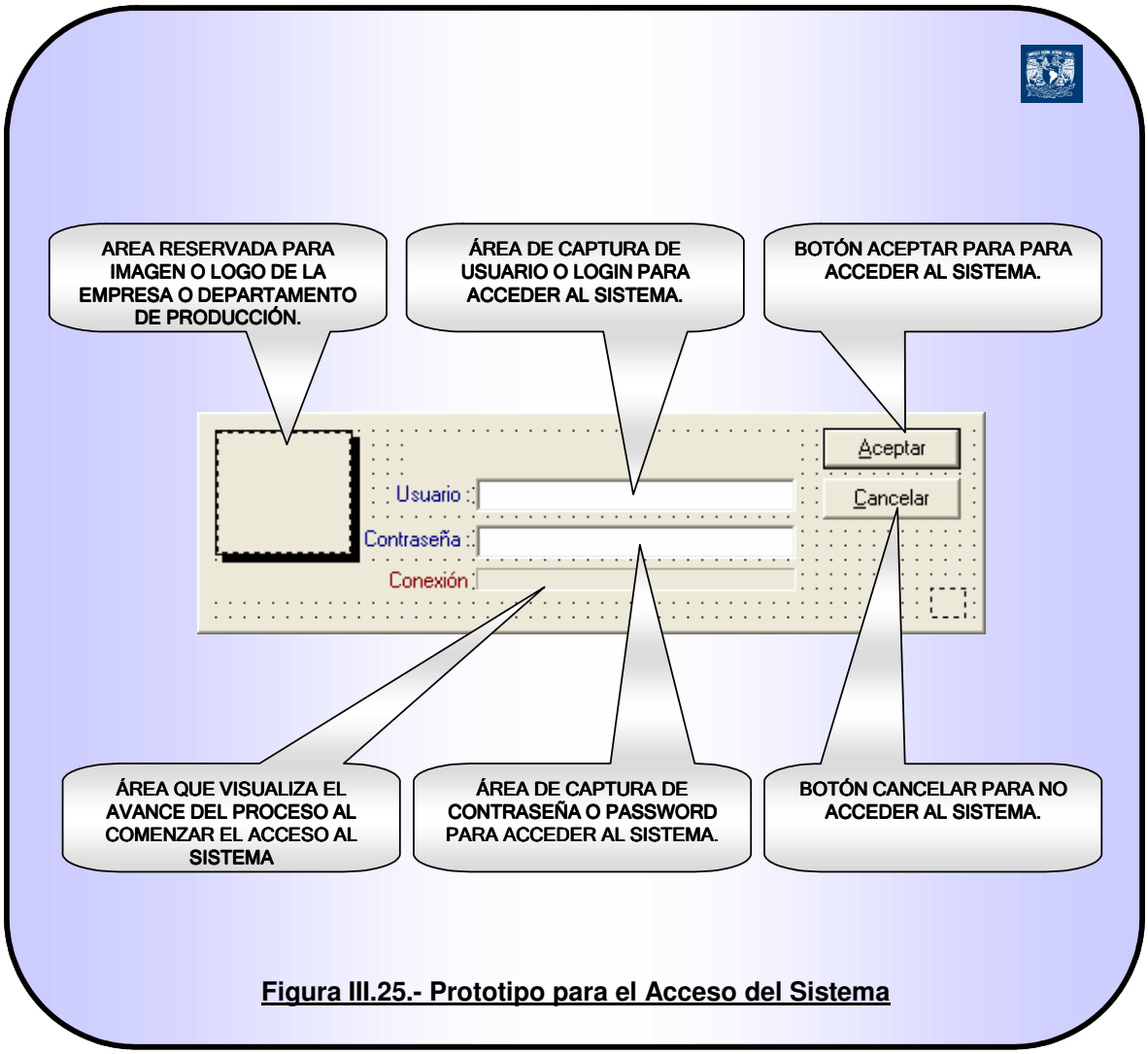



**Figura III.25.- Prototipo para el Acceso del Sistema**

### III.6.1.2 Búsqueda

Este es el objeto estándar para la búsqueda de cualquier tipo de dato en el sistema, pues permitirá definir criterios muy diversos. El prototipo correspondiente se muestra en la figura III.26

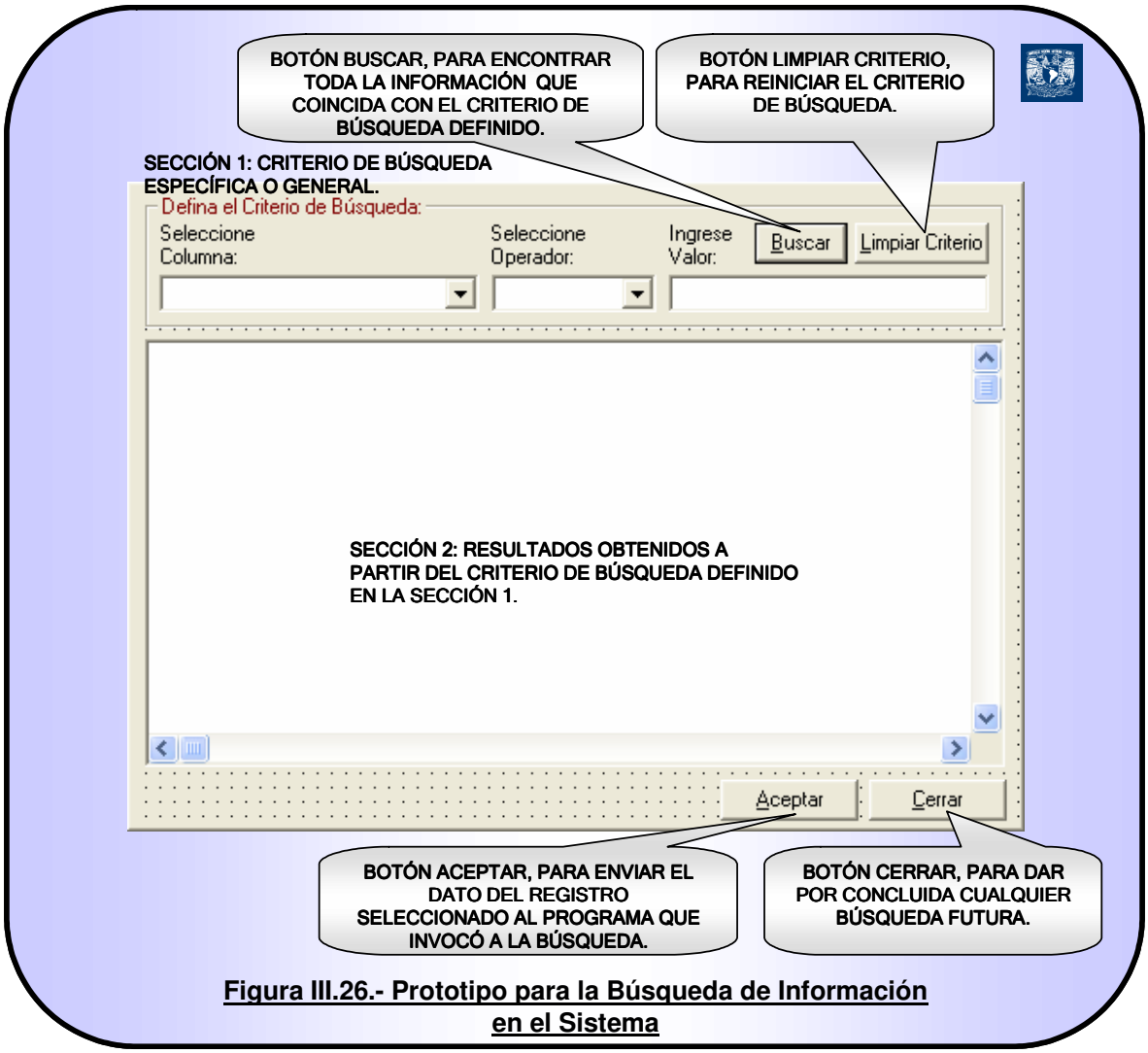



**Figura III.26.- Prototipo para la Búsqueda de Información en el Sistema**

### III.6.1.3 Explosión de Materiales

Su objetivo será la asignación de componentes y subcomponentes de un producto final, así como definir componentes o insumos opcionales. El prototipo correspondiente se muestra en la figura III.27.

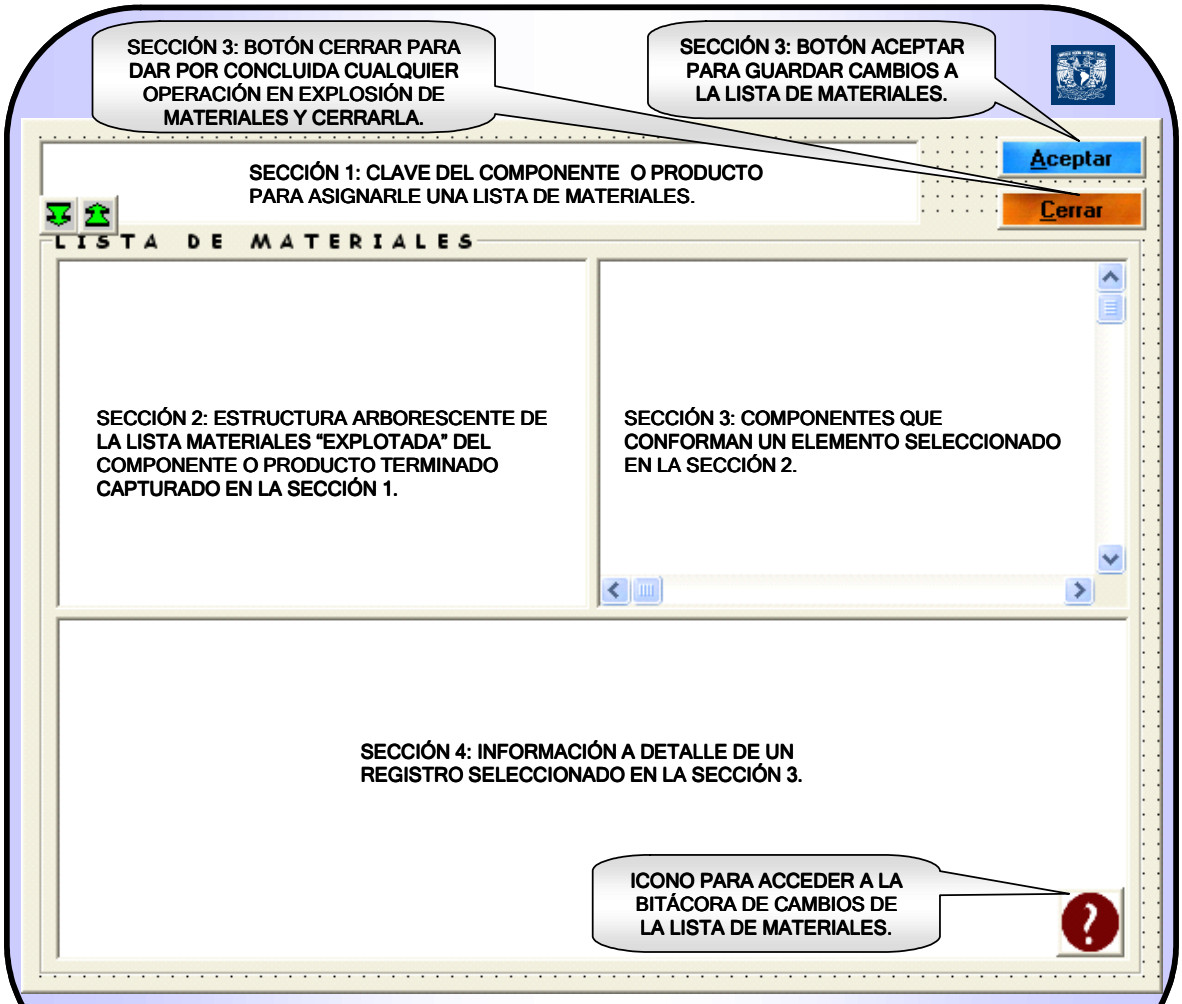



**Figura III.27.- Prototipo para la Explosión de Materiales**

### III.6.1.4 Orden de Producción

Objeto para la Generación, Consulta y Modificación de Ordenes de Producción de acuerdo a lo definido en la Explosión de Materiales. El prototipo correspondiente se muestra en la figura III.28.Esta opción deberá permitir asociar un Proceso y Fases de Producción a la Orden de Producción generada, para que, una vez que se encuentre la línea de ensamble o fabricación sea monitoreada.

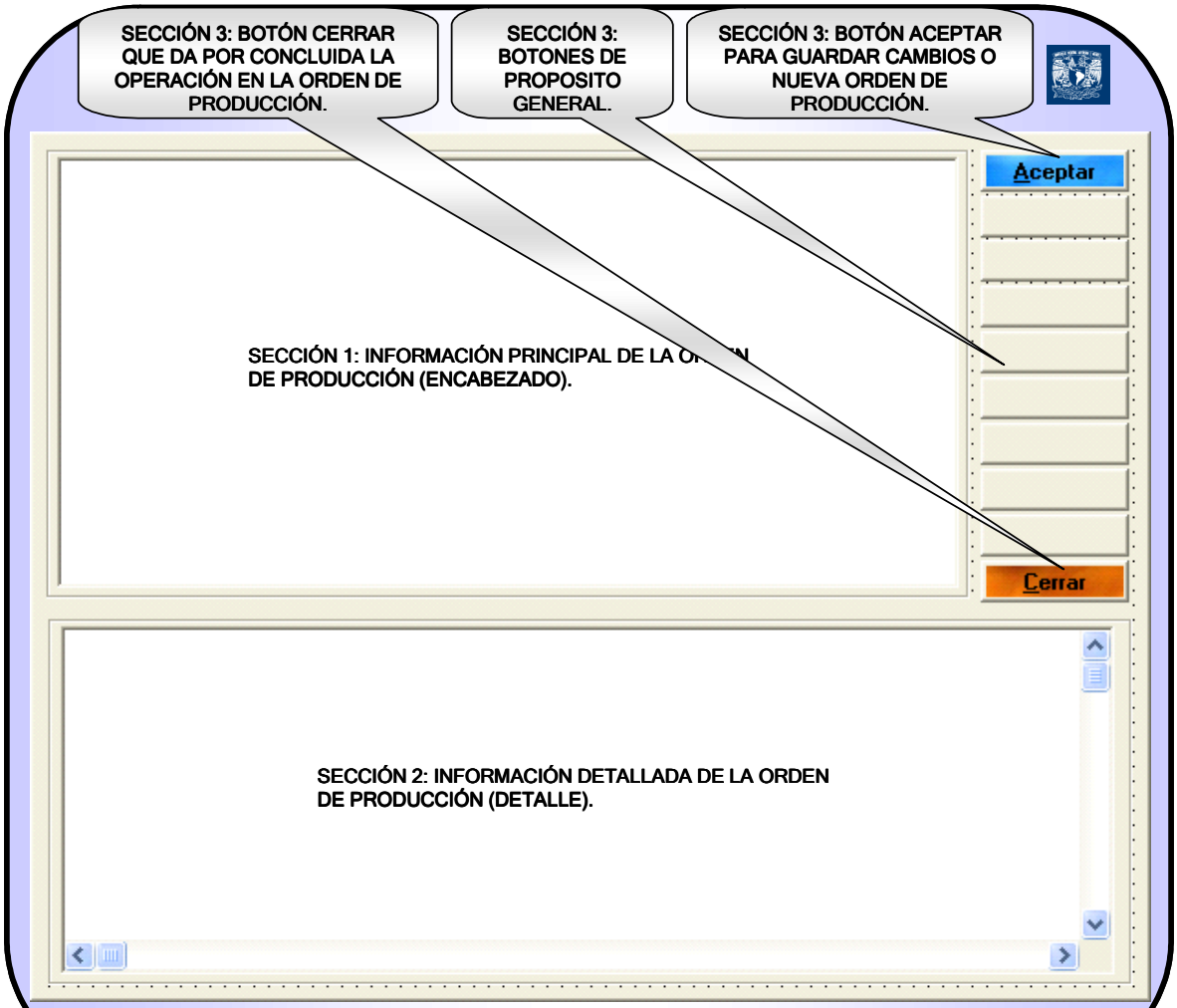



**Figura III.28.- Prototipo para la Orden de Producción**

### III.6.1.5 Plan Mensual de Producción

Programa para estructurar dicho Plan, a partir del día y mes actual, si así se desea. El prototipo correspondiente se muestra en la figura III.29

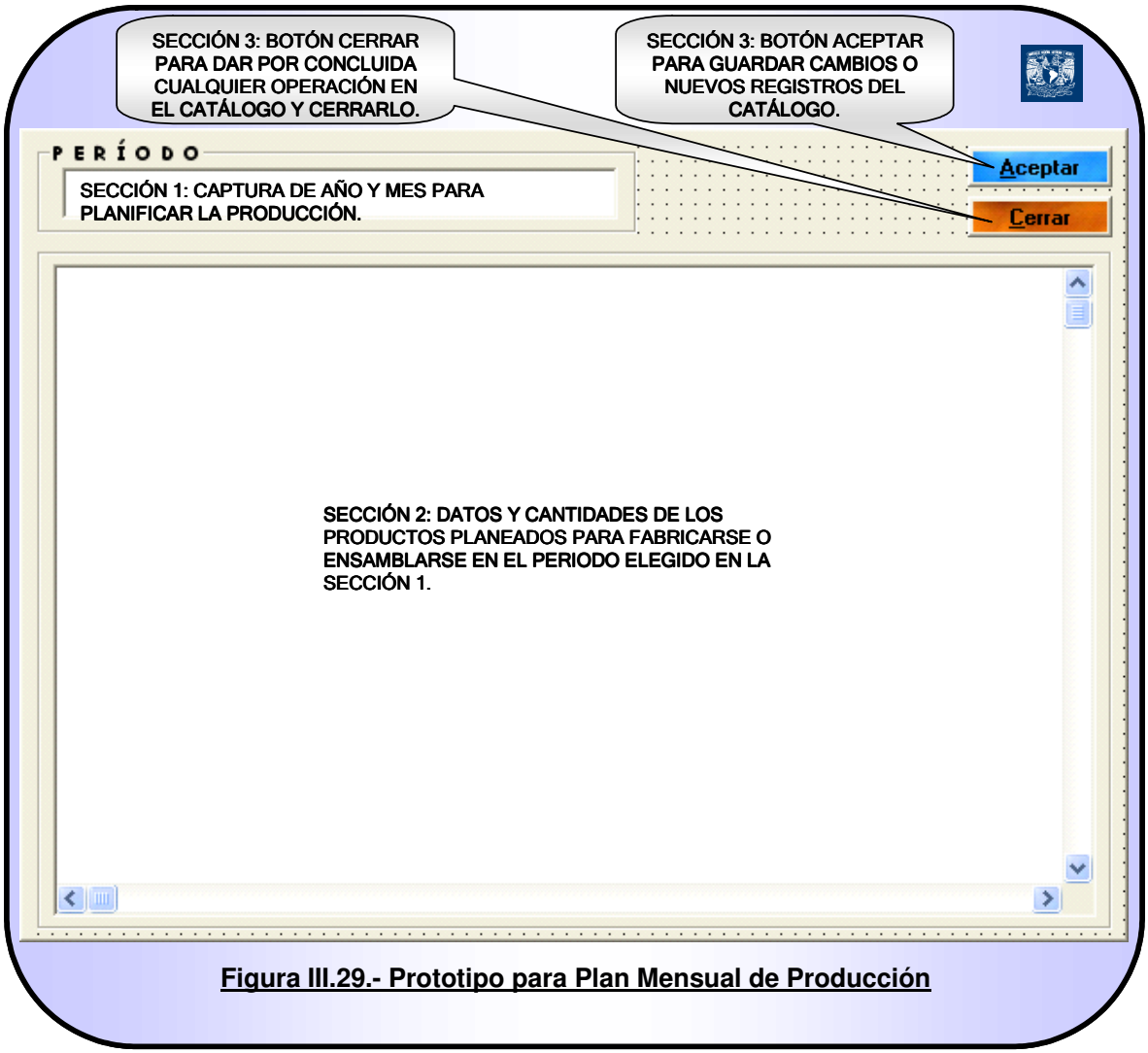



**Figura III.29.- Prototipo para Plan Mensual de Producción**

### III.6.1.6 Registro de Avance de Orden de Producción

Pequeña Aplicación auxiliar, cuyo objetivo es que el usuario final registre la clave del artículo y cantidad del mismo en la Fase que corresponda a la Terminal de Monitoreo actual, para una Orden de Producción cuando su estatus sea “En Producción”. El prototipo correspondiente se muestra en la figura III.30.

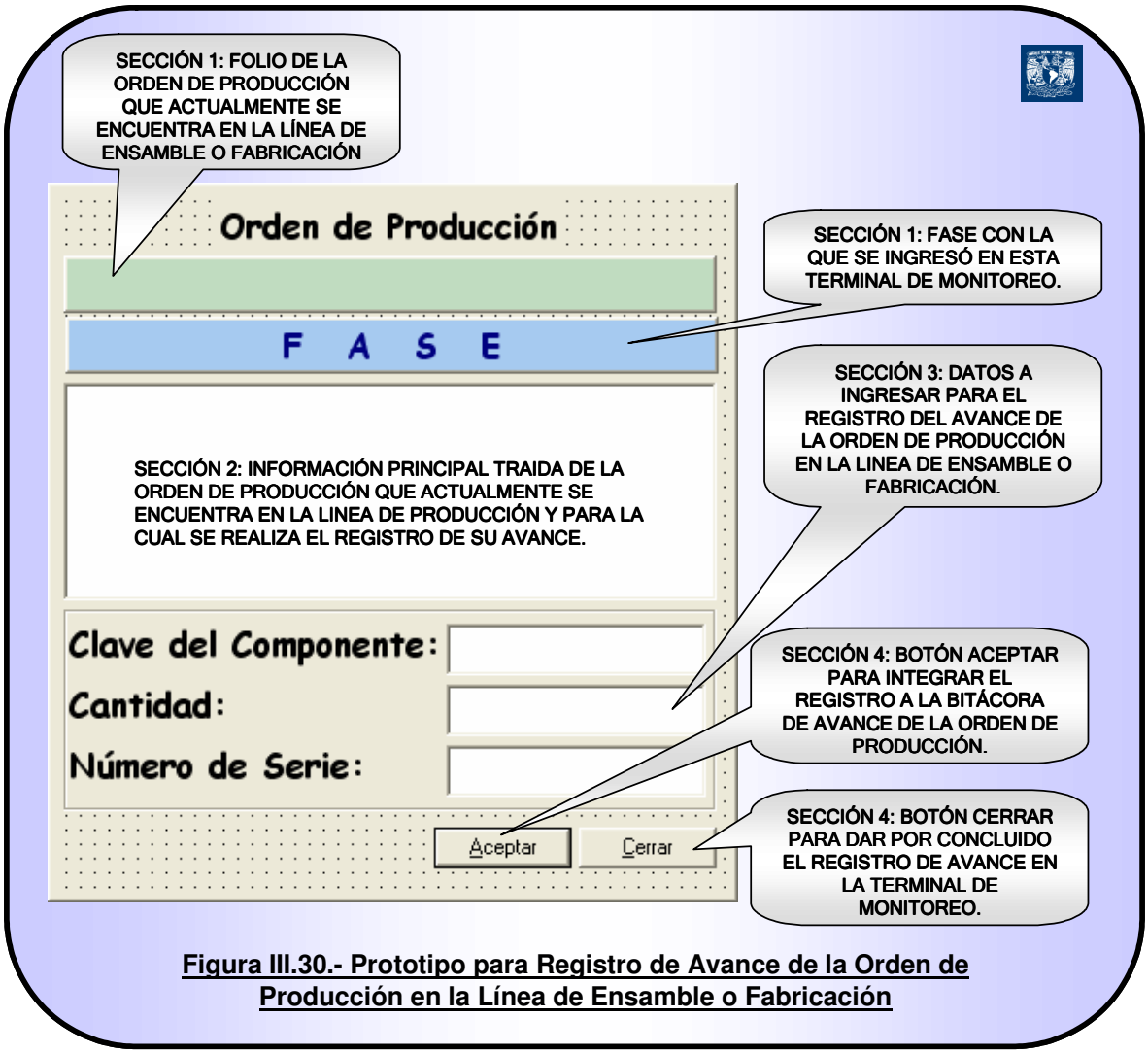



**Figura III.30.- Prototipo para Registro de Avance de la Orden de Producción en la Línea de Ensamble o Fabricación**

## III.6.2 CATÁLOGOS DEL SISTEMA

Plantilla estándar para cualquier catálogo que integre el sistema. El prototipo correspondiente se muestra en la figura III.31.

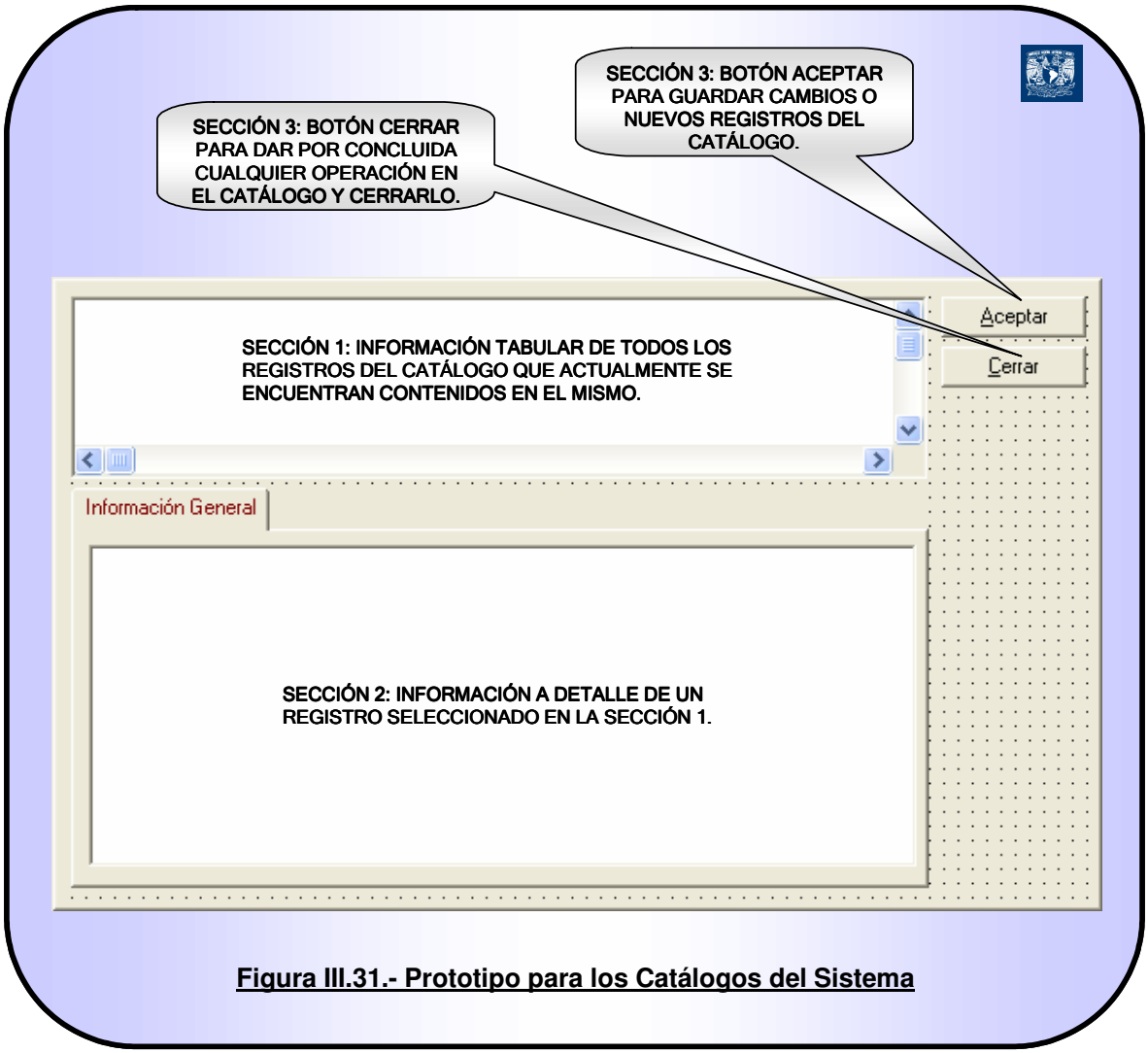



**Figura III.31.- Prototipo para los Catálogos del Sistema**

## III.6.3 REPORTES DE INFORMACIÓN

### III.6.3.1 Parámetros

En la figura III.32 se visualiza el prototipo de los objetos cuya tarea será el permitir que el usuario final ingrese la mayor parte de datos, que permitan filtrar un Reporte a Generar.

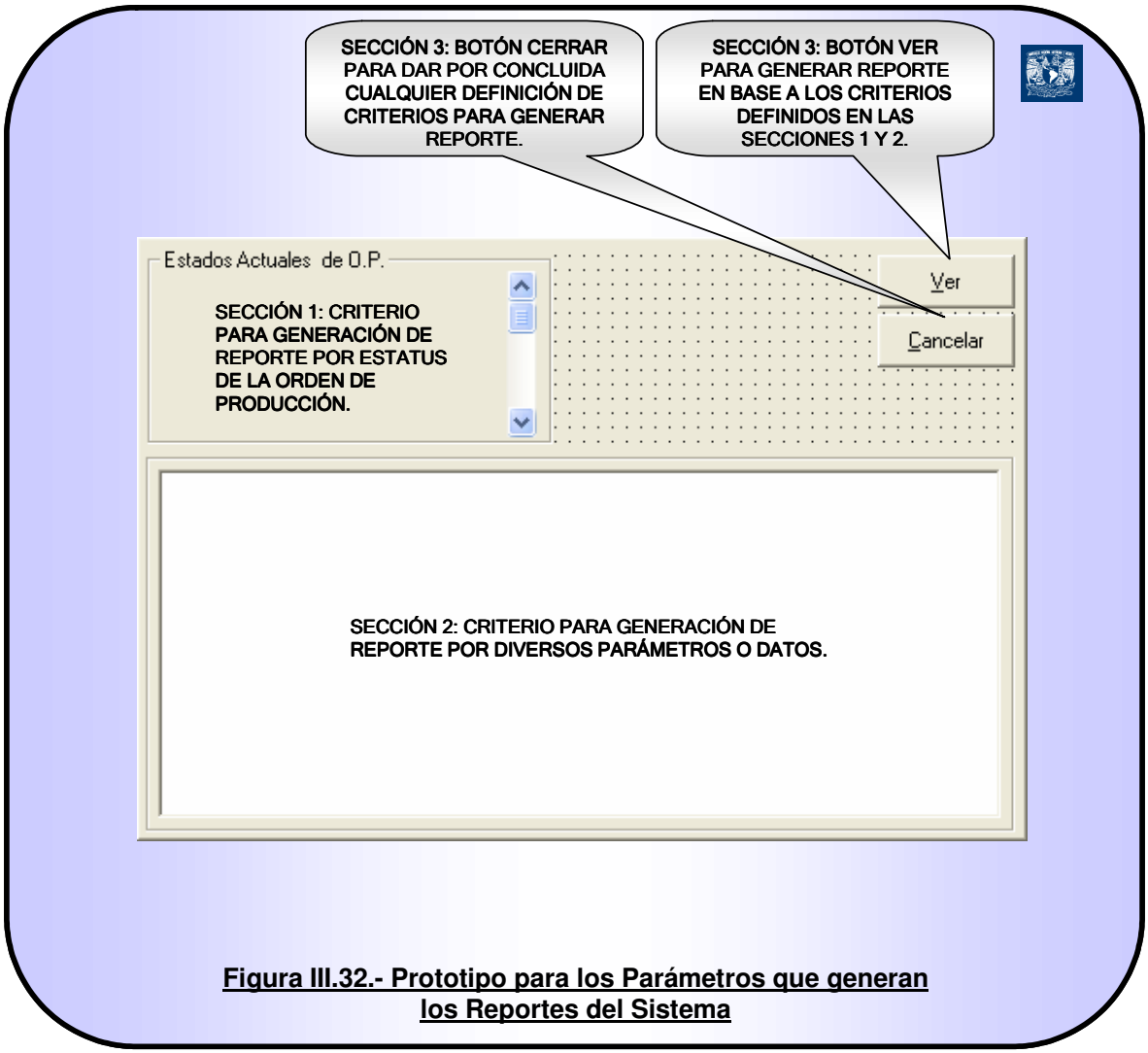



**Figura III.32.- Prototipo para los Parámetros que generan los Reportes del Sistema**

### III.6.3.2 Reportes

En la figura III.33 se observa lo que es el prototipo del Reporte obtenido a partir de los datos ingresados en el objeto de parámetros para la emisión de un Reporte específico.

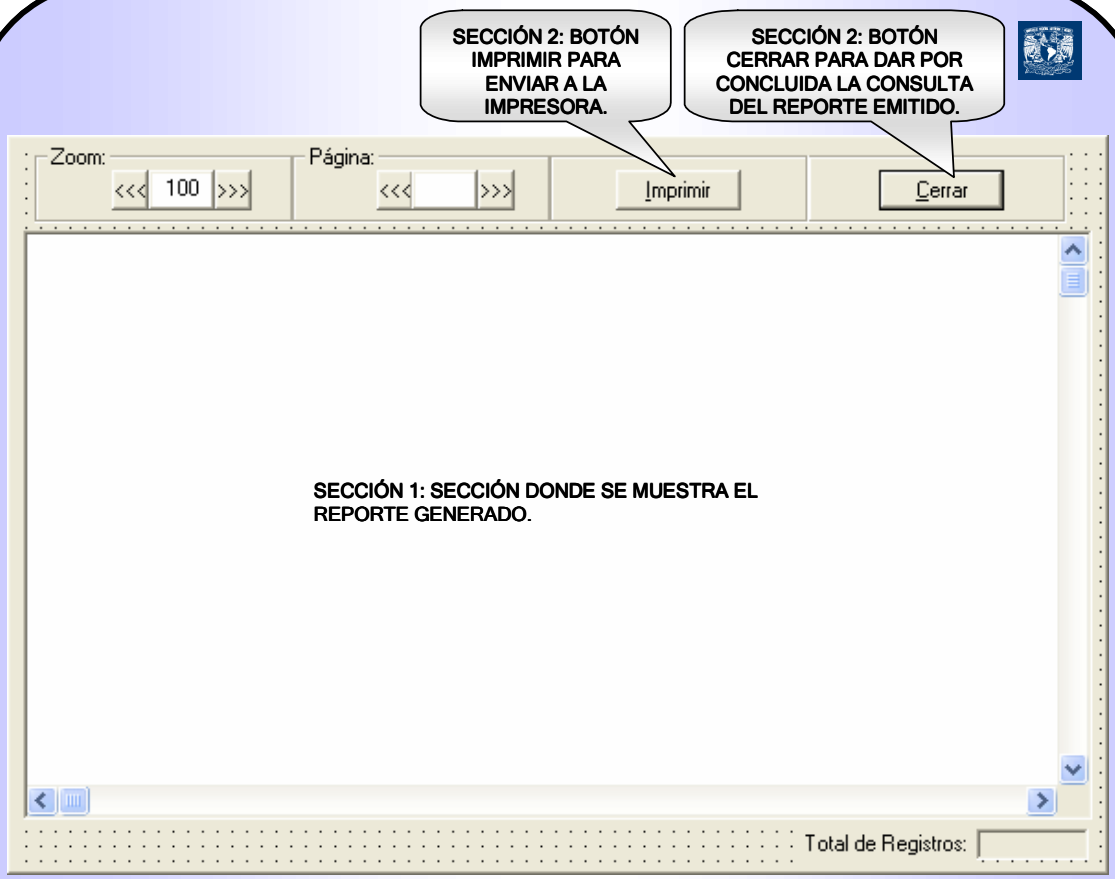



**Figura III.33.- Prototipo para los Reportes del Sistema**

A partir del desglose e identificación de requerimientos, se procedió a generar los correspondientes diagramas de modelamiento, y finalmente en esta etapa se han definido los prototipos de las Pantallas, cuya intención es mostrar la distribución idónea de todas las funcionalidades que ha de contener las principales operaciones del sistema.

Estos prototipos son de ámbito general, ya que a partir de estos se crearán programas, pantallas u objetos con una función propia; tal sería el caso de los distintos catálogos a generar, entre los cuales se encontrarán el de Unidades de Medida, Turnos, etcétera. Lo mismo aplica para los reportes cuya distribución será la misma a su correspondiente prototipo, sin embargo, cada uno ellos mostrará información única que los hará distintos entre sí, por ejemplo el reporte de explosión de materiales será distinto del de órdenes de producción, sin embargo su correspondiente pantalla tendrá la misma distribución que su prototipo. Y así sucesivamente para todos y cada uno de los prototipos que se han enunciado en esta parte del presente capítulo.

# *TEMA IV: Desarrollo*

OBJETIVO: Generar el Mapa Conceptual y Físico de los distintos objetos de Base de Datos que dan forma a los datos del sistema, así como sus Procesos Principales.

# IV.1 MODELO CONCEPTUAL DE DATOS

Cuando se diseña una base de datos, normalmente se comienza en el nivel conceptual. Un Modelo Conceptual de Datos representa la estructura lógica global de una base de datos, la cual es independiente de cualquier software o estructura de almacenamiento de datos. Un modelo conceptual comúnmente contiene objetos de datos que aún no han sido implementados físicamente en la base de datos. Es en sí una representación formal de los datos requeridos para realizar una actividad empresarial. De esto se parte sus objetivos son los siguientes:

- Representar la organización de datos en un formato gráfico.
- Verificar la validación del diseño de datos.
- Generar un Modelo Físico de Datos (PDM), el cual especifica la implementación física de la base datos.
- Y si así se desea, generar un Modelo Orientado a Objetos (OOM), el cual especifica la representación de objetos del Modelo Conceptual usando el estándar UML.

Por lo que el primer paso en el diseño de una Base de Datos es la creación del modelo conceptual. Inicialmente se examinan los diagramas de flujo de datos, que se pueden haber producido previamente, para identificar cada una de las áreas funcionales. La otra opción consiste en entrevistar a los usuarios, examinar los procedimientos, los informes y los formularios, y también observar el funcionamiento de la empresa.

El Modelo Conceptual de Datos se compone de entidades, relaciones, atributos, dominios de atributos e identificadores. Las etapas que conlleva la elaboración del Modelo Conceptual de Datos son las siguientes:

- Identificar las entidades.
- Identificar las relaciones.
- Identificar los atributos y asociarlos a entidades y relaciones.
- Determinar los dominios de los atributos.
- Determinar los identificadores.
- Determinar las jerarquías de generalización (si las hay).

## IV.1.1 IDENTIFICAR LAS ENTIDADES

En primer lugar hay que definir los principales objetos que interesan al usuario. Estos objetos serán las entidades. Una forma de identificar las entidades es examinar las especificaciones de requisitos de usuario. En estas especificaciones se buscan los nombres. También se buscan objetos importantes como personas, lugares o conceptos de interés, excluyendo aquellos nombres que sólo son propiedades de otros objetos.

Otra forma de identificar las entidades es buscar aquellos objetos que existen por sí mismos. Por ejemplo, *empleado* es una entidad porque los empleados existen, sepamos o no sus nombres, direcciones y teléfonos. Siempre que sea posible, el usuario debe colaborar en la identificación de las entidades.

A veces, es difícil identificar las entidades por la forma en que aparecen en las especificaciones de requisitos. Los usuarios, a veces, hablan utilizando ejemplos o analogías. En lugar de hablar de empleados en general, hablan de personas concretas, o bien, hablan de los puestos que ocupan esas personas.

Conforme se van identificando las entidades, se les dan nombres que tengan un significado y que sean obvias para el usuario. Los nombres de las entidades y sus descripciones se anotan en el diccionario de datos. Cuando sea posible, se debe anotar también el número aproximado de ocurrencias de cada entidad. Si una entidad se conoce por varios nombres, éstos se deben anotar en el diccionario de datos como alias o sinónimos.

## IV.1.2 IDENTIFICAR LAS RELACIONES

Una vez definidas las entidades, se deben definir las relaciones existentes entre ellas. Del mismo modo que para identificar las entidades se buscaban nombres en las especificaciones de requisitos, para identificar las relaciones se suelen buscar las expresiones verbales (por ejemplo: oficina tiene empleados, empleado gestiona inmueble, cliente visita inmueble). Si las especificaciones de requisitos reflejan estas relaciones es porque son importantes para la empresa y, por lo tanto, se deben reflejar en el modelo conceptual de datos.

Pero sólo interesan las relaciones que son necesarias. Por ejemplo, se han identificado las relaciones *empleado gestiona inmueble* y *cliente visita inmueble*. Se podría pensar en incluir una relación entre empleado y cliente: *empleado atiende a cliente*, pero observando las especificaciones de requisitos no parece que haya interés en modelar tal relación.

La mayoría de las relaciones son binarias (entre dos entidades), pero no hay que olvidar que también puede haber relaciones en las que participen más de dos entidades, así como relaciones recursivas.

Una vez identificadas todas las relaciones, hay que determinar la cardinalidad mínima y máxima con la que participa cada entidad en cada una de ellas. De este modo, el modelo conceptual representa de un modo más explícito la semántica de las relaciones. La cardinalidad es un tipo de restricción que se utiliza para comprobar y mantener la calidad de los datos. Estas restricciones son aserciones sobre las entidades que se pueden aplicar cuando se actualiza la base de datos para determinar si las actualizaciones violan o no las reglas establecidas sobre la semántica de los datos.

Conforme se van identificando las relaciones, se les van asignando nombres que tengan significado para el usuario. En el diccionario de datos se anotan los nombres de las relaciones, su descripción y las cardinalidades con las que participan las entidades en ellas.

### IV.1.3 IDENTIFICAR LOS ATRIBUTOS Y ASOCIARLOS A ENTIDADES Y RELACIONES

Al igual que con las entidades, se buscan nombres en las especificaciones de requisitos. Son atributos los nombres que identifican propiedades, cualidades, identificadores o características de entidades o relaciones.

Al identificar los atributos, hay que tener en cuenta si son simples o compuestos. El escoger entre atributo simple o compuesto depende de los requisitos del usuario. Si el usuario no necesita acceder a cada uno de los componentes de la dirección por separado, se puede representar como un atributo simple. Pero si el usuario quiere acceder a los componentes de forma individual, entonces se debe representar como un atributo compuesto.

También se deben identificar los atributos derivados o calculados, que son aquellos cuyo valor se puede calcular a partir de los valores de otros atributos. Por ejemplo, el número de empleados de cada oficina, la edad de los empleados o el número de inmuebles que gestiona cada empleado. Algunos diseñadores no representan los atributos derivados en los esquemas conceptuales. Si se hace, se debe indicar claramente que el atributo es derivado y a partir de qué atributos se obtiene su valor. Donde hay que considerar los atributos derivados es en el diseño físico.

Es muy útil elaborar una lista de atributos e ir eliminándolos de la lista conforme se vayan asociando a una entidad o relación. De este modo, uno se puede asegurar de que cada atributo se asocia a una sola entidad o relación, y que cuando la lista se ha acabado, se han asociado todos los atributos.

Conforme se van identificando los atributos, se les asignan nombres que tengan significado para el usuario. De cada atributo se debe anotar la siguiente información:

Nombre y descripción del atributo.

Alias o sinónimos por los que se conoce al atributo.

Tipo de dato y longitud.

Valores por defecto del atributo (si se especifican).

Si el atributo siempre va a tener un valor (si admite o no nulos).

Si el atributo es compuesto y, en su caso, qué atributos simples lo forman.

Si el atributo es derivado y, en su caso, cómo se calcula su valor.

Si el atributo es multievaluado.

### IV.1.4 DETERMINAR LOS DOMINIOS DE LOS ATRIBUTOS

El dominio de un atributo es el conjunto de valores que puede tomar el atributo. Un modelo conceptual de datos está completo si incluye los dominios de cada atributo: los valores permitidos para cada atributo, su tamaño y su formato.

### IV.1.5 DETERMINAR LOS IDENTIFICADORES

Cada entidad tiene al menos un identificador. En este paso, se trata de encontrar todos los identificadores de cada una de las entidades. Los identificadores pueden ser simples o compuestos. De cada entidad se escogerá uno de los identificadores como clave primaria en la fase del diseño lógico.

### IV.1.6 DETERMINAR LAS JERARQUÍAS DE GENERALIZACIÓN

Se deberá analizar las entidades que se han identificado para determinar si es necesario reflejar las diferencias entre distintas ocurrencias de una entidad, con lo que surgirán nuevas sub-entidades de esta entidad genérica; o bien, si hay entidades que tienen características en común y que realmente son sub-entidades de una nueva entidad genérica.

## IV.1.7 CARDINALIDAD

En la figura IV.1 se muestra la simbología propia de un Modelo Conceptual de Datos para definir la cardinalidad de las relaciones entre las distintas entidades.

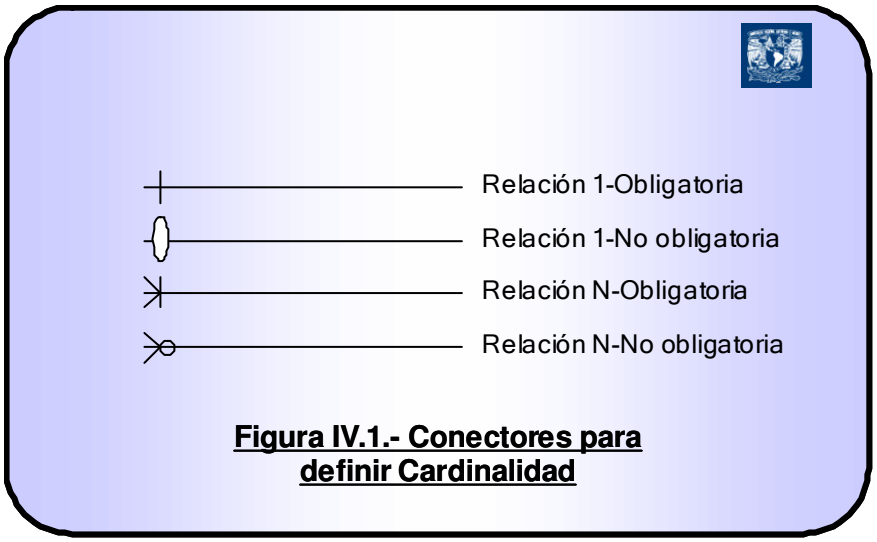



**Figura IV.1.- Conectores para definir Cardinalidad**

## IV.2 MODELO CONCEPTUAL DE DATOS DEL SISTEMA DE INFORMACIÓN PARA EL CONTROL DE PROCESOS DE PRODUCCIÓN

# *Aviso: CONSULTAR EL CONTENIDO DE LA HOJA 1 DEL ARCHIVO ANEXO No.1.*

# *Aviso: CONSULTAR EL CONTENIDO DE LA HOJA 2 DEL ARCHIVO ANEXO No.1.*

## IV.3 MODELO FÍSICO DE DATOS DEL SISTEMA DE INFORMACIÓN PARA EL CONTROL DE PROCESOS DE PRODUCCIÓN

# *Aviso: CONSULTAR EL CONTENIDO DE LA HOJA 1 DEL ARCHIVO ANEXO No. 2.*

# *Aviso: CONSULTAR EL CONTENIDO DE LA HOJA 2 DEL ARCHIVO ANEXO No. 2.*

# IV.4 DICCIONARIO DE DATOS

Un Diccionario de Datos es un catálogo de los elementos en un sistema. Dichos elementos se centran alrededor de los datos y la forma en que están estructurados para satisfacer los requerimientos de los usuarios y las necesidades de una empresa u organización. En un Diccionario de Datos se encuentra la lista de todos los elementos que forman parte del flujo de datos en todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario guarda los detalles y descripciones de todos estos elementos.

Ahora bien, dado que todas las partes de un sistema de información (transacciones, consultas, reportes, salidas, archivos y bases de datos), dependen de los datos, por lo tanto, el diccionario de datos contiene dos tipos de descripciones para el flujo de datos dentro del sistema, las cuales se mencionan a continuación. Los objetivos de un diccionario de datos son:

a) Manejar todos los detalles en sistemas grandes.

b) Comunicar el mismo significado para todos los elementos del sistema.

c) Documentar las características del sistema.

d) Facilitar el análisis de los detalles para evaluar las características y determinar dónde deben realizarse los cambios.

e) Localizar errores y omisiones en el sistema.

## IV.4.1 DATOS

Es la parte elemental de la información y son los bloques básicos de cualquier sistema, esta parte básica por sí misma no conlleva suficiente significado para ningún usuario. Un dato tiene una identificación propia conformada por las siguientes características:

- NOMBRE. Permite distinguir un dato de otro y se utiliza para hacer referencia a cada elemento durante todo el proceso de desarrollo de sistemas.

- DESCRIPCION. Permite identificar de manera breve lo que el dato representa en el sistema.

- ALIAS. Nombre alternativo del dato.

- LONGITUD. Espacio máximo necesario para cada dato.

- VALOR. Ocasionalmente un dato permite sólo valores específicos, tal es el caso de "S" o "N" para representar un SI o un NO.

### IV.4.2 ESTRUCTURA DE DATOS

Es un grupo datos elementales que están relacionados con otros y que en conjunto describen un componente del sistema. Dichas estructuras se construyen sobre cuatro relaciones de componentes; estos últimos pueden ser datos u otras estructuras de datos. Se pueden utilizar las siguientes combinaciones ya sea en forma individual o en conjunción con alguna otra.

- RELACIÓN SECUENCIAL. Define los componentes (datos u otras estructuras de datos) que siempre se incluyen en una estructura de datos en particular; concatenación de dos o más datos.

- RELACIÓN DE SELECCIÓN (Uno u Otro). Define alternativas para datos o estructuras de datos incluidas en una estructura de datos.

- RELACIÓN DE ITERACIÓN (Repetitiva). Define la repetición de un componente cero o más veces.

- RELACIÓN OPCIONAL. Es el caso especial de la iteración; los datos pueden estar o incluidos, esto es, una o ninguna iteración.

## IV.4.3 TABLAS DEL SISTEMA

| TABLA | NOMBRE TÉCNICO |
|---|---|
| Explosión de Materiales | prod_exp_materiales |
| Catálogo de Unidades de Medida | prod_cat_unidades_medida |
| Catálogo de Fases de Producción | prod_cat_fases_prod |
| Orden de Producción | prod_orden_prod |
| Catálogo de Turnos | prod_cat_turnos |
| Catálogo de Estados OP | prod_cat_estados_op |
| Catálogo de Líneas de Ensamble | prod_cat_lineas_ensamble |
| Plan de Producción | prod_plan_prod |
| Producción Diaria | prod_prod_diaria |
| Tiempos Muertos | prod_tiempos_muertos |
| Catalogo de Motivos de Tiempos Muertos | prod_cat_mot_tiempos_muertos |
| Avance Orden de Producción | prod_avance_orden_prod |
| Materiales Adicionales a Orden de Producción | prod_mat_adic_orden_prod |
| Catalogo de Tipos de Producción | prod_cat_tipos_prod |
| Merma de la Orden de Producción | prod_merma_orden_prod |
| Productos Terminados | prod_prod_terminados |
| Catalogo de Procesos de Producción | prod_cat_proc_prod |
| Proceso y Fases Asignadas a la Orden de Producción | prod_procfas_orden_prod |
| Bitácora de Orden de Producción | prod_bit_orden_prod |
| Artículo vs Proceso | prod_art_vs_proceso |
| Dependencia entre Fases | prod_depend_fases |
| Bitácora Cambio de Material | prod_bit_cambio_materiales |
| Basura Explosión de Materiales | prod_bas_exp_materiales |
| Basura Bitácora Cambio de Material | prod_bas_bit_cambio_materiales |
| Bitácora de Plan de Producción | prod_bit_plan_prod |
| Detalle de Orden de Producción | prod_det_orden_prod |

## IV.4.4 COLUMNAS DE LA TABLA ARTÍCULO VS PROCESO

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Articulo | Clave del Artículo asociadoa a un Proceso de Producción específico. | char(15) |
| Tipo de Material | Tipo de Material asignado al Artículo: Componente (CO), Insumo (IN), Producto Terminado (PT). | char(2) |
| Unidad de Medida | Unidad de Medida asignada al Artículo. | smallint |
| Proceso Asociado | Si el artículo es de tipo Producto Terminado (PT) o Componente (CO) podrá asociarse un proceso, sino no se asociará en caso de que el tipo sea Insumo (IN). Por lo tanto, no se define Llave Foránea. | numeric(7,0) |
| Empleado de Alta | Empleado que genera el nuevo registro de la relación Artículo - Proceso. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro de la relación Artículo - Proceso. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la relación Artículo - Proceso o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la relación Artículo - Proceso o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(255) |

## IV.4.5 COLUMNAS DE LA TABLA AVANCE DE ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Produccion | Folio de la OP. | numeric(7,0) |
| Folio de Asignación de la OP | Número del registro para la bitácora del Avance de la OP. | numeric(7,0) |
| Componente | Clave del Componente o Insumo cuyas cantidades producidas se contabilizan para el registro de la Bitácora del Avance de la OP. | char(15) |
| Cantidad | Cantidad del Componente o Insumo cuyas cantidades producidas se contabilizan para el registro de la Bitácora del Avance de la OP. | decimal(11,3) |
| Empleado | Empleado que realiza el registro de la Bitácora de Avance. | numeric(7,0) |
| Fecha de Bitácora | Fecha en la que se registra en esta Bitácora. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | text |

## IV.4.6 COLUMNAS DE LA TABLA BASURA BITÁCORA CAMBIO DE MATERIAL

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro en la Basura. | numeric(7,0) |
| Identificador Original | Identificador o Folio del Registro. | numeric(7,0) |
| Clave Padre | Clave del Artículo a integrar. | char(15) |
| Clave Hijo | Clave del Nuevo Componente del Artículo a integrar. | char(15) |
| Clave Hijo Anterior | Clave del Componente Anterior del Artículo a integrar. | char(15) |
| Tipo de Material Anterior | Tipo de Material (CO-Componente, IN-Insumo, PT-Producto Terminado) del Componente Anterior del Artículo a integrar. | char(2) |
| Cantidad Anterior | Cantidad del Componente Anterior del Artículo a integrar. | decimal(11,3) |
| Unidad de Medida Anterior | Unidad de Medida del Componente Anterior del Artículo a integrar. | smallint |
| Empleado | Empleado que realizó el cambio del Componente. | numeric(7,0) |
| Fecha de Bitácora | Fecha en que se hizo el cambio de Componente. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que realizó el Cambio de Materiales. | char(250) |

## IV.4.7 COLUMNAS DE LA TABLA BASURA EXPLOSIÓN DE MATERIALES

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | numeric(7,0) |
| Clave Padre | Clave del Artículo a integrar. | char(15) |
| Clave Hijo | Clave del Componente del Artículo a integrar. | char(15) |
| Tipo de Material | Tipo de Material (CO-Componente, IN-Insumo, PT-Producto Terminado) del Componente del Artículo a integrar. | char(2) |
| Opcional de | Clave del Componente Principal. | char(15) |
| Cantidad | Cantidad de Componentes de esta Clave que conformarán el Artículo a integrar. | decimal(11,3) |
| Unidad de Medida | Unidad de Medida del Componente del Artículo a integrar. | smallint |
| Importancia | Indicativo de que el Componente es el Principal (P) u Opcional (O). | char(1) |
| Empleado de Alta | Empleado que estructuró la lista de Materiales del Artículo a integrar, con este Componente. | numeric(7,0) |
| Fecha de Alta | Fecha en que el Empleado estructuró la lista de Materiales del Artículo a integrar, con este Componente. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de este Componente o bien aquel que la deshabilitó para no integrar temporalmente al Artículo a Fabricar o Ensamblar. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de este Componente o bien, fecha cuando lo deshabilitó para no integrar temporalmente al Artículo a Fabricar o Ensamblar. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.8 COLUMNAS DE LA TABLA BITÁCORA DE ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio Orden de Producción | Folio de la OP. | numeric(7,0) |
| Folio de Bitácora | Número del registro para la bitácora del Avance de la OP estatus por estatus. | numeric(7,0) |
| Estado | Estatus que tenía la OP al momento de registrar esta bitácora. | smallint |
| Empleado | Empleado que registró esta bitácora. | numeric(7,0) |
| Fecha | Fecha en que se registró esta bitácora. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que modifica o registra en esta bitácora.. | char(250) |

## IV.4.9 COLUMNAS DE LA TABLA BITÁCORA DE CAMBIO DE MATERIAL

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | numeric(7,0) |
| Clave Padre | Clave del Artículo a integrar. | char(15) |
| Clave Hijo | Clave del Nuevo Componente del Artículo a integrar. | char(15) |
| Clave Hijo Anterior | Clave del Componente Anterior del Artículo a integrar. | char(15) |
| Tipo de Material Anterior | Tipo de Material (CO-Componente, IN-Insumo, PT-Producto Terminado) del Componente Anterior del Artículo a integrar. | char(2) |
| Cantidad Anterior | Cantidad del Componente Anterior del Artículo a integrar. | decimal(11,3) |
| Unidad de Medida Anterior | Unidad de Medida del Componente Anterior del Artículo a integrar. | smallint |
| Empleado | Empleado que realizó el cambio del Componente. | numeric(7,0) |
| Fecha de Bitácora | Fecha en que se hizo el cambio de Componente. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que realizó el Cambio de Materiales. | char(250) |

## IV.4.10 COLUMNAS DE LA TABLA BITÁCORA DE PLAN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | numeric(7,0) |
| Periodo | Período Año - Mes del registro de Cantidades Totales planeadas para producirse en un día dado. | char(6) |
| Producto Final | Clave del Producto a producir. | char(15) |
| Dia | Día de la producción que ha sufrido cambios en el Plan de Producción. | char(2) |
| Cantidad Actual | Nueva Cantidad planeada. | decimal(11,3) |
| Cantidad Anterior | Antigua Cantidad planeada. | decimal(11,3) |
| Empleado | Empleado que realizó el cambio. | numeric(7,0) |
| Fecha | Fecha del cambio. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que modifica el Plan de Producción. | char(250) |

## IV.4.11 COLUMNAS DE LA TABLA CATALOGO DE MOTIVOS DE TIEMPOS MUERTOS

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | numeric(7,0) |
| Descripción | Significado o descripción del Tiempo Muerto. | char(250) |
| Empleado de Alta | Empleado que genera el nuevo registro para un nuevo Tiempo Muerto. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con un nuevo Tiempo Muerto. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro del Tiempo Muerto, o bien aquel que lo deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro del Tiempo Muerto, o bien fecha en que lo deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.12 COLUMNAS DE LA TABLA CATALOGO DE PROCESOS DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | numeric(7,0) |
| Descripcion | Significado o descripción del Proceso. | char(250) |
| Empleado de Alta | Empleado que genera el nuevo registro para un nuevo Proceso. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con un nuevo Proceso. | datetime |
| Empleado de Baja o Modificacion | Empleado que realizó la última modificación sobre el registro del Proceso o bien aquel que lo deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificacion | Fecha en que el Empleado realizó la última modificación sobre el registro del Proceso o bien fecha en que lo deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.13 COLUMNAS DE LA TABLA CATALOGO TIPOS DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Clave | Clave que identifica al Tipo de Producción. | char(2) |
| Descripción | Significado o descripción del Tipo de Producción. | char(250) |
| Empleado de Alta | Empleado que genera el nuevo registro para un nuevo Tipo de Producción. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con un nuevo Tipo de Producción. | datetime |
| Empleado de Baja o Modificacion | Empleado que realizó la última modificación sobre el registro del Tipo de Producción o bien aquel que lo deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificacion | Fecha en que el Empleado realizó la última modificación sobre el registro del Tipo de Producción o bien fecha en que lo deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.14 COLUMNAS DE LA TABLA CATÁLOGO DE ESTADOS OP

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Clave del Estatus. | smallint |
| Descripción | Descripción del Estatus. | char(50) |
| Equivalente a | Significado o Descripción Original del Estatus. | char(50) |
| Empleado de Alta | Empleado que genera el nuevo registro para un nuevo Estatus. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con un nuevo Estatus. | datetime |
| Empleado de Modificación | Empleado que realizó la última modificación sobre el registro del Estatus. | numeric(7,0) |
| Fecha de Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro del Estatus. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.15 COLUMNAS DE LA TABLA CATÁLOGO DE FASES DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador Fase | Identificador o Folio del Registro. | numeric(7,0) |
| Descripción | Significado o descripción de la Fase de Producción. | char(250) |
| Empleado de Alta | Empleado que genera el nuevo registro para una nueva Fase. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con una nueva Fase. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la Fase de Producción o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la Fase de Producción o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.16 COLUMNAS DE LA TABLA CATÁLOGO DE LÍNEAS DE ENSAMBLE

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | smallint |
| Descripción | Significado o descripción de la Línea de Ensamble. | char(50) |
| Empleado de Alta | Empleado que genera el nuevo registro para una nueva Línea de Ensamble. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con una nueva Línea de Ensamble. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la Línea de Ensamble o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la Línea de Ensamble o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.17 COLUMNAS DE LA TABLA CATÁLOGO DE TURNOS

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | smallint |
| Descripción | Significado o descripción del Turno. | char(30) |
| Cantidad Empleados | Cantidad de Empleados asignados al Turno. | numeric(7,0) |
| Hora Inicio | Hora en que comienza el Turno. | datetime |
| Hora Fin | Hora en que finaliza el Turno. | datetime |
| Empleado de Alta | Empleado que genera el nuevo registro para un nuevo Turno. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con un nuevo Turno. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro del Turno o bien aquel que lo deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro del Turno o bien fecha en que lo deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.18 COLUMNAS DE LA TABLA CATÁLOGO DE UNIDADES DE MEDIDA

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Identificador | Identificador o Folio del Registro. | smallint |
| Clave Unidad | Clave asignada por el usuario en base al Sistema Internacional de Medidas. | char(3) |
| Descripción | Significado de la Unidad de Medida. | char(30) |
| Empleado de Alta | Empleado que genera el nuevo registro con una Unidad de Medida. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con una Unidad de Medida. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la Unidad de Medida o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la Unidad de Medida o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.19 COLUMNAS DE LA TABLA DEPENDENCIA ENTRE FASES

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Proceso | Proceso que engloba a las Fases asociadas. | numeric(7,0) |
| Fase Dependiente | Fase de Producción que sigue a continuación de una Fase Anterior. | numeric(7,0) |
| Fase Independiente | Fase de Producción que antecede a una Fase Posterior. | numeric(7,0) |
| Porcentaje del Proceso | Porcentaje de Tiempo o Importancia de la Fase dentro del Proceso de Producción. | tinyint |
| Empleado de Alta | Empleado que genera el nuevo registro para una nueva Dependencia entre Fases de Producción. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado genera el nuevo registro con una nueva Dependencia entre Fases de Producción. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la Dependencia o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la Dependencia o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.20 COLUMNAS DE LA TABLA DETALLE DE ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Producción | Folio de la OP. | numeric(7,0) |
| Folio de Detalle de Orden de Producción | Número de registro del Detalle. | numeric(7,0) |
| Nivel | Nivel de Explosión de Materiales. | tinyint |
| Clave Padre | Clave del Producto a integrar. | char(15) |
| Importancia | Importancia del Componente o Insumo: Principal (P) u Opcional (O). | char(1) |
| Obliga Sugiere Impide Fabricar | Indicativo para obligar, sugerir o impedir la fabricación de un componente de nivel superior. | char(1) |
| Integrar Fabricar o Ensamblar | Indica si deberá fabricarse el Componente o bien al tener existencias de este componente no será necesario fabricarlo. | char(1) |
| Clave Hijo | Clave del Componente que integra a otro Componente de nivel superior. | char(15) |
| Opcionales | Componentes Opcionales. | char(1) |
| Costo Unitario | Costo Unitario del Componente. | decimal(11,3) |
| Cantidad Requerida | Cantidad requerida de Componentes para fabricar o ensamblar el Producto Terminado de la OP. | decimal(11,3) |
| Cantidad Máxima Permitida | Cantidad Máxima permitida de un mismo componente. | decimal(11,3) |
| Unidad de Medida | Unidad de Medida del Componente. | smallint |
| Existencia | Existencias actuales al surtir la OP. | decimal(11,3) |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Cantidad Utilizada | Cantidad utilizada durante el proceso de producción. | decimal(11,3) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o genera la OP. | char(250) |

## IV.4.21 COLUMNAS DE LA TABLA EXPLOSION DE MATERIALES

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Clave Padre | Clave del Artículo a integrar. | char(15) |
| Clave Hijo | Clave del Componente del Artículo a integrar. | char(15) |
| Tipo de Material | Tipo de Material (CO-Componente, IN-Insumo, PT-Producto Terminado) del Componente del Artículo a integrar. | char(2) |
| Opcional de | Clave del Componente Principal. | char(15) |
| Cantidad | Cantidad de Componentes de esta Clave que conformarán el Artículo a integrar. | decimal(11,3) |
| Unidad de Medida | Unidad de Medida del Componente del Artículo a integrar. | smallint |
| Importancia | Indicativo de que el Componente es el Principal (P) u Opcional (O). | char(1) |
| Merma | Indica si el producto va a absorber el costo por Merma. | char(1) |
| Serializable | Indicativo de que se deberá ingresar el No. de Serie del Componente Hijo. | char(1) |
| Empleado de Alta | Empleado que estructuró la lista de Materiales del Artículo a integrar, con este Componente. | numeric(7,0) |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Fecha de Alta | Fecha en que el Empleado estructuró la lista de Materiales del Artículo a integrar, con este Componente. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de este Componente o bien aquel que la deshabilitó para no integrar temporalmente al Artículo a Fabricar o Ensamblar. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de este Componente o bien, fecha cuando lo deshabilitó para no integrar temporalmente al Artículo a Fabricar o Ensamblar. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.22 COLUMNAS DE LA TABLA MATERIALES ADICIONALES A ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Producción | Folio de la OP. | numeric(7,0) |
| Folio de Orden de Producción Adicional | Número de Folio de la OP Adicional para la OP. | numeric(7,0) |
| Componente o Insumo | Clave del Componente que se utiliza de Material Adicional. | char(15) |
| Cantidad Solicitada | Cantidad adicional requerida. | decimal(11,3) |
| Cantidad Surtida | Cantidad surtida en base a la cantidad solicitada. | decimal(11,3) |
| Empleado de Alta | Empleado que registra la petición de Material Adicional. | numeric(7,0) |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Fecha de Alta | Fecha en que el empleado registra la petición de Material Adicional. | datetime |
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre el registro de la petición de Material Adicional o bien aquel que la deshabilitó. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre el registro de la Petición de Material Adicional o bien fecha en que la deshabilitó. | datetime |
| Estado | Estatus que guarda el registro (Habilitado o Deshabilitado). | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.23 COLUMNAS DE LA TABLA MERMA DE LA ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Producción | Folio de la OP para la cual se registra la Merma. | numeric(7, 0) |
| Folio de Asignación de la OP | Número de Registro para una misma OP. | numeric(7, 0) |
| Componente o Insumo | Clave del Componente o Insumo que se integra a la Merma. | char(15) |
| Cantidad | Cantidad de Componentes o Insumos en base a la Unidad de Medida. | decimal(11 ,3) |
| Empleado | Empleado que realiza el registro de la Merma. | numeric(7, 0) |
| Fecha | Fecha en la que se registra la Merma. | datetime |
| Observaciones | Observaciones que considere útiles el empleado que modifica o da de alta. | char(250) |

## IV.4.24 COLUMNAS DE LA TABLA ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Producción | Folio de la OP. | numeric(7,0) |
| Fecha de Producción | Fecha definida para comenzar a fabricar o ensamblar el Producto asignado a la OP. | datetime |
| Periodo | Período Año - Mes de la Fecha en que comenzará a fabricarse o ensamblarse el Artículo de la OP. | char(6) |
| Dia | Día de la Fecha en que comenzará a fabricarse o ensamblarse el Artículo de la OP. | smallint |
| Turno | Turno asignado a la OP. | smallint |
| Línea de Ensamble | Línea de Ensamble asignada a la OP. | smallint |
| Tipo de Produccion | Tipo de Producción definida para la OP. | char(2) |
| Producto Final | Clave del Producto Final o a fabricarse. | char(15) |
| Cantidad | Cantidad a producir. | decimal(11,3) |
| Unidad de Medida | Unidad de Medida asignada al Producto a producir. | smallint |
| Proceso de Producción | Proceso de Producción asignado a la OP. | numeric(7,0) |
| Costo Inicial | Costo inicial con que se genera la OP. | decimal(11,3) |
| Costo Final | Costo Final con que se concluye la OP. | decimal(11,3) |
| Moneda | Tipo de Moneda asignada al Artículo de la OP. | char(3) |
| Merma | Indica si el producto va a absorber el costo por Merma. | char(1) |
| Serializable | Indicativo de que el Producto está constituido por Componentes con No. de Serie registrado. | char(1) |
| Número de Lote | Número de Lote de producción que corresponde a esta OP. | char(50) |
| Origen Plan de Producción | A partir de dónde se generó esta OP: Plan de Producción o Manualmente. | char(1) |
| Detenida | | char(1) |
| Empleado de Alta | Empleado que generó la OP. | numeric(7,0) |
| Fecha de Alta | Fecha en que el empleado generó la OP. | datetime |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Empleado de Baja o Modificación | Empleado que realizó la última modificación sobre la OP. | numeric(7,0) |
| Fecha de Baja o Modificación | Fecha en que el Empleado realizó la última modificación sobre la OP. | datetime |
| Estado | Estatus de la OP. | Smallint |
| Observaciones | Observaciones que considere útiles el empleado que modifica o genera la OP. | char(250) |

## IV.4.25 COLUMNAS DE LA TABLA PLAN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Periodo | Período Año - Mes del registro de Cantidades Totales planeadas para producirse en un día dado. | char(6) |
| Producto Final | Clave del Producto a producir. | char(15) |
| Dia 01 | Cantidad Total a producir para el día 1 del Mes. | decimal(11,3) |
| Dia 02 | Cantidad Total a producir para el día 2 del Mes. | decimal(11,3) |
| Dia 03 | Cantidad Total a producir para el día 3 del Mes. | decimal(11,3) |
| Dia 04 | Cantidad Total a producir para el día 4 del Mes. | decimal(11,3) |
| Dia 05 | Cantidad Total a producir para el día 5 del Mes. | decimal(11,3) |
| Dia 06 | Cantidad Total a producir para el día 6 del Mes. | decimal(11,3) |
| Dia 07 | Cantidad Total a producir para el día 7 del Mes. | decimal(11,3) |
| Dia 08 | Cantidad Total a producir para el día 8 del Mes. | decimal(11,3) |
| Dia 09 | Cantidad Total a producir para el día 9 del Mes. | decimal(11,3) |
| Dia 10 | Cantidad Total a producir para el día 10 del Mes. | decimal(11,3) |
| Dia 11 | Cantidad Total a producir para el día 11 del Mes. | decimal(11,3) |
| Dia 12 | Cantidad Total a producir para el día 12 del Mes. | decimal(11,3) |
| Dia 13 | Cantidad Total a producir para el día 13 del Mes. | decimal(11,3) |
| Dia 14 | Cantidad Total a producir para el día 14 del Mes. | decimal(11,3) |
| Dia 15 | Cantidad Total a producir para el día 15 del Mes. | decimal(11,3) |
| Dia 16 | Cantidad Total a producir para el día 16 del Mes. | decimal(11,3) |
| Dia 17 | Cantidad Total a producir para el día 17 del Mes. | decimal(11,3) |
| Dia 18 | Cantidad Total a producir para el día 18 del Mes. | decimal(11,3) |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Dia 19 | Cantidad Total a producir para el día 19 del Mes. | decimal(11,3) |
| Dia 20 | Cantidad Total a producir para el día 20 del Mes. | decimal(11,3) |
| Dia 21 | Cantidad Total a producir para el día 21 del Mes. | decimal(11,3) |
| Dia 22 | Cantidad Total a producir para el día 22 del Mes. | decimal(11,3) |
| Dia 23 | Cantidad Total a producir para el día 23 del Mes. | decimal(11,3) |
| Dia 24 | Cantidad Total a producir para el día 24 del Mes. | decimal(11,3) |
| Dia 25 | Cantidad Total a producir para el día 25 del Mes. | decimal(11,3) |
| Dia 26 | Cantidad Total a producir para el día 26 del Mes. | decimal(11,3) |
| Dia 27 | Cantidad Total a producir para el día 27 del Mes. | decimal(11,3) |
| Dia 28 | Cantidad Total a producir para el día 28 del Mes. | decimal(11,3) |
| Dia 29 | Cantidad Total a producir para el día 29 del Mes. | decimal(11,3) |
| Dia 30 | Cantidad Total a producir para el día 30 del Mes. | decimal(11,3) |
| Dia 31 | Cantidad Total a producir para el día 31 del Mes. | decimal(11,3) |

## IV.4.26 COLUMNAS DE LA TABLA PROCESO Y FASES ASIGNADAS A LA ORDEN DE PRODUCCIÓN

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio Orden de Producción | Folio de la OP. | numeric(7,0) |
| Folio de Asignación de la OP | Número de Registro para una misma OP. | numeric(7,0) |
| Proceso de Producción | Proceso de Producción asociado a la OP. | numeric(7,0) |
| Orden de Ejecución | Orden en que se va a ejecutar esta Fase dentro del Proceso de Producción. | numeric(7,0) |
| Fase | Fase que integra al Proceso asociado para la OP. | numeric(7,0) |
| Incluir | Indicativo de incluir o no esta Fase al Proceso de Producción de esta OP. | char(1) |

## IV.4.27 COLUMNAS DE LA TABLA PRODUCCIÓN DIARIA

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Periodo | Período Año - Mes del registro de Cantidades Totales producidas para un día dado. | char(6) |
| Producto Final | Clave del Producto producido. | char(15) |
| Dia 01 | Cantidad Total producida el día 1 del Mes. | decimal(11,3) |
| Dia 02 | Cantidad Total producida el día 2 del Mes. | decimal(11,3) |
| Dia 03 | Cantidad Total producida el día 3 del Mes. | decimal(11,3) |
| Dia 04 | Cantidad Total producida el día 4 del Mes. | decimal(11,3) |
| Dia 05 | Cantidad Total producida el día 5 del Mes. | decimal(11,3) |
| Dia 06 | Cantidad Total producida el día 6 del Mes. | decimal(11,3) |
| Dia 07 | Cantidad Total producida el día 7 del Mes. | decimal(11,3) |
| Dia 08 | Cantidad Total producida el día 8 del Mes. | decimal(11,3) |
| Dia 09 | Cantidad Total producida el día 9 del Mes. | decimal(11,3) |
| Dia 10 | Cantidad Total producida el día 10 del Mes. | decimal(11,3) |
| Dia 11 | Cantidad Total producida el día 11 del Mes. | decimal(11,3) |
| Dia 12 | Cantidad Total producida el día 12 del Mes. | decimal(11,3) |
| Dia 13 | Cantidad Total producida el día 13 del Mes. | decimal(11,3) |
| Dia 14 | Cantidad Total producida el día 14 del Mes. | decimal(11,3) |
| Dia 15 | Cantidad Total producida el día 15 del Mes. | decimal(11,3) |
| Dia 16 | Cantidad Total producida el día 16 del Mes. | decimal(11,3) |
| Dia 17 | Cantidad Total producida el día 17 del Mes. | decimal(11,3) |

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Dia 18 | Cantidad Total producida el día 18 del Mes. | decimal(11,3) |
| Dia 19 | Cantidad Total producida el día 19 del Mes. | decimal(11,3) |
| Dia 20 | Cantidad Total producida el día 20 del Mes. | decimal(11,3) |
| Dia 21 | Cantidad Total producida el día 21 del Mes. | decimal(11,3) |
| Dia 22 | Cantidad Total producida el día 22 del Mes. | decimal(11,3) |
| Dia 23 | Cantidad Total producida el día 23 del Mes. | decimal(11,3) |
| Dia 24 | Cantidad Total producida el día 24 del Mes. | decimal(11,3) |
| Dia 25 | Cantidad Total producida el día 25 del Mes. | decimal(11,3) |
| Dia 26 | Cantidad Total producida el día 26 del Mes. | decimal(11,3) |
| Dia 27 | Cantidad Total producida el día 27 del Mes. | decimal(11,3) |
| Dia 28 | Cantidad Total producida el día 28 del Mes. | decimal(11,3) |
| Dia 29 | Cantidad Total producida el día 29 del Mes. | decimal(11,3) |
| Dia 30 | Cantidad Total producida el día 30 del Mes. | decimal(11,3) |
| Dia 31 | Cantidad Total producida el día 31 del Mes. | decimal(11,3) |

## IV.4.28 COLUMNAS DE LA TABLA PRODUCTOS TERMINADOS

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio de Orden de Produccion | Folio de la OP. | numeric(7,0) |
| Consecutivo | | numeric(7,0) |
| Producto Final | Clave del Producto Final. | char(15) |
| Numero de Serie | Número de Serie de un Producto Terminado. | char(50) |

## IV.4.29 COLUMNAS DE LA TABLA TIEMPOS MUERTOS

| COLUMNA | PROPÓSITO | TIPO DE DATO |
|---|---|---|
| Folio OP | Folio de la OP. | numeric(7,0) |
| Folio Tiempo Muerto | Tiempo Muerto registrado. | numeric(7,0) |
| Fecha | Fecha en que se suscitó el Tiempo Muerto. | datetime |
| Turno | Turno en que se presentó el Tiempo Muerto. | smallint |
| Motivo | POR DEFINIR. | numeric(7,0) |
| Empleado de Inicio | | numeric(7,0) |
| Fecha y Hora de Inicio | Hora en que comenzó el Tiempo Muerto. | datetime |
| Empleado de Conclusión | | numeric(7,0) |
| Fecha y Hora de Conclusión | Hora en que terminó el Tiempo Muerto. | datetime |
| Estado | | char(1) |
| Observaciones | Observaciones que considere útiles el empleado que registra el Tiempo Muerto. | char(250) |

# IV.5 PRINCIPALES PROCESOS

Si bien, este Sistema está compuesto por una infinidad de instrucciones y consideraciones escritas en el Lenguaje de programación PowerScript (propio de PowerBuilder para el desarrollo de aplicaciones Cliente-Servidor), varias de estas son similares, pues tan sólo contienen ciertas diferencias en código para cada caso en específico si así se requiere, tal es el caso de los procesos para Generar, Consultar o Modificar cualquier información, entre muchos otros procesos más. Por lo que, se ha considerado como primordial el mostrar tan sólo los siguientes procesos, ya que estos son la parte medular del objetivo principal que persigue el Sistema de Información para el Control de Procesos de Producción.

## IV.5.1 OF_EXPLOSIONA_MATERIALES

Desarrollado en PowerScript de PowerBuilder, su objetivo es explotar la lista de Materiales de un Producto específico, hasta su nivel más inferior, a través de una operación recursiva. La estructura de esta función es la siguiente y se utiliza en la opción “Orden de Producción” y “Explosión de Materiales”:

```
public function integer of_explosiona_materiales (string as_cveartpadre, integer ai_nivel, ref string as_mensaje)

CHAR lc_prodifo = 'S', lc_importancia, lc_importaux, lc_opcionales = 'S'

DATASTORE lds_1

DECIMAL{3} ldec_costounit, ldec_cantidad, ldec_existencia, ldec_cantidadtot

INTEGER li_retorno, li_idunimed, li_nivel

LONG ll_i, li_j, ll_renglondw2, ll_totregs, ll_totregslistmat

STRING ls_cveartpadre, ls_cvearthijo


lds_1 = CREATE DATASTORE

lds_1.DataObject = "dw_exp_mat_elementovsexist"

lds_1.SetTransObject(SQLCA)


ll_totregs = lds_1.Retrieve(as_cveartpadre, gi_almacenes)

IF ll_totregs > 0 THEN

  ll_totregslistmat = dw_2.RowCount()

  IF ai_nivel > 0 AND ll_totregslistmat > 0 THEN dw_2.SetItem(ll_totregslistmat, "prod_dop_ifo", lc_prodifo)
```

```
FOR ll_i=1 TO ll_totregs
  ls_cveartpadre = lds_1.GetItemString(ll_i, "prod_em_cve_padre")
  IF IsNull(ls_cveartpadre) OR Len(Trim(ls_cveartpadre)) <= 0 THEN
    as_mensaje = "No fue posible determinar los componentes del artículo: " + ls_cveartpadre + "."
    li_retorno = -1
    EXIT
  ELSE
    lc_importancia = lds_1.GetItemString(ll_i, "prod_em_importancia")
    ls_cvearthijo = lds_1.GetItemString(ll_i, "prod_em_cve_hijo")
    ldec_costounit = lds_1.GetItemDecimal(ll_i, "cat_articulos_art_cost_prom")
    ldec_cantidad = lds_1.GetItemDecimal(ll_i, "prod_em_cantidad")
    ls_cvearthijo = lds_1.GetItemString(ll_i, "prod_em_cve_hijo")
    li_idunimed = lds_1.GetItemNumber(ll_i, "prod_em_id_unidad")
    ll_renglondw2 = dw_2.inv_rowmanager.Of_InsertRow(0)
    dw_2.SetItem(ll_renglondw2, "prod_dop_cve_padre", ls_cveartpadre)
    dw_2.SetItem(ll_renglondw2, "prod_dop_importancia", lc_importancia)
    dw_2.SetItem(ll_renglondw2, "prod_dop_cve_hijo", ls_cvearthijo)
    dw_2.SetItem(ll_renglondw2, "prod_dop_costo_unit", ldec_costounit)
    ldec_cantidadtot = idec_cantprodterm * ldec_cantidad
    IF lc_importancia = 'P' THEN
      dw_2.SetItem(ll_renglondw2, "prod_dop_cantidad", ldec_cantidadtot)
    ELSE
      FOR li_j=ll_renglondw2 - 1 TO 1 STEP -1
        lc_importaux = dw_2.GetItemString(li_j, "prod_dop_importancia")
        IF lc_importaux = 'P' THEN
          dw_2.SetItem(li_j, "prod_dop_opcionales", lc_opcionales)
          EXIT
        END IF
```

```
        NEXT
      END IF
      dw_2.SetItem(ll_renglondw2, "prod_dop_cantmaxperm", ldec_cantidadtot)
      dw_2.SetItem(ll_renglondw2, "mov_existencia_exi_disponible", ldec_existencia)
      dw_2.SetItem(ll_renglondw2, "prod_dop_id_unidad", li_idunimed)
      dw_2.SetItem(ll_renglondw2, "prod_dop_nivel", ai_nivel)
      li_nivel = ai_nivel + 1
      li_retorno = Of_Explosiona_Materiales(ls_cvearthijo, li_nivel, as_mensaje)
      IF li_retorno < 0 THEN EXIT
    END IF
  NEXT
END IF
DESTROY lds_1
RETURN li_retorno
```

## IV.5.2 OF_OBTEN_CANTSOPS

Desarrollado en PowerScript de PowerBuilder, su objetivo es definir las cantidades opcionales a elegir, para cada uno de los componentes que integran un producto final, a cualquier nivel. La estructura de esta función es la siguiente y se utiliza en la opción “Ordenes de Producción”:

```
public subroutine of_obten_cantsops (decimal adec_cantmax, decimal adec_completos)
DECIMAL{3} ldec_i, ldec_cantops
STRING ls_opciones
ls_opciones = ls_opciones + String(0, "#,###,##0") + "~t" + String(0, "#,###,##0") + "/"
FOR ldec_i = 1 TO ( idec_cantprodterm - adec_completos )
  ldec_cantops = ldec_i * ( adec_cantmax / idec_cantprodterm )
  ls_opciones = ls_opciones + String(ldec_cantops, "#,###,##0") + "~t" + String(ldec_cantops, "#,###,##0") + "/"
NEXT
dw_2.Object.prod_dop_cantidad.values = ls_opciones
```

## IV.5.3 OF_BUSCA_ANCESTROREPETIDO

Desarrollado en PowerScript de PowerBuilder, su objetivo es evitar que un mismo componente lo integre el mismo. La estructura de esta función es la siguiente y se utiliza en la opción "Explosión de Materiales":

```
public function integer of_busca_ancestrorepetido (string as_cvearthijoact, string as_cvearthijo, ref string as_mensaje)

DECIMAL{3} ldec_i, ldec_cantops

STRING ls_opciones

ls_opciones = ls_opciones + String(0, "#,###,##0") + "~t" + String(0, "#,###,##0") + "/"

FOR ldec_i = 1 TO ( idec_cantprodterm - adec_completos )

   ldec_cantops = ldec_i * ( adec_cantmax / idec_cantprodterm )

   ls_opciones = ls_opciones + String(ldec_cantops, "#,###,##0") + "~t" + String(ldec_cantops, "#,###,##0") + "/"

NEXT

dw_2.Object.prod_dop_cantidad.values = ls_opciones
```

# IV.6 PROCEDIMIENTOS ALMACENADOS (STORE PROCEDURE)

Es un conjunto de instrucciones precompiladas de Transact-SQL almacenadas en una base de datos. Debido a que los procedimientos almacenados son precompilados, por lo regular ofrecen el mejor desempeño de cualquier tipo de consulta. Hay muchos procedimientos almacenados del sistema definidos con un prefijo "sp_" que reúnen información de tablas del sistema y que son en especial útiles para la administración. También se puede crear procedimientos almacenados propios, los cuales al crearse, implícitamente se ejecutan los siguientes 5 pasos:

- El procedimiento se descompondrá en las piezas que lo componen.
- Cuando los componentes hacen referencia otros objetos de la base de datos (tablas, vistas, etcétera) se verifican éstos para confirmar su existencia. Esto se conoce también como resolución.
- Cuando termina la resolución, el nombre del procedimiento se almacena en la tabla *sysobjects* y el código para crear el procedimiento almacenado se guarda en *syscomments*.
- La compilación continúa, y durante ella se crea un anteproyecto de cómo ejecutar la consulta. Con frecuencia a este anteproyecto se le denomina plan normalizado o árbol de consulta. El árbol de consulta se almacenará en la tabla *sysprocedures*.
- Cuando se ejecuta por primera vez el procedimiento almacenado, primero se leerá el plan de consulta y se optimizará por completo dentro de un plan de procedimiento y luego se ejecutará. Esto provocará un ahorro de tiempo al descomponer, resolver y compilar un árbol de consulta cada vez que se ejecute el procedimiento almacenado.

## IV.6.1 VENTAJAS DE LOS PROCEDIMIENTOS ALMACENADOS

- Después de ejecutado, el plan del procedimiento almacenado se almacenará en la caché de los procedimientos. Esto significa que la siguiente vez que se use ese mismo procedimiento almacenado en la misma sesión, se leerá y ejecutará directamente de la caché.
- Si se requiere, se pueden usar procedimientos almacenados para encapsular reglas de negocios. Después de encapsuladas, varias aplicaciones pueden usar estas reglas, dándole así una interfaz de datos consistente.
- Los procedimientos almacenados pueden recibir argumentos para obtener datos.
- Pueden ejecutarse automáticamente en una tarea a través del cron de la base de datos o cada vez que arranque el servidor de la base datos.

- Mediante ellos es posible extraer o modificar información.
- Un procedimiento almacenado debe ser invocado, ya sea desde una aplicación, tarea, secuencia de comandos o procesos por lotes.

## IV.6.2 SIA_PROD_ALTA_ACCESOS

Este procedimiento fue creado para generar automáticamente cada uno de los permisos a las opciones del menú que integran al sistema de información para el control de procesos de producción, y asociarlos a un grupo específico de usuarios. De tal manera que cada vez que se instale este sistema, se ejecute este proceso, del cual a continuación se muestra el código que lo integra.

```
CREATE PROCEDURE sia_prod_alta_accesos ( @psi_idmod SMALLINT, @pc_nombremod VARCHAR(20), @pc_usuario CHAR(20) ) AS



DECLARE @li_idopcion INTEGER, @li_idgrupo INTEGER, @lc_opmf CHAR(10), @lti_tabid TINYINT, @lc_descopm CHAR(40), @lc_tipoventana CHAR(1),

@lc_nombv CHAR(40), @lc_opmv CHAR(10), @lc_nombdw CHAR(40), @lc_tipoacc CHAR(1), @lti_contador TINYINT, @lti_totalops TINYINT



SELECT @lti_tabid = 1, @lc_tipoventana = 'M', @lc_tipoacc = 'N', @lti_totalops = 13



IF NOT EXISTS ( SELECT 1 FROM adm_modules WHERE mod_id = @psi_idmod )

  BEGIN

    INSERT INTO adm_modules

     ( mod_id, mod_ref_id, mod_lang_id, mod_name, mod_version, mod_release, mod_upgrade, mod_enabled, mod_bitmap, mod_adb_id )

    VALUES ( @psi_idmod, 0, 1, @pc_nombremod, 2, 0, 'N', 1, NULL, 0 )

    IF @@error != 0

      BEGIN

        SELECT -10

        RETURN

      END

  END
```

```
SELECT @li_idgrupo = adm_users.usr_grp_id

FROM adm_users, adm_groups

WHERE adm_users.usr_grp_id = adm_groups.grp_id

AND adm_groups.grp_enabled = 1

AND adm_users.name = @pc_usuario

IF @@rowcount < 1

  BEGIN

    SELECT -20

    RETURN

  END


SELECT @lti_contador = 1

WHILE @lti_contador <= @lti_totalops

  BEGIN

    IF @lti_contador = 1 SELECT @lc_opmf =  '1.1', @lc_descopm = 'Explosión de Materiales', @lc_nombv = 'w_explosion_materiales', @lc_opmv = '1.1', @lc_nombdw = 'dw_explosion_materiales'

    ELSE IF @lti_contador = 2 SELECT @lc_opmf =  '1.2', @lc_descopm = 'Plan de Producción', @lc_nombv = 'w_prod_plan_prod', @lc_opmv = '1.2', @lc_nombdw = 'dw_prod_plan_prod'

    ELSE IF @lti_contador = 3 SELECT @lc_opmf =  '1.3', @lc_descopm = 'Orden de Producción', @lc_nombv = 'w_prod_orden_prod', @lc_opmv = '1.3', @lc_nombdw = 'dw_prod_orden_prod'

    ELSE IF @lti_contador = 4 SELECT @lc_opmf =  '2.1', @lc_descopm = 'Unidades de Medida', @lc_nombv = 'w_cat_unidades_medida', @lc_opmv = '3.1', @lc_nombdw = 'dw_cat_unidades_medida_gral'

    ELSE IF @lti_contador = 5 SELECT @lc_opmf =  '2.2', @lc_descopm = 'Turnos', @lc_nombv = 'w_cat_turnos', @lc_opmv = '3.2', @lc_nombdw = 'dw_cat_turnos_gral'

    ELSE IF @lti_contador = 6 SELECT @lc_opmf =  '2.3', @lc_descopm = 'Lineas de Ensamble', @lc_nombv = 'w_cat_lineas_ensamble', @lc_opmv = '3.3', @lc_nombdw = 'dw_cat_lineas_ensamble_gral'

    ELSE IF @lti_contador = 7 SELECT @lc_opmf =  '2.4', @lc_descopm = 'Motivos de Tiempos Muertos', @lc_nombv = 'w_cat_motivos_tm', @lc_opmv = '3.4', @lc_nombdw = 'dw_cat_motivos_tm_gral'

    ELSE IF @lti_contador = 8 SELECT @lc_opmf =  '2.6', @lc_descopm = 'Tipos de Produccion', @lc_nombv = 'w_cat_tipos_produccion', @lc_opmv = '3.6', @lc_nombdw = 'dw_cat_tipos_produccion_gral'

    ELSE IF @lti_contador = 9 SELECT @lc_opmf =  '2.7', @lc_descopm = 'Estados de Orden de Produccion', @lc_nombv = 'w_cat_estados_op', @lc_opmv = '3.7', @lc_nombdw = 'dw_cat_estados_op_gral'
```

```
ELSE IF @lti_contador = 10 SELECT @lc_opmf =  '2.8', @lc_descopm = 'Fases de Produccion', @lc_nombv = 'w_cat_fases_produccion', @lc_opmv = '3.8', @lc_nombdw = 'dw_cat_fases_produccion_gral'

ELSE IF @lti_contador = 11 SELECT @lc_opmf =  '2.9', @lc_descopm = 'Procesos de Produccion', @lc_nombv = 'w_cat_procesos_produccion', @lc_opmv = '3.9', @lc_nombdw = 'dw_cat_procs_prod_gral'

ELSE IF @lti_contador = 12 SELECT @lc_opmf =  '2.10', @lc_descopm = 'Artículos vs Proceso', @lc_nombv = 'w_cat_art_vs_proc', @lc_opmv = '3.10', @lc_nombdw = 'dw_cat_art_vs_proc_gral'

ELSE IF @lti_contador = 13 SELECT @lc_opmf =  '3.1', @lc_descopm = 'Búsqueda y Eliminación de Materiales', @lc_nombv = 'w_busq_elim_materiales', @lc_opmv = '4.1', @lc_nombdw = 'dw_busq_elim_materiales'


SELECT @li_idopcion = NULL

SELECT @li_idopcion = mnu_id FROM adm_menuitems WHERE mnu_mod_id = @psi_idmod AND mnu_order4 = @lc_opmv


IF @li_idopcion = NULL OR @li_idopcion = 0

  BEGIN

    INSERT INTO adm_menuitems ( mnu_mod_id, mnu_order, mnu_tab_id, mnu_descrip, mnu_wintype, mnu_winname, mnu_wintitle, mnu_enabled, mnu_order4, mnu_dwname )

    VALUES ( @psi_idmod, @lc_opmf, @lti_tabid, @lc_descopm, @lc_tipoventana, @lc_nombv, @lc_descopm, 1, @lc_opmv, @lc_nombdw )

    IF @@error != 0

      BEGIN

        SELECT -30

        RETURN

      END

    SELECT @li_idopcion = @@identity

  END


IF NOT EXISTS ( SELECT 1 FROM adm_access WHERE acc_mod_id = @psi_idmod AND acc_mnu_id = @li_idopcion AND acc_grp_id = @li_idgrupo )

BEGIN

  INSERT INTO adm_access ( acc_type, acc_mod_id, acc_mnu_id, acc_grp_id )

  VALUES ( @lc_tipoacc, @psi_idmod, @li_idopcion, @li_idgrupo )
```

```
      IF @@error != 0
        BEGIN
          SELECT -30
          RETURN
        END
    END
  SELECT @lti_contador = @lti_contador + 1
END
;


IF EXISTS (SELECT  1 FROM sysobjects WHERE type = 'P' AND name = 'sia_prod_alta_accesos' )
    GRANT EXECUTE ON sia_prod_alta_accesos TO usu_sia ;
```

### IV.6.3 SIA_PROD_PLANVSOPS

Este procedimiento fue creado para generar un listado de las claves de los productos finales cuyas cantidades a producir están registradas en el plan de producción. De tal manera que sean importados para generar una nueva orden de producción. A continuación se muestra el código que lo integra.

```
CREATE PROCEDURE sia_prod_planvsops ( @pc_periodo CHAR(6), @pi_dia TINYINT, @pdt_fecha DATETIME ) AS

DECLARE @lc_cveart CHAR(15), @lc_artdescrip CHAR(70), @ldec_cantplaneada DECIMAL(11,3), @ldec_cantenops
DECIMAL(11,3), @li_mientras INTEGER

CREATE TABLE #temp_planvsops
(
  prod_cve_art       CHAR(15) NOT NULL,
  prod_artdescrip    CHAR(70) NOT NULL,
  prod_cant_plan     DECIMAL(11,3) DEFAULT 0 NOT NULL,
  prod_cant_enops    DECIMAL(11,3) DEFAULT 0 NOT NULL
)
```

```
DECLARE cur_prodplanvsops CURSOR FOR

  SELECT prod_pp_cve_prod_final,

  CASE

    WHEN @pi_dia = 1 THEN prod_pp_dia_01 WHEN @pi_dia = 2 THEN prod_pp_dia_02

    WHEN @pi_dia = 3 THEN prod_pp_dia_03 WHEN @pi_dia = 4 THEN prod_pp_dia_04

    WHEN @pi_dia = 5 THEN prod_pp_dia_05 WHEN @pi_dia = 6 THEN prod_pp_dia_06

    WHEN @pi_dia = 7 THEN prod_pp_dia_07 WHEN @pi_dia = 8 THEN prod_pp_dia_08

    WHEN @pi_dia = 9 THEN prod_pp_dia_09 WHEN @pi_dia = 10 THEN prod_pp_dia_10

    WHEN @pi_dia = 11 THEN prod_pp_dia_11 WHEN @pi_dia = 12 THEN prod_pp_dia_11

    WHEN @pi_dia = 13 THEN prod_pp_dia_13 WHEN @pi_dia = 14 THEN prod_pp_dia_14

    WHEN @pi_dia = 15 THEN prod_pp_dia_15 WHEN @pi_dia = 16 THEN prod_pp_dia_16

    WHEN @pi_dia = 17 THEN prod_pp_dia_17 WHEN @pi_dia = 18 THEN prod_pp_dia_18

    WHEN @pi_dia = 19 THEN prod_pp_dia_19 WHEN @pi_dia = 20 THEN prod_pp_dia_20

    WHEN @pi_dia = 21 THEN prod_pp_dia_21 WHEN @pi_dia = 22 THEN prod_pp_dia_22

    WHEN @pi_dia = 23 THEN prod_pp_dia_23 WHEN @pi_dia = 24 THEN prod_pp_dia_24

    WHEN @pi_dia = 25 THEN prod_pp_dia_25 WHEN @pi_dia = 26 THEN prod_pp_dia_26

    WHEN @pi_dia = 27 THEN prod_pp_dia_27 WHEN @pi_dia = 28 THEN prod_pp_dia_28

    WHEN @pi_dia = 29 THEN prod_pp_dia_29 WHEN @pi_dia = 30 THEN prod_pp_dia_30

    WHEN @pi_dia = 31 THEN prod_pp_dia_31 END

    FROM prod_plan_prod

    WHERE prod_pp_periodo = @pc_periodo
```

```
  FOR READ ONLY

SELECT @li_mientras = 0

OPEN cur_prodplanvsops

WHILE @li_mientras = 0

  BEGIN

    FETCH cur_prodplanvsops INTO @lc_cveart, @ldec_cantplaneada

    IF @@fetch_status = -1 BREAK

    SELECT @li_mientras = @@fetch_status

    IF @ldec_cantplaneada > 0

      BEGIN

        SELECT @ldec_cantenops = IsNull(sum(prod_op_cantidad), 0)

        FROM prod_orden_prod

        WHERE ( prod_op_origen_plan_prod = "S" )

        AND ( prod_op_id_estado NOT IN (4,7) )

        AND ( prod_op_cve_prod_final = @lc_cveart )

        AND (CONVERT(CHAR(10), prod_op_fecha_prod, 103)) = CONVERT(CHAR(10), @pdt_fecha, 103)



        SELECT @lc_artdescrip = art_descripcion

        FROM cat_articulos

        WHERE art_cve_art = @lc_cveart

        INSERT INTO #temp_planvsops

         ( prod_cve_art, prod_artdescrip, prod_cant_plan, prod_cant_enops )
```

```
        VALUES ( @lc_cveart, @lc_artdescrip, @ldec_cantplaneada, @ldec_cantenops )

      END

  END

CLOSE cur_prodplanvsops

DEALLOCATE cur_prodplanvsops

SELECT prod_cve_art, prod_artdescrip, prod_cant_plan, prod_cant_enops

FROM #temp_planvsops

WHERE prod_cant_plan <> prod_cant_enops

ORDER BY prod_cve_art

RETURN

;

IF EXISTS (SELECT  1 FROM sysobjects WHERE type = 'P' AND name = 'sia_prod_planvsops' )

            GRANT EXECUTE ON sia_prod_planvsops TO usu_sia ;
```

# IV.7 DESENCADENADOR (TRIGGER)

También conocido como disparador, su objetivo es mantener la integridad de los datos. Es en sí un procedimiento almacenado en la base de datos y se ejecuta automáticamente siempre que la información de una tabla específica cambie. Es un poderoso mecanismo para los administradores de bases de datos y para los desarrolladores para asegurar la confiabilidad de la información, pues evita que se hagan cambios no autorizados o inconsistentes.

Los desencadenadores no tienen parámetros o argumentos, y no se les puede invocar de manera explícita. Es decir, debe existir una modificación de datos para accionar un desencadenador. En cuanto a su desempeño, tienen una sobrecarga relativamente baja. La mayor parte del tiempo que emplea en su ejecución un desencadenador se utiliza en hacer referencia a otras tablas. Estas referencias pueden ser rápidas si las otras tablas están en memoria, o un poco más lentas si deben leerse desde el disco.

## IV.7.1 TABLAS DE INSERCIONES Y ELIMINACIONES

Los desencadenadores usan las tablas de inserciones y eliminaciones. Estas dos tablas contienen la misma estructura de la tabla base o "tabla del desencadenador", en donde se crea el desencadenador. Las tablas de inserciones y eliminaciones residen en RAM debido a que son tablas lógicas. Sí se agrega un nuevo registro a la tabla base, el registro se grabará tanto en la tabla base como en la tabla de inserciones. Tener disponibles los valores en las tabla de inserciones permite acceder a la información sin tener que crear variables que la contengan. Cuando se elimine un registro, éste se almacena en la tabla de eliminaciones. Una actualización es muy similar a una eliminación y una posterior inserción, si se actualiza un registro, el original se almacena en la tabla de eliminaciones y el registro modificado en la tabla base, en la tabla de inserciones.

## IV.7.2 PROD_TRIG_UPD_EXP_MATERIALES

Su objetivo es generar una copia idéntica de un componente cuando este es eliminado de la lista de materiales, de tal manera que, si así se desea, se tenga conocimiento de los componentes que alguna vez integraron la lista de materiales para un producto final específico. Además al eliminar un componente o insumo también realiza automáticamente una copia de su bitácora de cambios. A continuación se enuncia la estructura de este desencadenador.

```
DROP TRIGGER prod_trig_upd_exp_materiales ;

CREATE TRIGGER prod_trig_upd_exp_materiales ON prod_exp_materiales FOR UPDATE AS

DECLARE @lc_estatus CHAR(1), @ln_id NUMERIC(7,0), @ln_totregsmat NUMERIC(7,0), @ln_delregsmat NUMERIC(7,0)

IF UPDATE(prod_em_status)
```

```
BEGIN
  SELECT @lc_estatus = prod_em_status FROM inserted
  IF @lc_estatus = 'E'
    BEGIN
      INSERT INTO prod_bas_exp_materiales
      ( prod_bem_cve_padre, prod_bem_cve_hijo, prod_bem_tipo_material,
      prod_bem_opcional_de, prod_bem_cantidad, prod_bem_id_unidad,
      prod_bem_importancia, prod_bem_num_emp_alta,
      prod_bem_fec_alta, prod_bem_num_emp_bajamodif,
      prod_bem_fec_bajamodif, prod_bem_status, prod_bem_observaciones )
      SELECT inserted.prod_em_cve_padre, inserted.prod_em_cve_hijo,
      inserted.prod_em_tipo_material, inserted.prod_em_opcional_de,
      inserted.prod_em_cantidad, inserted.prod_em_id_unidad,
      inserted.prod_em_importancia, inserted.prod_em_num_emp_alta,
      inserted.prod_em_fec_alta, inserted.prod_em_num_emp_bajamodif,
      inserted.prod_em_fec_bajamodif, inserted.prod_em_status,
      inserted.prod_em_observaciones
      FROM inserted, deleted
      WHERE inserted.prod_em_cve_padre = deleted.prod_em_cve_padre
      AND inserted.prod_em_cve_hijo = deleted.prod_em_cve_hijo
      AND inserted.prod_em_opcional_de = deleted.prod_em_opcional_de

      SELECT @ln_totregsmat = @@rowcount

      IF @ln_totregsmat > 0
        INSERT INTO prod_bas_bit_cambio_materiales
        ( prod_bbc_id, prod_bbc_orig_id, prod_bbc_cve_padre, prod_bbc_cve_hijo,
        prod_bbc_cve_hijo_ant, prod_bbc_tipo_material_ant, prod_bbc_cantidad_ant,
```

```
prod_bbc_id_unidad_ant, prod_bbc_num_emp, prod_bbc_fec,
prod_bbc_observaciones )
SELECT @@identity, prod_bit_cambio_materiales.prod_bcc_id,
prod_bit_cambio_materiales.prod_bcc_cve_padre,
prod_bit_cambio_materiales.prod_bcc_cve_hijo,
prod_bit_cambio_materiales.prod_bcc_cve_hijo_ant,
prod_bit_cambio_materiales.prd_bcc_tipo_material_ant,
prod_bit_cambio_materiales.prod_bcc_cantidad_ant,
prod_bit_cambio_materiales.prod_bcc_id_unidad_ant,
prod_bit_cambio_materiales.prod_bcc_num_emp,
prod_bit_cambio_materiales.prod_bcc_fec,
prod_bit_cambio_materiales.prod_bcc_observaciones
FROM prod_bit_cambio_materiales, inserted, deleted
WHERE prod_bit_cambio_materiales.prod_bcc_cve_padre =inserted.prod_em_cve_padre
AND prod_bit_cambio_materiales.prod_bcc_cve_hijo = inserted.prod_em_cve_hijo
AND prod_bit_cambio_materiales.prod_bcc_cve_padre = deleted.prod_em_cve_padre
AND prod_bit_cambio_materiales.prod_bcc_cve_hijo = deleted.prod_em_cve_hijo

IF @@error = 0 AND @ln_totregsmat > 0
BEGIN
  DELETE prod_exp_materiales
  FROM prod_exp_materiales, inserted, deleted
  WHERE prod_exp_materiales.prod_em_cve_padre = inserted.prod_em_cve_padre
  AND prod_exp_materiales.prod_em_cve_hijo = inserted.prod_em_cve_hijo
  AND prod_exp_materiales.prod_em_opcional_de = inserted.prod_em_opcional_de
  AND inserted.prod_em_cve_padre = deleted.prod_em_cve_padre
  AND inserted.prod_em_cve_hijo = deleted.prod_em_cve_hijo
  AND inserted.prod_em_opcional_de = deleted.prod_em_opcional_de
```

```
        SELECT @ln_delregsmat = @@rowcount
        IF @ln_delregsmat > 0
          DELETE prod_bit_cambio_materiales
          FROM prod_bit_cambio_materiales, inserted, deleted
          WHERE prod_bit_cambio_materiales.prod_bcc_cve_padre =inserted.prod_em_cve_padre
          AND prod_bit_cambio_materiales.prod_bcc_cve_hijo = inserted.prod_em_cve_hijo
          AND prod_bit_cambio_materiales.prod_bcc_cve_padre =deleted.prod_em_cve_padre
          AND prod_bit_cambio_materiales.prod_bcc_cve_hijo = deleted.prod_em_cve_hijo
      END
    END
END
;
```

En esta etapa se han identificado las entidades primarias a partir de los requerimientos definidos en el tema anterior. Esto trae como consecuencia la generación física en la base de de datos de dichas entidades a través de tablas compuestas por columnas y renglones. Se ha constituido un Diagrama Entidad-Relación donde se plasma la relación y dependencia entre todas las tablas que almacenarán la información generada por el sistema de información. Además se han generado un par de archivos de instrucciones SQL para llevar a cabo operaciones directas en la base de datos.

Se optó por incluir algunos de los principales procesos contenidos en el código fuente del sistema, de tal manera que permita conocer y analizar su estructura y funcionalidad.

# *TEMA V: Implementación*

OBJETIVO: Enunciar los requerimientos mínimos para la instalación del Sistema, así como la serie de pasos de instalación para lograr la satisfactoria operación del mismo.

# V.1 REQUERIMIENTOS DE HARDWARE

Dado que un equipo particular de cómputo puede ser apropiado para una carga de trabajo e inadecuado para otro, es importante definir los requerimientos mínimos de tamaño y capacidad del mismo. Los puntos que deben tomarse en cuenta para una correcta elección son:

- Tamaño de Memoria Interna.
- Velocidad del ciclo del sistema para procesamiento.
- Número de Puertos USB, Paralelos y Serie para entradas, salidas y comunicaciones.
- Características de los dispositivos externos, además del teclado y ratón.
- Tipos y números de unidades de almacenamiento.

Además es importante, hacer una distinción entre un servidor de aplicaciones, un servidor de Bases de Datos y terminales de usuario, ya que, por obvias razones no será la misma carga de trabajo que tengan uno respecto al otro.

## V.1.1 MEMORIA INTERNA

La cantidad de memoria del servidor de aplicaciones y/o servidor de Base de Datos dependerá proporcionalmente del número de usuarios que invoquen o accedan a tales servidores; ya sea mediante el sistema, o través de transacciones directas a la información que estos almacenan. La cantidad mínima que permite la instalación del Manejador de Base de Datos y un desempeño aceptable del servidor en general, son 512 Mb. Ahora bien, sí se pretende programar procesos por lotes o simples tareas de respaldo y actualización de información de la base de datos, lo ideal resultará tener un 1 Gb para que estas operaciones no deriven en desbordamientos. En cambio, una terminal de trabajo o PC, requiere para la utilización del sistema, un mínimo de 128 Mb.

## V.1.2 VELOCIDAD DEL CICLO DEL SISTEMA PARA PROCESAMIENTO

Un servidor de aplicaciones y/o servidor de Base de Datos deberá tener como mínimo una velocidad de 1.8 Ghz. Mientras que una terminal de trabajo, con una velocidad de 660 Mhz contará con un seguro desempeño para el acceso remoto a realizar.

### V.1.3 NÚMERO DE PUERTOS USB, PARALELOS Y SERIE PARA ENTRADAS, SALIDAS Y COMUNICACIONES

El servidor de aplicaciones deberá disponer de un puerto USB adicional a los ya definidos para teclado y ratón, útil para impresoras o unidades externas de almacenamiento secundario.

### V.1.4 CARACTERÍSTICAS DE LOS DISPOSITIVOS EXTERNOS, ADEMÁS DEL TECLADO Y RATÓN

No es indispensable contar con teclados y ratones muy sofisticados, siempre cuando, en el caso de los teclados, estos cumplan con el estándar de 102 teclas en inglés o español. Mientras que, la calidad y tipo de impresoras y multifuncionales no es dato que influya en la operación del sistema.

### V.1.5 TIPOS Y NÚMEROS DE UNIDADES DE ALMACENAMIENTO

Además de contar con un disco duro de 80 Gb como mínimo para el Servidor y de 100 Mb como mínimo para una terminal de trabajo, idealmente el servidor deberá contar con una disposición de discos duros adicionales para trabajar como espejos del disco duro principal, y así recurrir a la información almacenada en estos, en caso de contingencia o problemas que se presenten en el disco duro principal.

# V.2 REQUERIMIENTOS DE SOFTWARE

Son dos los elementos de Software primordiales para llevar a cabo la instalación y operaciones del sistema:

- Sistema Operativo.
- Manejador de Bases de Datos.

## V.2.1 SISTEMA OPERATIVO

Este sistema sólo puede operar bajo ambiente Windows, por lo que para otros sistemas operativos deberá utilizarse un emulador para el Microsoft Windows ya que este sistema de información requiere la utilización de API's. En cuanto a su versión, para terminales de usuario se requiere como mínimo Windows 98, en cambio para el servidor de aplicaciones y/o base de datos es requisito que tenga instalado al menos Windows NT Server o preferentemente, Windows 2000 Server.

## V.2.2 MANEJADOR DE BASES DE DATOS

Idealmente debe ser SQL-Server, sin embargo, para SYBASE deberán hacerse las adecuaciones necesarias de acuerdo a la sintaxis y estándares definidos por este último manejador, de tal manera que cualquier proceso puede ser satisfactoriamente interpretado y con ello se realice correctamente cada una de las operaciones definidas o programadas. En cambio, para un manejador como ORACLE, las adecuaciones necesarias a implementar son más complejas, lo que implicará demasiadas horas-hombre ya que, tan sólo para la definición de las Tablas de Bases de Datos de este sistema, específicamente en la declaración de los tipos de dato, varía considerablemente respecto a los dos manejadores mencionados anteriormente.

## V.3 GUÍA DE INSTALACIÓN DEL SISTEMA

1. Se debe crear un directorio para el sistema, que por ejemplo puede llamarse PRODUCCIÓN (este nombre puede cambiarse).

2. Determinar la ubicación de las librerías del sistema en el servidor de aplicaciones. Las librerías y el archivo ejecutable del sistema son los siguientes:

- pfcapp.pbd.
- pfcapsrv.pbd.
- pfcdwsrv.pbd.
- pfcmain.pbd.
- pfcutil.pbd.
- pfcwnsrv.pbd.
- pfeapsrv.pbd.
- pfedwsrv.pbd.
- pfemain.pbd.
- pfeutil.pbd.
- pfewnsrv.pbd.
- sia_prod1.pbd.
- sia_prod2.pbd.
- sia_prod3.pbd.
- sia_prod.pbr.
- pfcwnsrv.pbr.
- pfcutil.pbr.
- pfcmain.pbr.
- pfcdwsrv.pbr.

- pfcapsrv.pbr.
- sia_prod.exe.
- sia_settings.ini.

3. Además deben copiarse las siguientes imágenes en el directorio (o puede hacer un directorio de bmp's y jpg's):

- cross.gif.
- principal1.gif.
- review.gif.
- pb_bitacora.jpg.
- pb_bitacora2.jpg.
- ar_down.bmp.
- ar_right.bmp.
- ar_up.bmp.
- folder.bmp.
- folderopen.bmp.

4. El Sistema requiere de los archivos de conexión del Openclient (en caso de SYBASE) o el Client Network Utility (en caso de SQL-Server), para establecer las conexiones necesarias al Servidor de la Base de Datos. Una vez que se disponga de esta utilería, deberán configurarse, para definir la dirección ip y el nombre del servidor de la Base de Datos.

5. Una vez que mediante la ejecución del paso anterior se han definido los datos principales del servidor de base de datos, estos deben establecer se en el archivo sia_settings.ini para la conexión del sistema a la base de datos. Tal y como se muestra en la figura V.1.


**Figura V.1.- Parámetros para la Conexión a la Base de Datos en el archivo sia_settings.ini**

6. Para que un sistema desarrollado en PowerBuilder 8.0 funcione correctamente, se debe incluir las siguientes librerías propias de esta herramienta de desarrollo:

- PBVM80.DLL. De propósito general.
- LIBJCC.DLL. De propósito general.
- PBDWE80.DLL. Para el funcionamiento de DataWindows y DataStores.
- PBRTC80.DLL. Para el funcionamiento de Objetos de Líneas de Edición.
- PBFNT80.INI. Para el reconocimiento de las fuentes de texto, disponible.
- PBLAB80.INI. Para el funcionamiento de los distintos formatos predefinidos de DataWindows.
- PBTRA80.DLL. Para la conexión a la Base de Datos.

7. Reconstruir los objetos de Bases de Datos enunciados en el archivo crebas.sql, a través de la herramienta de Software Query-Analyzer.

# V.4 PRUEBAS

Dado que el análisis y desarrollo de este sistema parte más de una investigación que de una lista de requerimientos de un cliente específico o de una empresa de la industria manufacturera, se han establecido y cubierto un conjunto de etapas de simulación y pruebas que confirman el correcto funcionamiento del sistema que sustenta esta tesis. Las cuales consistieron en:

- Acceso validado al sistema y a la Base de Datos.
- Monitoreo de Bloqueos en la Base de Datos como consecuencia de múltiples sesiones.
- Integración de información en todos los Catálogos.
- Administración de Listas de Materiales.
- Administración de Plan Mensual de Producción.
- Generación de Ordenes de Producción.
- Monitoreo de Avance de Ordenes de Producción.
- Generación de Reportes.

Para cuestiones de presentación y pruebas del sistema se dispone de una copia de una base de datos de producción, la cual contiene toda la estructura de objetos requeridos para su funcionalidad.

No se han incluido pantallas o imágenes del sistema, pues se ha optado por hacer una presentación en tiempo real, donde podrá apreciarse su alcance y veracidad.

# V.5 LIBERACIÓN

El sistema que sustenta este trabajo de tesis no fue desarrollado en base a requerimientos de una empresa o cliente específico, de ahí que no tenga una liberación inmediata para el ramo al que va dirigido. Sin embargo la información de la que dispone y a su vez genera en una base de datos, está bien fundamentada en datos reales como consecuencia de que la empresa Desarrollos de Tecnología de Información, S.A. de C.V., la cual prestó su infraestructura (entiéndase por ello servidores de bases de datos, servidores de aplicaciones, red de computadoras, inclusive bibliografía y asesoramiento) para la creación del mismo, cuenta con un conjunto de bases de datos de producción para la implantación, pruebas y liberación final de los sistemas de información que desarrolla y a los cuales también proporciona un mantenimiento periódico. Por lo que, el control de Procesos de Producción que pretende llevarse a cabo mediante la utilización de este sistema, no es fortuito ni empírico. Aunado a ello, está basado en una investigación de campo y documental, la cual si bien no fue muy extensa, si proporcionó la información suficiente.

Además, dado que existió una continúa asesoría y una gran aportación en ideas por parte del área administrativa de la empresa Desarrollos de Tecnología de Información, S.A. de C.V., en la medida de que éste sistema le sea útil a la misma, podrán disponer de él, de ahí que el visto bueno que hasta hoy día ésta le ha dado al sistema, es una manera de certificar que se va por buen camino, y tan sólo requerirá de un mantenimiento para integrarlo al conjunto de módulos que integra su suite de administración empresarial, si dicha empresa así lo considera.

# V.6 ESTRUCTURA Y PRINCIPALES OPCIONES DEL SISTEMA

A continuación se muestran imágenes del menú y de las principales opciones del sistema.

## V.6.1 MENÚS

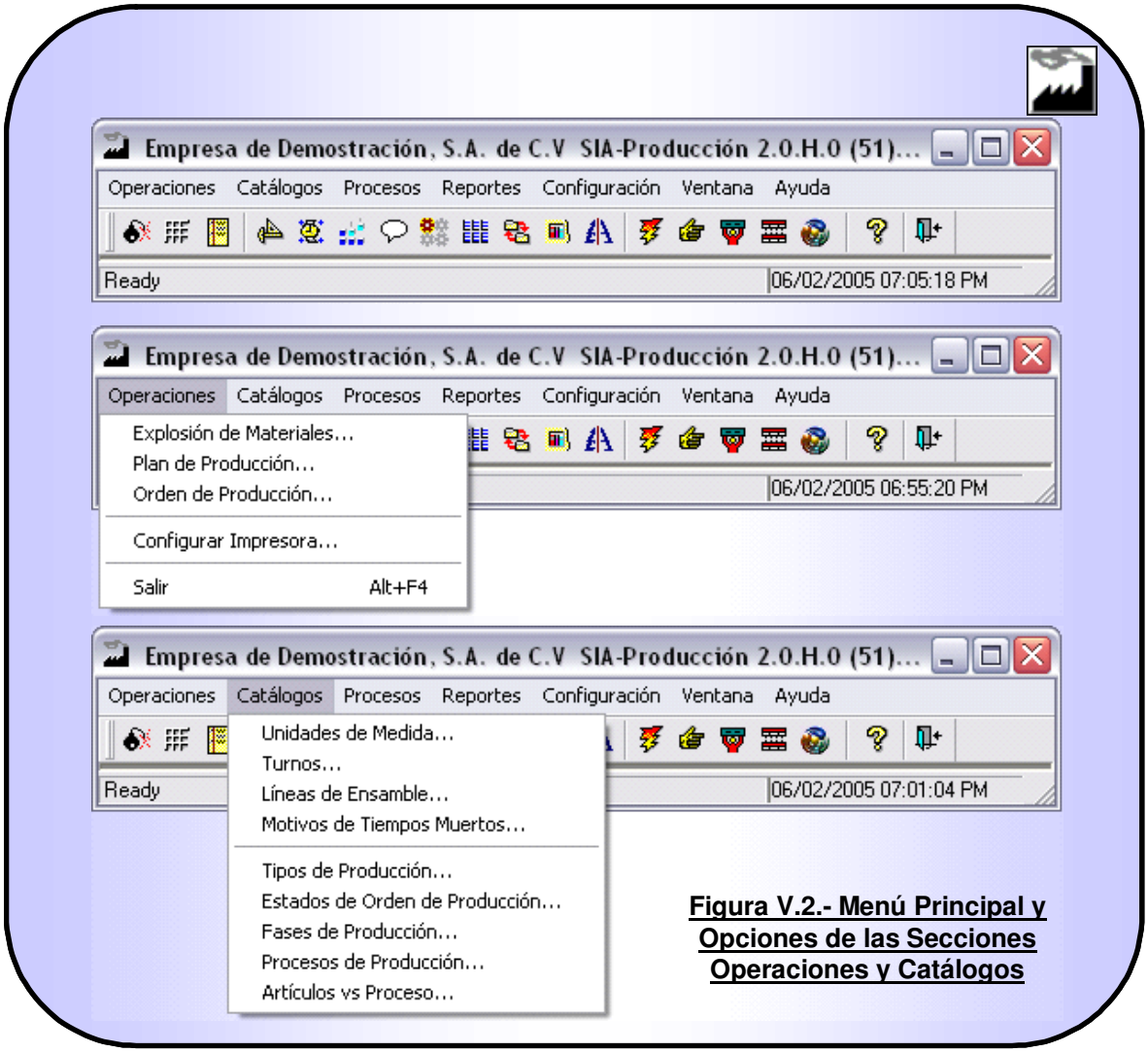



**Figura V.2.- Menú Principal y Opciones de las Secciones Operaciones y Catálogos**


**Figura V.3.- Opciones de las Secciones Procesos, Reportes y Configuración**

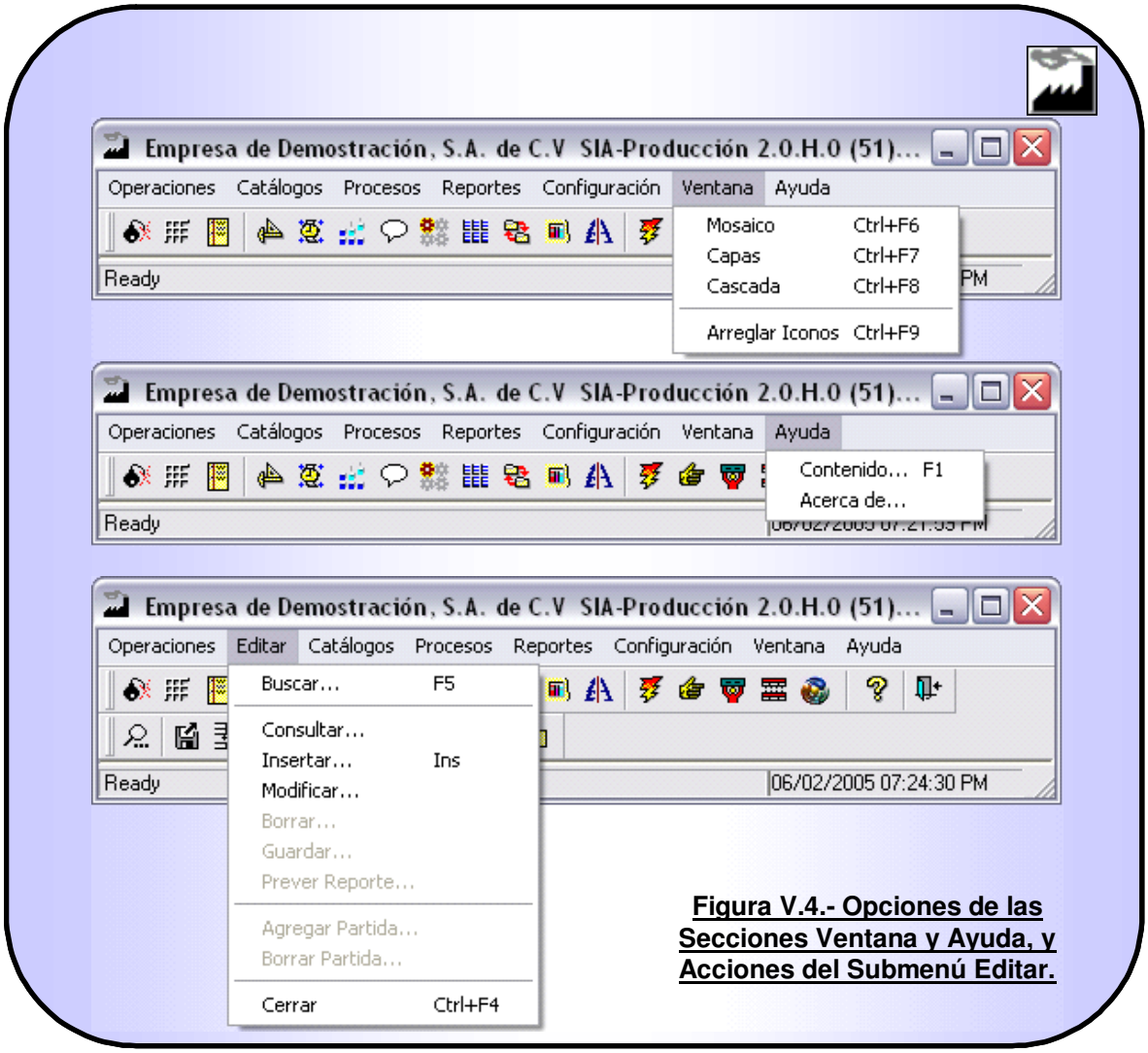



**Figura V.4.- Opciones de las Secciones Ventana y Ayuda, y Acciones del Submenú Editar.**

## V.6.2 CATÁLOGOS

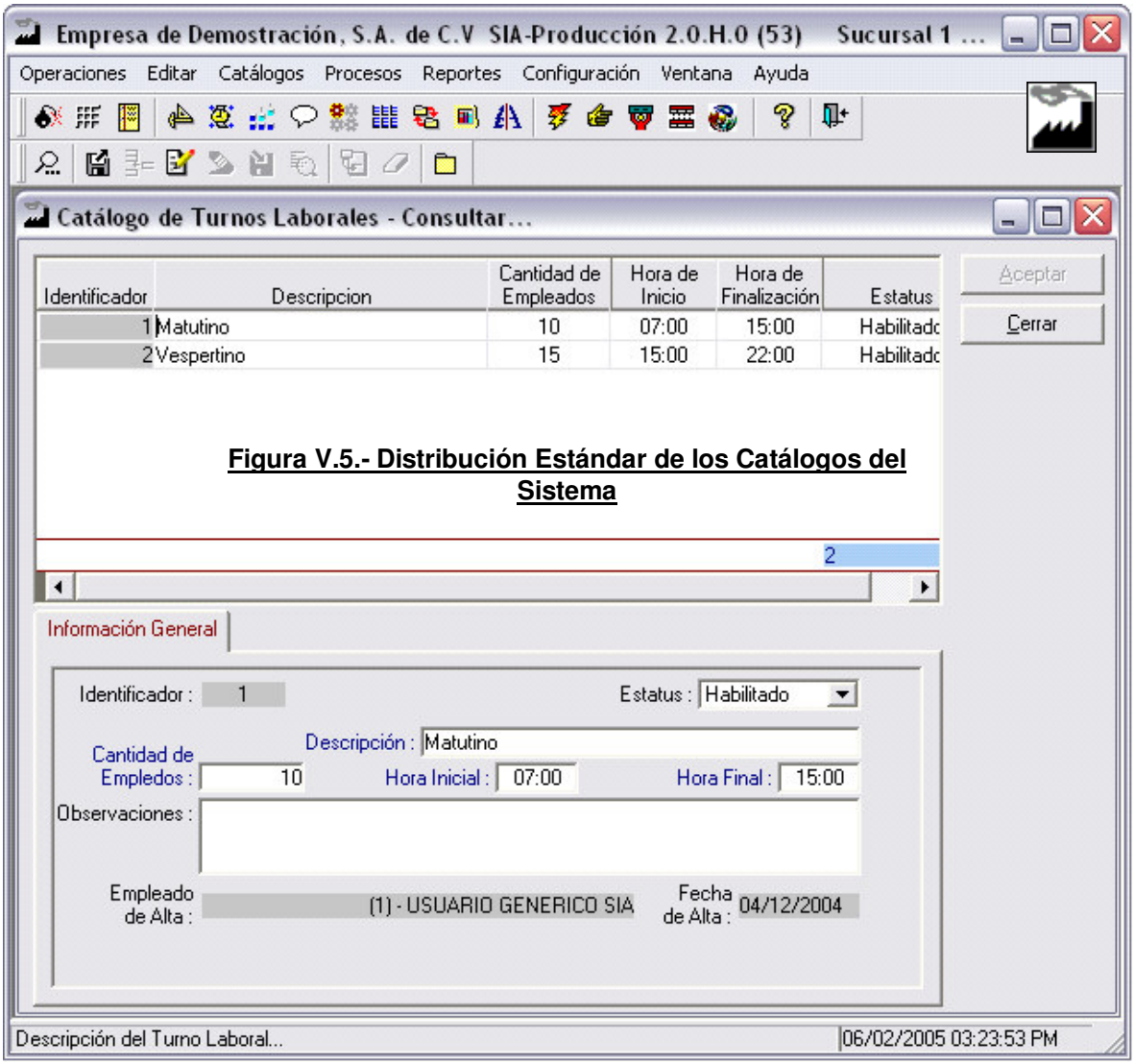



**Figura V.5.- Distribución Estándar de los Catálogos del Sistema**

## V.6.3 BÚSQUEDAS

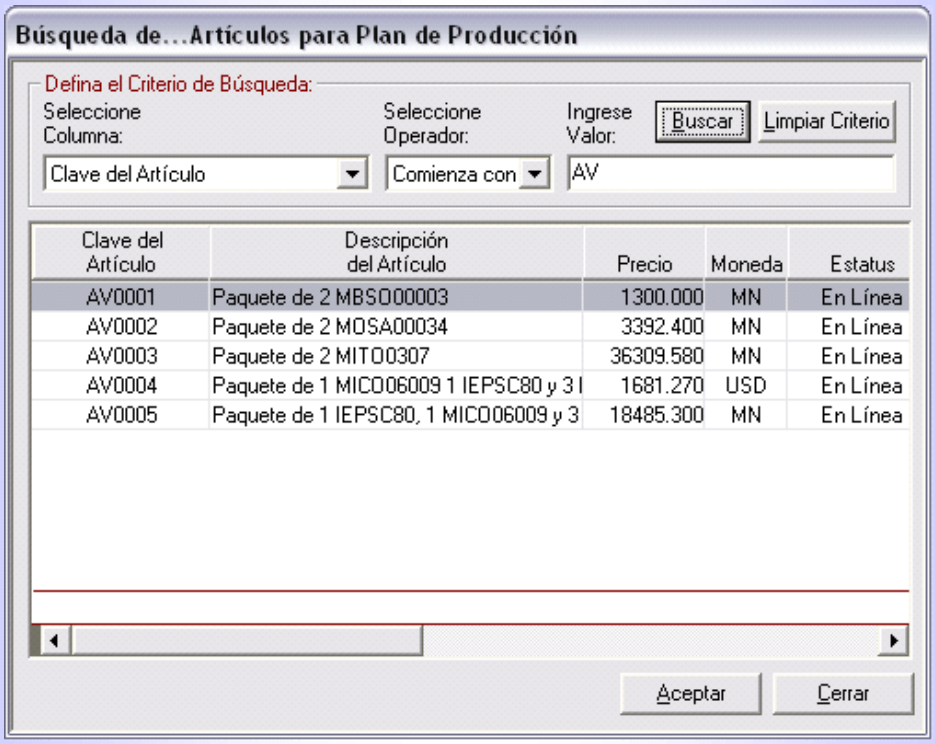



**Figura V.6.- Distribución Estándar de Búsquedas**

## V.6.4 OPERACIONES

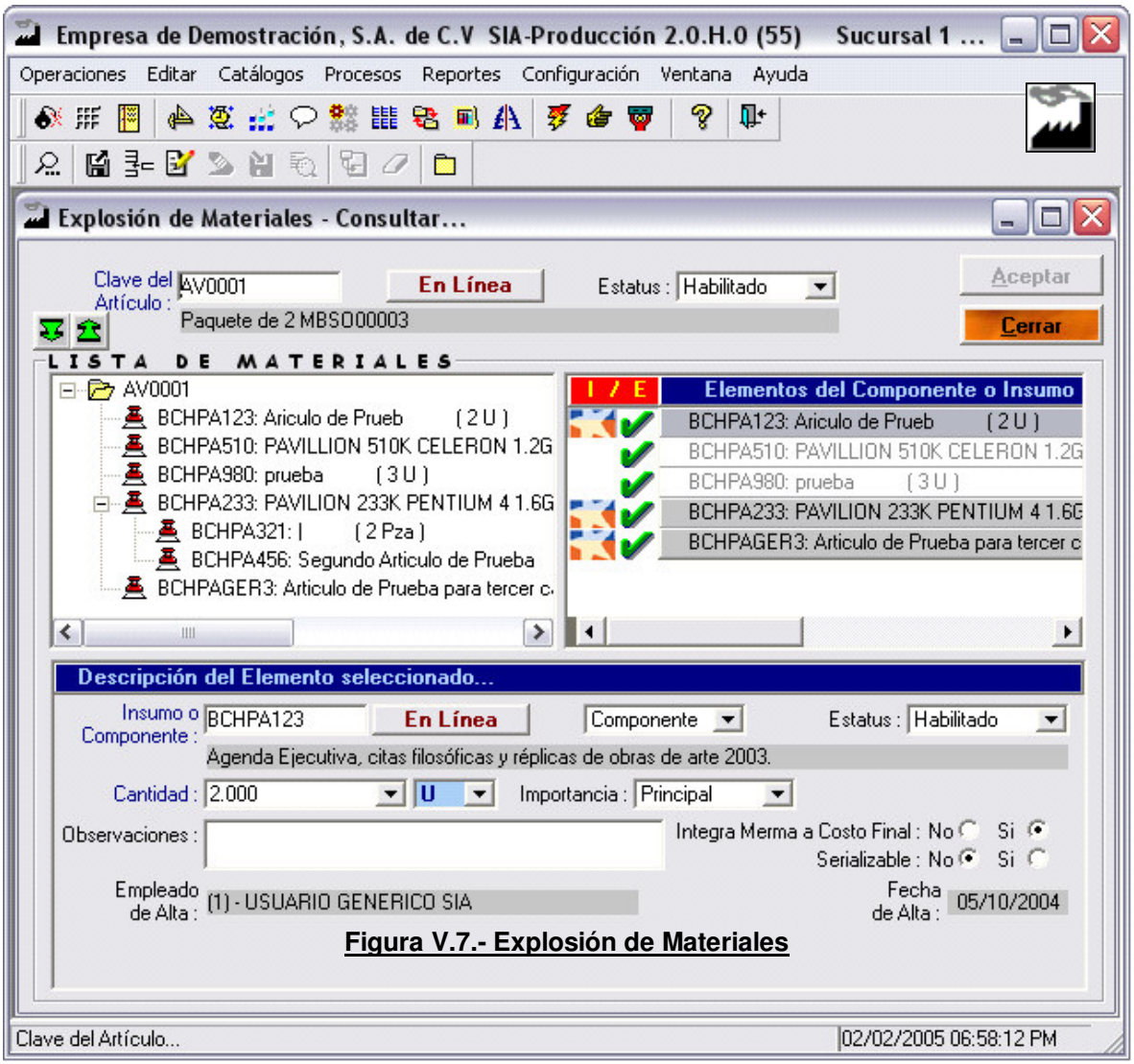



**Figura V.7.- Explosión de Materiales**

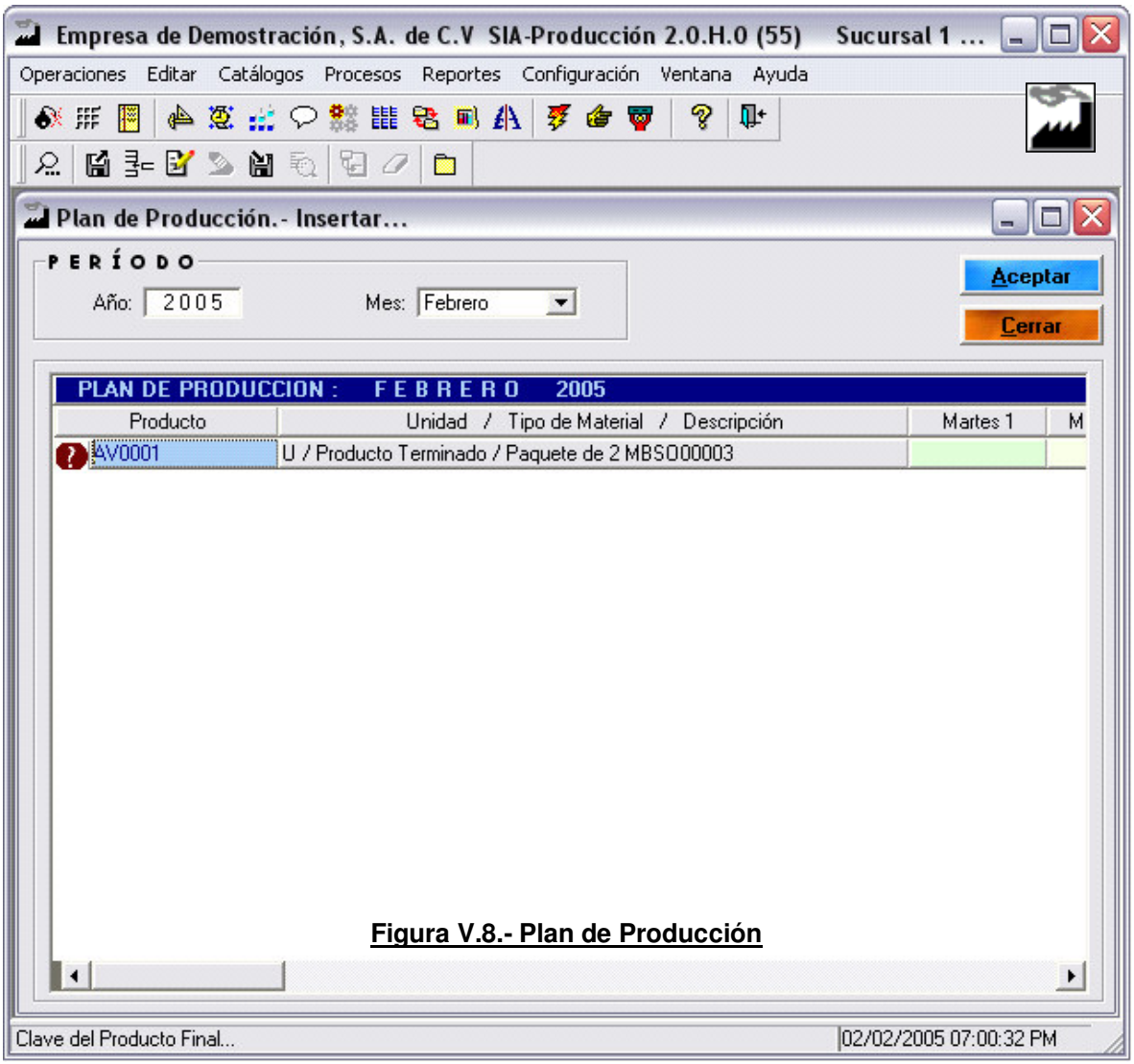



**Figura V.8.- Plan de Producción**

Empresa de Demostración, S.A. de C.V SIA-Producción 2.0.H.0 (55) Sucursal 1 ...

Operaciones Editar Catálogos Procesos Reportes Configuración Ventana Ayuda

Orden de Producción.- Consultar...

Información General...

Folio : 2 Fecha de Producción : Estatus : EN PRODUCCION

Turno : Matutino (1) Línea de Ensamble : Principal (1) Tipo de Producción : Artesanal (Ar)

Producto Final : AV0001 Cantidad a Producir : 3 U

Paquete de 2 MBSO00003

Costo Inicial : 2243.990 MN Costo Final : 2243.990 M

Proceso de Producción : Ensamble de Computadoras Básicas. (1) Lote :

Observaciones : Prueba de Fredy Integra Merma a Costo Final : No Si

Serializable : No Si

Empleado de Alta : (1) - USUARIO GENERICO SIA Fecha de Alta : 22/12/200

**Figura V.9.- Orden de Producción**

Lista de Materiales...

| Fabricar o Ensamblar | Clave Componente | Costo Unitario | Cantidad Requerida | Cantidad Máxima Permitida | |
|---|---|---|---|---|---|
| ☐ | BCHPA123 | .000 | 2 | 6 | U |
| ☐ | BCHPA510 | 7,558.000 | 1 | 3 | U |
| ☐ | BCHPA980 | .000 | 3 | 9 | U |
| ☑ | BCHPA233 | 132.394 | 3 | 3 | U |
| ☐ | BCHPA321 | .000 | 2 | 6 | Pza |

Folio de la Orden de Producción... 02/02/2005 07:03:10 PM

## V.6.5 PROCESOS

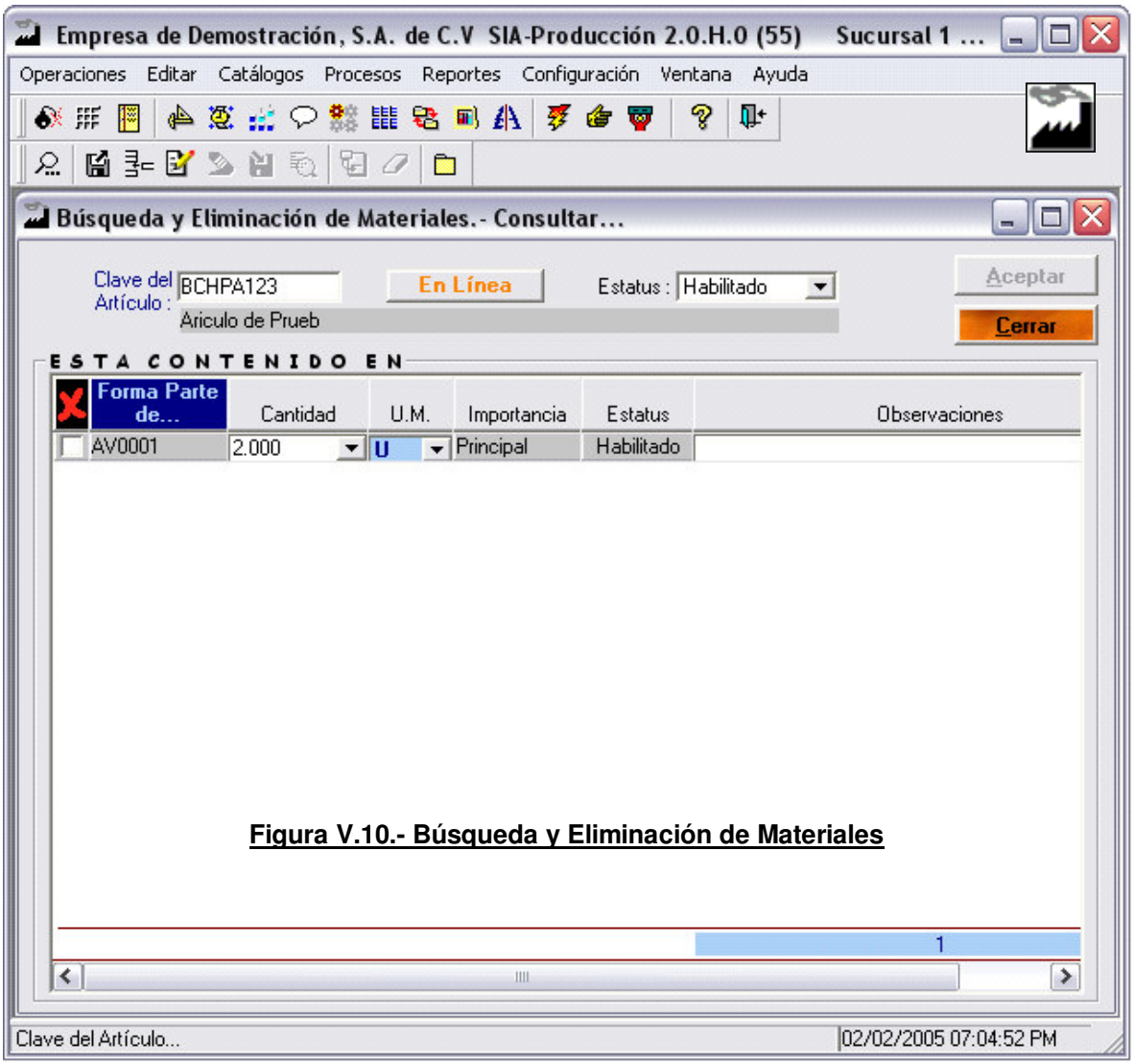



**Figura V.10.- Búsqueda y Eliminación de Materiales**

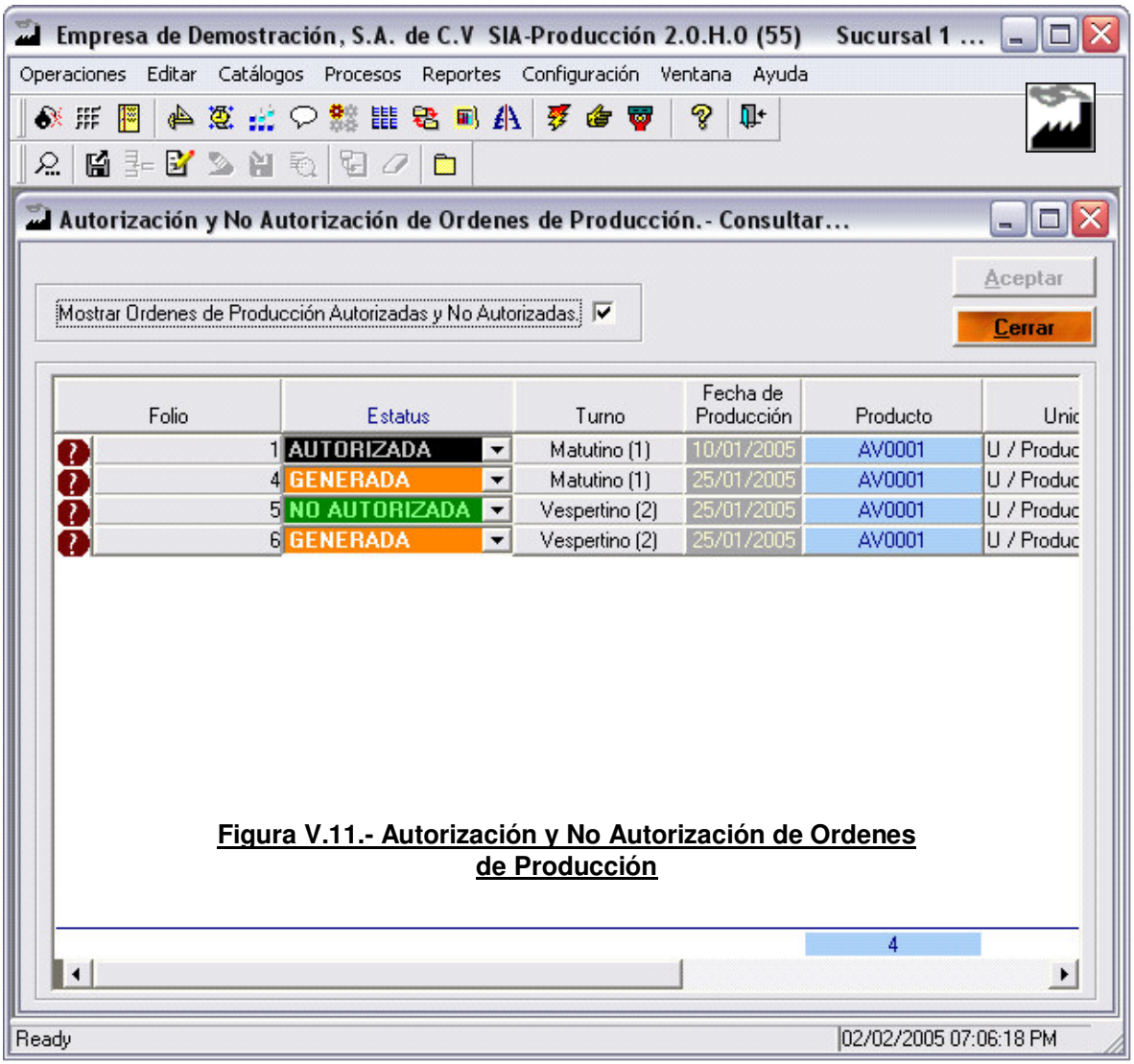



**Figura V.11.- Autorización y No Autorización de Ordenes de Producción**

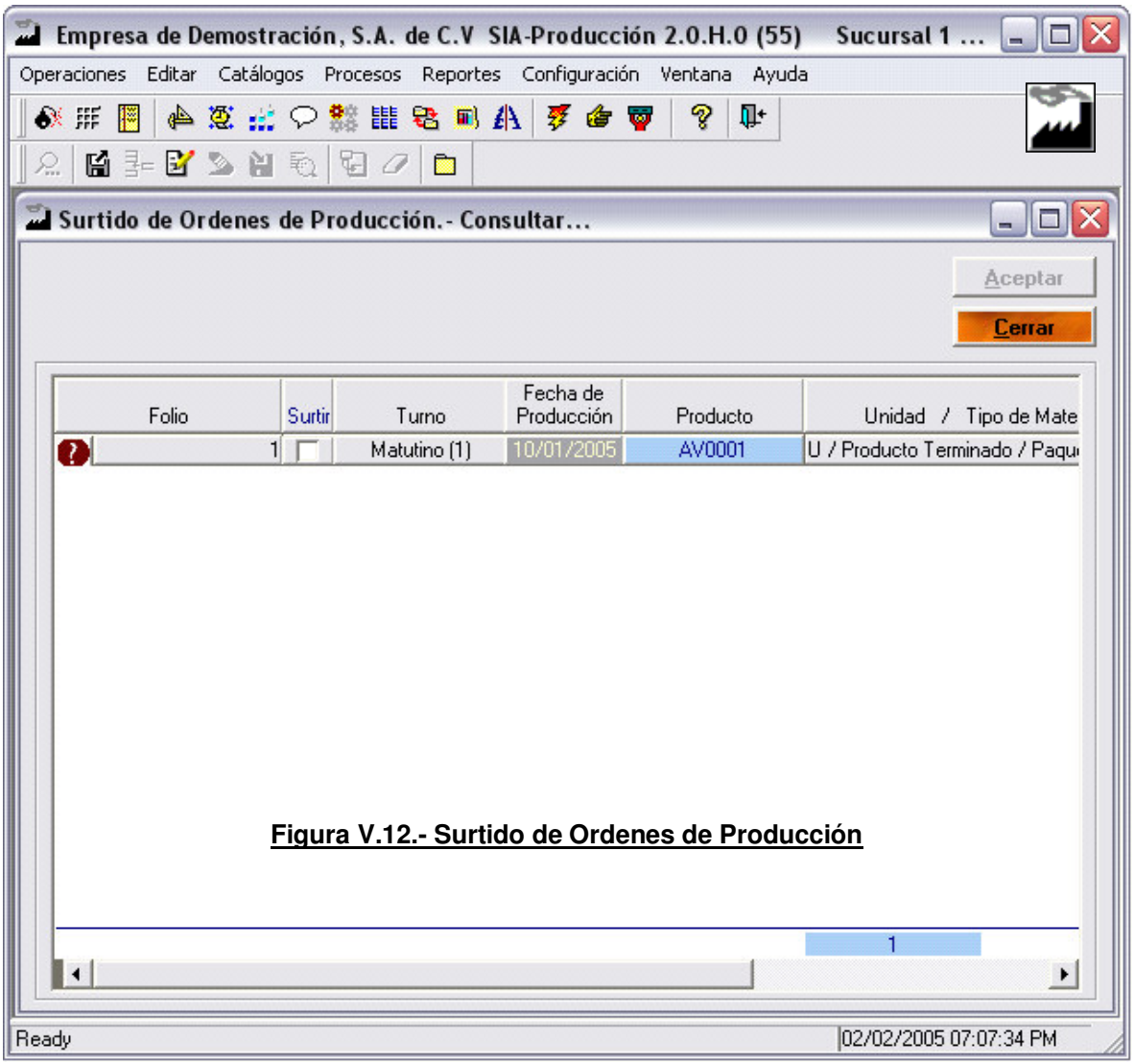



**Figura V.12.- Surtido de Ordenes de Producción**

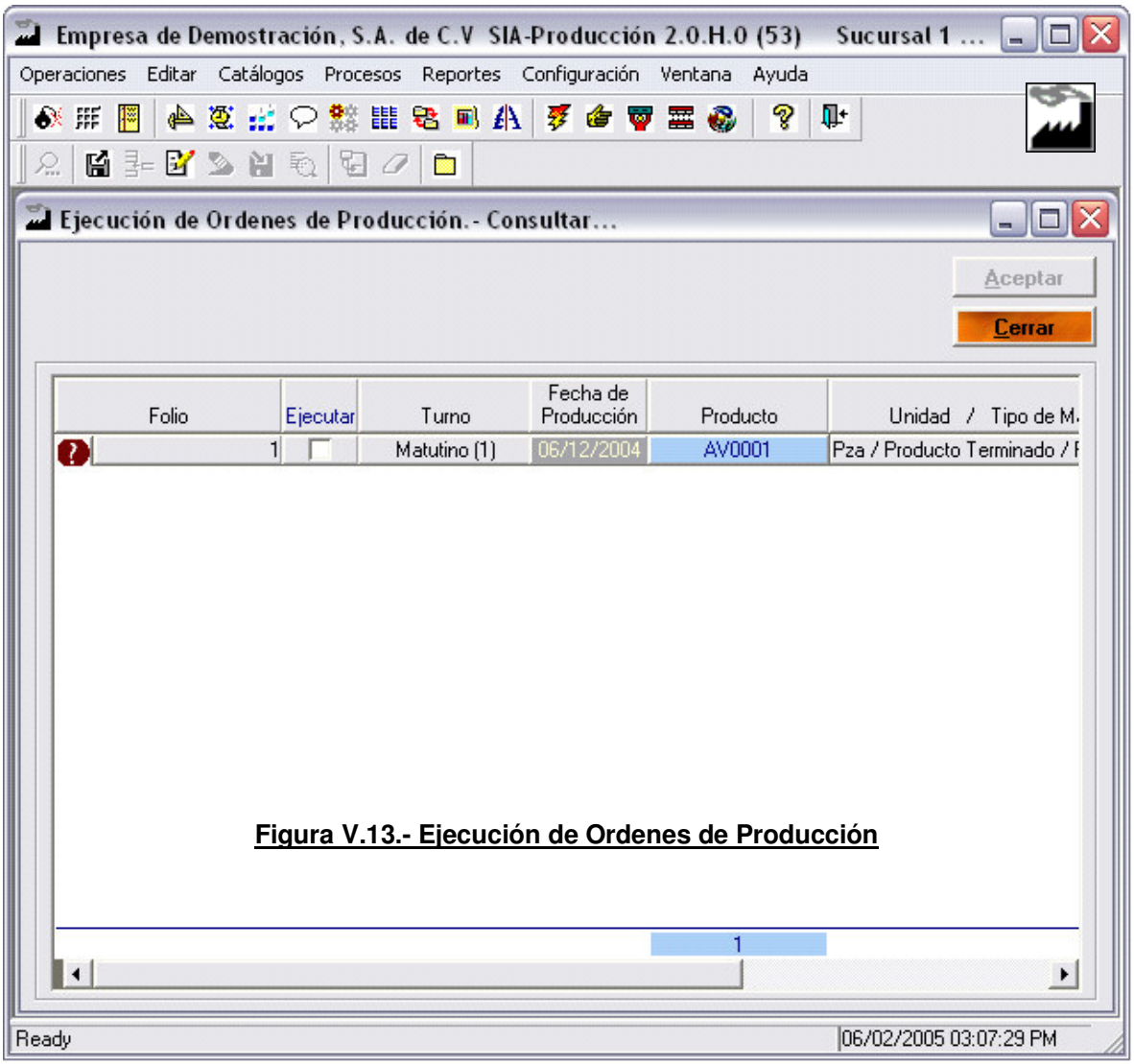



**Figura V.13.- Ejecución de Ordenes de Producción**

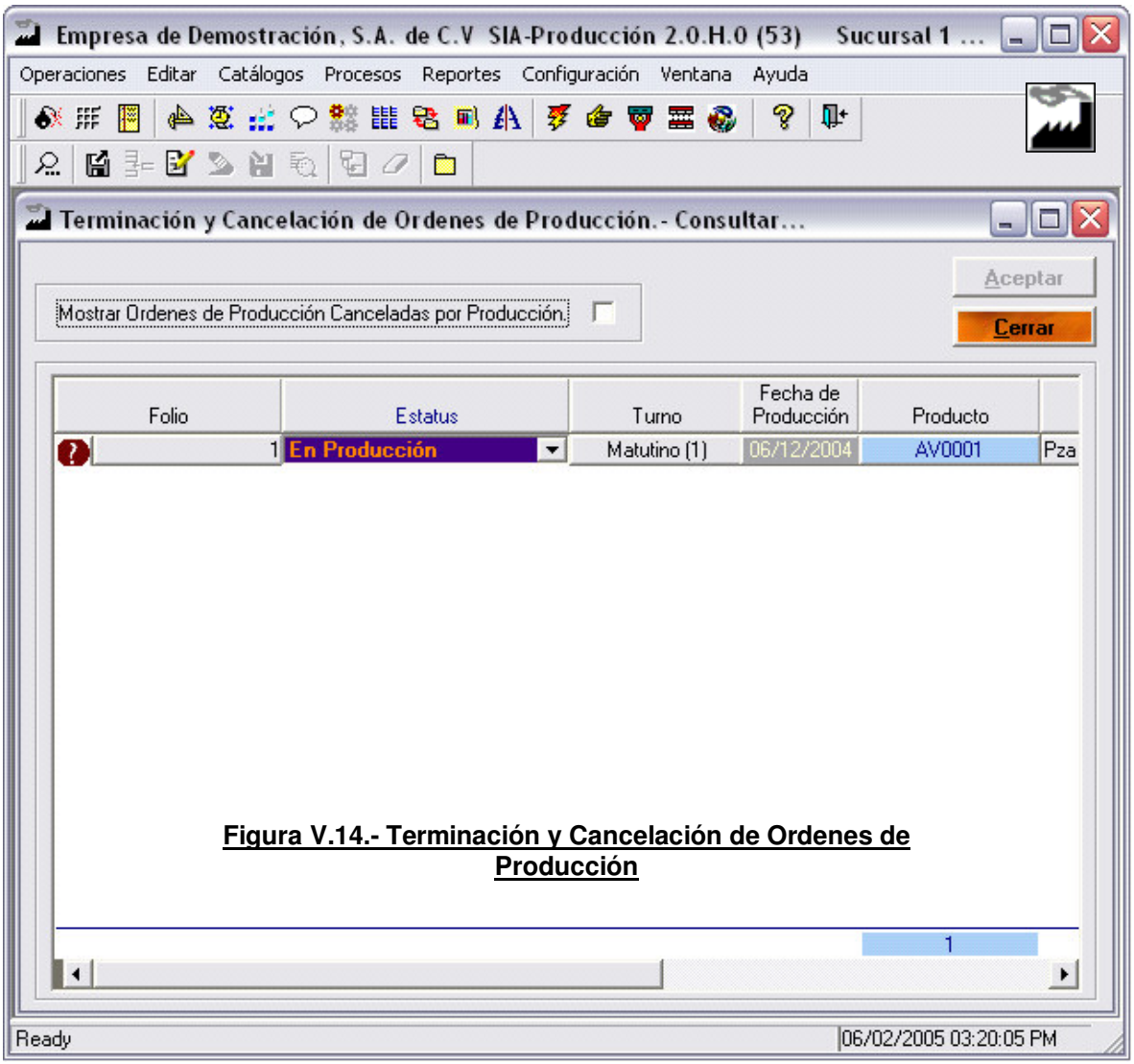



**Figura V.14.- Terminación y Cancelación de Ordenes de Producción**

## V.6.6 PARÁMETROS PARA GENERAR REPORTES

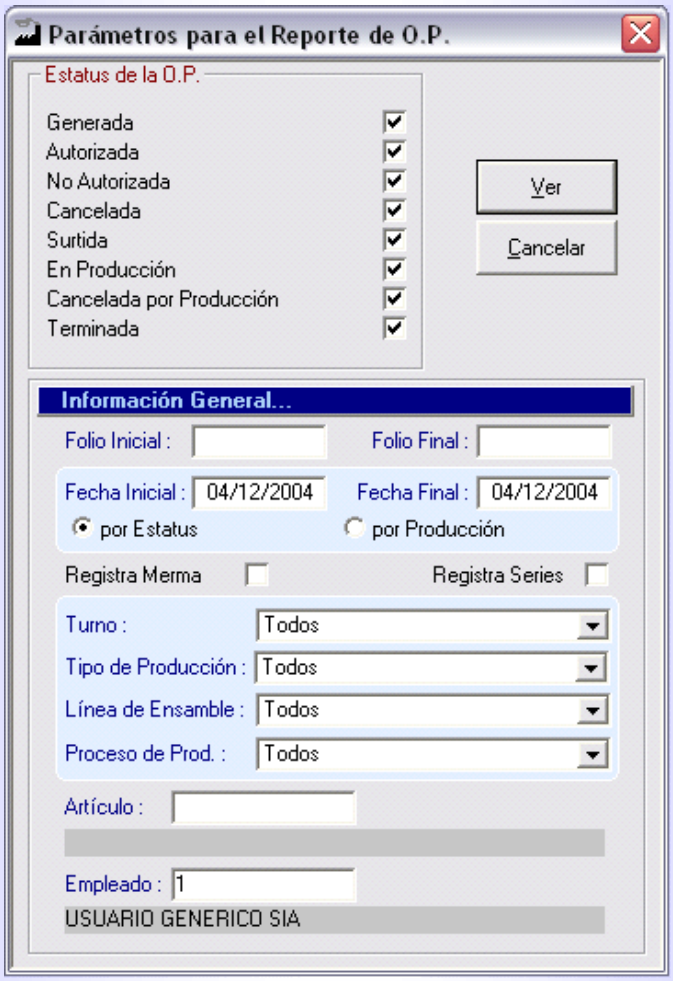



**Figura V.15.- Parámetros que generan los Reportes del Sistema**

# *Conclusiones*

Todos los sistemas tienen sus limitaciones, dependiendo de para qué se han diseñado, para el caso que trata este trabajo de tesis, está orientado totalmente a cualquier área de producción. Tal especialización no ha sido siquiera incipientemente explotada por el sector informático en nuestro país, pues son contadas las empresas de TI que ofrecen a sus clientes una aplicación informática de esta naturaleza. De ahí que resultó interesante abordar un tema de ésta índole.

A nuestra universidad se le proporciona un trabajo de tesis y un sistema que por el simple hecho de ser novedosos por el tema que tratan, tiene una gran valía; no obstante para cuestiones de investigación, resultará una bibliografía harto interesante por la secuencia precisa y finita que conllevó es trabajo, pues si bien, su intención no es didáctica, si fue importante el enunciar, definir y en algunos explicar los conceptos que día a día se iban utilizando y desarrollando. Como consecuencia la intención siempre fue el cubrir los objetivos inicialmente establecidos, además de explicar todos y cada uno de los conceptos, métodos y herramientas que conllevaron al desarrollo del sistema de información; esto con la intención de primero enunciar y después proceder a la utilización o aplicación del término en cuestión.

Referente al sistema, si bien es cierto que aún no es un producto muy robusto, si abarca las partes fundamentales de cualquier proceso de producción, es decir, la Explosión de Materiales y la Orden de Producción, pues cualquier otro sistema cuya razón de existir sea la misma y contenga los elementos antes mencionados igualmente logrará su propósito.

Pensando a futuro, respecto al monitoreo de las fases del proceso productivo, se podrá robustecer mediante la tecnología de los PDA, ya que bien podría crearse el mismo programa que realiza esta tarea en el sistema, a través de una aplicación Pocket PowerBuilder, que implicaría una conexión remota e inalámbrica al servidor de la base de datos, con lo cual se ahorraría en equipos completos de cómputo. Además podría desarrollarse una interfaz para el surtido de las ordenes de producción a través del Almacén de Materia Prima, inclusive explotar la información actualmente generada para obtener reportes más robustos o analíticos.

Este sistema puede crecer demasiado operacional hablando, más sin embargo, por el momento cumple su principal encomienda. El mantenimiento que este requerirá a través del tiempo dependerá directamente de las numerosas puestas de producción del mismo en igual número de empresas del sector manufacturero.

Gracias.

# *Bibliografía*

BERSON, Alex y Anderson George (1997). **Sybase and Client/Server Computing: Featuring System II**. 2ª. Edición. Estados Unidos: Edit. McGraw-Hill Companies, Inc.

BLAS, Clemente de (1991). **PC: Guía del Usuario**. México: Macrobit Editores, S.A. de C.V.

CUSUMANO, Michael A. y Selby Richard W. (1996). **El Secreto de Microsoft**. México: Edit. Prentice Hall Hispanoamericana, S.A.

CHANDAK, Ramesh y Chandak Purshottam (1999). **Advanced PowerBuilder 7**. Estados Unidos: Edit. John Wiley & Sons, Inc.

FAIRLEY, Richard E. (1988). **Metodología de la Investigación**. México: McGraw-Hill / Interamericana de México, S.A. de C.V.

HERNÁNDEZ, Sampieri Roberto y otros (1998). **Metodología de la Investigación**. México: McGraw-Hill Interamericana Editores, S.A. de C.V.

HEYS, Williams B. (1998). **Edición Especial PowerBuilder 6**. España: Edit. Prentice Hall Iberia, S.R.L.

HOPEMAN, Richard J. (1986). **Administración de Producción y Operaciones: Planeación, Análisis y Control**; México: Compañía Editorial Continental, S.A. de C.V.

LARMAN, Craig (1999). **UML y Patrones: Introducción al Análisis y Diseño Orientado a Objetos**. México: Edit. Prentice Hall Hispanoamericana, S.A.

LÓPEZ, Nathalie y otros (1998). **Integrar UML en los Proyectos**. España: Edit. Ediciones Gestión 2000, S.A.

LÓPEZ-FUENSALIDA, Antonio (1991). **Metodologías de Desarrollo: Producción Automática de Software con Herramientas CASE**. México: Macrobit Editores, S.A. de C.V.

MAHLER, Paul (1996), **PowerBuilder: Desarrollo de Aplicaciones Cliente - Servidor**. España: Edit. Prentice Hall Hispanoamericana, S.A.

MONDEN, Yasuhiro (), **El Sistema de Producción de Toyota: Just In Time**. Argentina: Ediciones Macchi.

MURDICK Robert G. Y Munson John C. (1988), **Sistemas de Información Administriva**. México: Edit. Prentice Hall Hispanoamericana, S.A.

RIGGS, James L. (1976). **Sistemas de Producción: Planeación, Análisis y Control**. México: Edit. Limusa, S.A. de C.V.

SCHMULLER, Joseph (2000). **Aprendiendo UML en 24 horas**. México: Edit. Prentice Hall Hispanoamericana, S.A.

SENN, James A. (1988). **Análisis y Diseño de Sistemas de Información**. México: Edit. McGraw-Hill / Interamericana de México, S.A. de C.V.

VASQUEZ, Martínez Heliodoro (1992). **Productividad y Seguridad en el Trabajo: Problema actual de la Industria**. México: Edit. Diana, S.A. de C.V.

WAYMIRE, Richar y Sawtell Rick (2000). **Aprendiendo Microsoft SQL Server 7 en 21 días**. México: Edit. Prentice Hall Hispanoamericana, S.A.