12

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UN SISTEMA PARA LA VERIFICACIÓN DE FACTIBILIDAD DE MANUFACTURA DE UN PRODUCTO

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERA EN COMPUTACIÓN

P R E S E N T A N :
GABRIELA ÁVILA ZARAZÚA
MARÍA SOFÍA MARTÍNEZ ARAUJO
LIZBETH VILLARRUEL FLORES

DIRECTOR DE TESIS:
M.I. ÁLVARO AYALA RUÍZ

CIUDAD UNIVERSITARIA **2002.**

PAGINACIÓN DISCONTINUA

Dedicada a mis padres Sofía y Gerardo; quienes han sido pacientes a lo largo de toda mi carrera, especialmente a Mamá por todo su apoyo y ayuda, sobre todo en los momentos difíciles, la cual ha sido y seguirá siendo mi modelo a seguir. Ustedes significan mucho para mí y saben que los quiero de igual forma.

A mis hermanas que han compartido conmigo toda una vida de alegrías y tristezas pero que siempre hemos logrado salir adelante juntas. A Graciela con sus ocurrencias, a Isabel con sus regaños, pero más que eso, yo sé que eran consejos para seguir luchando.

A Eduardo Hernández por su comprensión, apoyo, cariño y sonrisas durante todo el tiempo que hemos compartido juntos. Un gran amigo, más que un hermano, la persona más especial en este momento de mi vida. Gracias por participar en mis logros, "mi estrellita de la suerte". Te quiero mucho.

A Patricia Zempoalteca que contribuyó en mucho para la realización de la tesis con sus ejemplos, le agradezco haberme enseñado algunos trucos y por brindarme su amistad.

A mi asesor Álvaro Ayala por haberme permitido ser parte de este proyecto.

A mis verdaderos amigos y a todo aquel que un día me dijo sinceramente: "cuenta conmigo, estoy de tu lado". Especialmente a Salvador García por sus palabras de aliento y consejos que siempre tendré presentes.

María Sofía Martínez Araujo

Agradezco a Dios por permitirme terminar mi carrera,, por darme a mis padres que son pilar de mi vida, GRACIAS.

Gracias a mis Padres Rafael y Blanca,, por apoyarme y no dejarme sola en los momentos difíciles, por ser mi motor para salir adelante y lograr todo lo que me proponga, maestros de mil enseñanzas. Decir GRACIAS es poco por todo lo que me han dado y no me alcanzara la vida para darles lo mucho que ellos se merecen. Los amo muchísimo.

Silvia gracias por ser mi compañera de juegos, amiga y cómplice en aventuras, gracias por ayudarme y apoyarme en todo momento. Te quiero mucho.

Tía Paty, por aguantarme, ayudarme y apoyarme cuando lo he necesitado gracias por todo.

A mis Tíos y Primos que siempre me apoyaron y me daban ánimos para seguir adelante. A mis Sobrinos por brindarme sus sonrisas e iluminarme de alegría.

M. I. Alvaro, "Inge", gracias por su paciencia, por sus enseñanzas, por su confianza y apoyo. Gracias a su esposa Mayra, por darme ánimos, a Shanty y Stephy por sus sonrisas.

Sofi y Liz gracias por brindarme su amistad, por ayudarme y dejarme ser parte de este gran logro.

Gabriela Ávila Zarazúa

A la memoria de mí Padre:

Profre: Ricardo Villarruel Ramírez

Gracias por enseñarme el camino de la perseverancia y la mejora continua en la vida. Esta tesis es la culminación del gran tesoro que tu me mostraste, y aunque ya no estés aquí, se que mis logros por muchos o pocos que sean, honran tu memoria papá.

A mi Mamá y Hermanos:

Profra: M. De Jesús Flores Mecalco
Lic. Alma Nelly Villarruel Flores
Ing. Ignacio Ramírez Flores
Ing. Guadalupe Ramírez Flores

Gracias por el apoyo brindado en el transcurso de mis estudios y por enseñarme valores como la tenacidad y la constancia para el logro de esta meta. Gracias mamá por darme uno de los tesoros más grandes en la vida, como lo es mi profesión.

A mi Asesor:

M.I. Álvaro Ayala Ruiz

Gracias por brindarnos sus conocimientos, apoyo y tiempo en el transcurso de la elaboración de esta tesis, esperando que este trabajo haya sido satisfactorio para usted.

A mis amigas y compañeras:

Gabriela Ávila Zarazúa
Sofía Martínez Araujo

Gracias por brindarme su apoyo, amistad , y conocimientos, gracias por compartir momentos difíciles y alegres al realizar esta tesis, ya que sin su ayuda este logro hubiera sido más difícil. Gracias amigas.

Lizbeth Villarruel Flores

ÍNDICE

CAPÍTULO IV

4. CASO DE ESTUDIO

INTRODUCCIÓN

Para lograr que las empresas reduzcan tiempos y costos en el diseño y manufactura de sus productos, se recurre al uso de sistemas de cómputo que asistan a los diseñadores en la manufactura de un producto.

Los sistemas que asisten al diseño de manufactura del producto son llamados Sistemas CAE (Ingeniería Asistida por Computadora). Estos sistemas consultan y almacenan información en bases de datos, de tal forma que puedan contener toda la información referente a un producto. Estas bases de datos se les llama Modelos de Información, las cuales permiten integrar la información del producto para que pueda ser utilizada por diferentes programas.

Dentro de la Facultad de Ingeniería se desarrolló un sistema CAE llamado *"Sistema Auxiliar para el Diseño de Ejes de Transmisión": SADET* (proyecto J-27775U de CONACYT), el cual está formado por diferentes aplicaciones que utilizan dos modelos de información: Modelo del Producto (MP) y Modelo de Manufactura (MM). El MP contiene información basada en las características del producto. El MM almacena información relacionada con las capacidades de manufactura de las instalaciones que soportan a la Ingeniería Concurrente.

Cada aplicación o programa de SADET permite poblar los modelos de información MM y MP, con ayuda del manejador de Bases de Datos Object Store. Sin embargo no se cuenta con una aplicación que integre la información para ayudar en el diseño y manufactura de productos. Por ello se propone en esta tesis desarrollar el Sistema de Evaluación de Manufactura (SEM) para un producto con el objetivo principal de evaluar la factibilidad de manufactura del producto en base a las capacidades de manufactura proporcionadas. El sistema SEM recupera información tanto del MP como del MM y a partir de ella es posible determinar la factibilidad o la no factibilidad del producto a evaluar generando una lista de procesos para manufacturarlo.

En el capítulo uno se estudian los Modelos de Información, así como las aplicaciones que hacen uso de ellos. Dentro del capítulo dos se desarrolla el planteamiento del problema y se establecen los objetivos a cumplir por parte de SEM. Mientras que en el tercer capítulo se hace el diseño para la construcción del sistema utilizando UML como herramienta modeladora, también se habla de las herramientas usadas para lograr la implementación del sistema, las cuales son: Java (Lenguaje de Programación) y Object Store (herramienta usada para la implementación de las bases de datos), este capítulo incluye además la Implementación del sistema, describiendo la interacción con el usuario. Por último, en el capítulo cuatro se presenta un caso de estudio para comprobar el comportamiento del sistema con las bases de datos propuestas en SADET, también se describe la interacción del usuario con el sistema para realizar la evaluación y finalmente se presentan las conclusiones, apéndices y referencias bibliográficas.

CAPÍTULO I

1. MODELOS DE INFORMACIÓN

1.1 INTRODUCCIÓN

Para realizar la manufactura de productos es necesario conocer y trabajar con los diferentes puntos de vista que existen en una empresa. La principal tarea en el diseño de productos, es la determinación y evaluación de las características y propiedades del producto, sin embargo para obtener estas el diseñador se enfrenta a diversos problemas que pueden ser directos o indirectos.

Algunos se refieren a las formas y estructuras que son desarrolladas específicamente para cumplir con funcionalidad, con especificaciones, requerimientos estéticos, etc., sin embargo la definición de la estructura también determina una serie de características indirectas del producto, tales como calidad del producto, costos, logística, manufactura, etc.

El diseñador finalmente se enfrenta a desarrollar sus tareas con una influencia en diversas áreas, así que la calidad de sus tareas depende de la eficiencia y la capacidad de relacionarse con los diversos departamentos, de tal manera que los cambios en el diseño sean provocados por la concurrencia de los diversos departamentos.

Para poder desarrollar estas tareas, el diseñador cuenta con una serie de técnicas metódicas y de soluciones organizacionales, una de estas técnicas es el diseño para la manufactura (CAD-CAM, CAPP) [Ayala, 2001].

Es necesario integrar las diferentes herramientas de cómputo (CAD-CAM, CAPP) que se usan durante la realización de productos para lograr así un soporte integral que asista eficazmente a equipos de diseño. Esto es posible si todos los programas consultan y almacenan datos empleando el mismo depósito de información o base de datos [Pasad, 1996].

1.2 INTEGRACIÓN DE LOS SISTEMAS CAD/CAM

En años recientes el diseño ha visto un número de herramientas de software producidas para auxilio de los diseñadores. En donde para el CAD (Diseño Asistido por Computadora) van desde simples auxilios en 2D hasta sistemas con capacidades paramétricas, para modelos 3D, dando capacidades de visualización poderosas. Por el mismo camino se encuentra el CAM (Manufactura Asistida por Computadora) con sus respectivas herramientas de software.

Muchas funciones de manufactura usan la información que se encuentra en los planos de ensamble, fabricación, etc. como una mejor entrada que permite intentar traer

juntos a los paquetes de software CAD y CAM. El cual es un camino para que el software CAM pueda usar datos generados provenientes del uso de paquetes CAD.

Generalmente los paquetes comerciales que proveen el auxilio al diseño y manufactura atacan problemas específicos, en paquetes de software ligados por la provisión de una interfaz. Por lo que se ha intentado mejorar la generalidad de interfaz para paquetes CAD, de ahí que se ha estado avanzando para algunas extensiones, con Sistemas de Intercambio de Gráficas Iniciales (IGES) intentando proveer gráficos estándar y recientemente incluyendo estándares para la descripción de información que se encuentra en los planos de ensamble, fabricación, etc., así como su forma.

La escala del problema existe en gran medida por la interfaz. Los simples sistemas CAD/CAM dependen extremadamente del usuario con información limitada proveniente del sistema de computo. Esto es necesario por dos razones: la decisión del usuario no es realmente almacenada en la forma software y la información en la cual el usuario basa sus decisiones necesita interactuar con los procesos de decisión.

De ahí que se ha estado trabajando en paquetes de software individuales en los años 70´s mientras que para los 80´s fue en sistemas de interfaz, así como un simple intento de sistemas integrados en los 90´s.

Para alcanzar el intento de integración se plantea establecer una base común de datos de producto en donde se proveerán ligar eslabones integrados para las funciones de diseño y manufactura.

1.2.1 Elementos del sistema CAD/CAM integrado.

Son varios elementos por explorar para lograr exitosamente los sistemas integrados CAD/CAM . Dos de los elementos clave son la representación de datos (la cual será la base de decisiones) y la exploración de módulos funcionales de software (los cuales pueden actuar en estos datos para proveer información útil a los ingenieros para el diseño y manufactura).

La necesidad fundamental en la investigación para la integración esta dada con un énfasis particular por Ham y Lu. Ellos identifican que varios problemas deben ser resueltos ante sistemas que son realmente integrados, sugiriendo 4 problemas de software básicos que indican la necesidad para la elaboración de una base de datos común como:

1) Problema combinatorio. (El número de interfaces exploradas con el incremento en número de sistemas).

2) El problema de redundancia e inconsistencia. (Múltiple almacenaje de datos; diferentes estados de poner al día).

3) El problema de paquetes de software cerrado. (No accesa la estructura de datos; los algoritmos no son disponibles).

4) El modelo del problema. (Diferentes modelos en diferentes sistemas de software).

Por considerar un sistema CAD/CAM, ellos identifican que el problema de integración puede solamente ser resuelto por una planeación correcta del modelo de estructura. Ellos reclaman "el completo modelado de un producto, incluye toda la información necesaria para manufactura en los datos, que es un requisito básico para la integración de CAD y CAPP" [Young, 1989].

1.3 INGENIERÍA CONCURRENTE

La Ingeniería Concurrente (IC) es una filosofía de desarrollo del producto que afecta en su funcionamiento a todas las áreas de la empresa.

Precisa la implantación de un trabajo en equipo de técnicos de las distintas áreas para lograr en un tiempo reducido un producto que responda a las expectativas de los usuarios con una calidad y costo adecuados.

Este equipo dispone de una serie de tecnologías y metodologías de trabajo más o menos conocidas y en general poco utilizadas que se han ido desarrollando independientemente y que en ocasiones se solapan y en algunos se contraponen.

Dispone de elementos de automatización basados en software de CAD-CAE-CAM que en muchas ocasiones presentan dificultades de integración.

No existe una metodología universalmente aceptada para la implantación de la Ingeniería Concurrente. Si bien se habla ya en bastantes ocasiones de que se trabaja utilizando la Ingeniería Concurrente, la realidad de experiencias de equipos de diseño multifuncional amplios y de utilización de las distintas tecnologías y metodologías de estudio y análisis es muy escasa. Quizás solo pueda destacarse una amplia utilización del CAD pero en aplicaciones de ingeniería clásica. [Sohlenius, 1992]

Para potenciar la utilización de la Ingeniería Concurrente se precisa [Sohlenius, 1992]:

- Desarrollar planes y métodos de formación eficaces para la difusión y conocimientos de esta técnica.
- Desarrollar y probar metodologías de implantación que orienten a las empresas y faciliten alcanzar buenos resultados.
- Desarrollar criterios de definición de las diferentes tecnologías y estudiar y sistematizar su aplicación.

- Desarrollar bases de datos del producto que integren toda la información de diseño, pruebas, fabricación, calidad, etc., que permita una fácil comunicación e intercambio de información entre los distintos departamentos.

1.4 LOS PRIMEROS MODELOS DE MANUFACTURA

1.4.1 Definición de Modelo de Manufactura

La literatura revisada ha mostrado que el Modelo de Manufactura es un nombre genérico que identifica 2 tipos de modelo:

1. Modelos CIM (Manufactura Integrada por Computadora) que representan las funciones de la empresa, módulos de Software, arquitecturas de sistemas de información y control que pueden ser usadas para diseñar e implementar sistemas CIM por ejemplo: el modelo IBM, el modelo Jerárquico NIST-AMRF, el modelo de Siemens AG, el Modelo DEC y el modelo de referencia de manufactura de partes discretas.

La investigación sobre los modelos CIM ha contribuido al desarrollo de nuevos modelos de manufactura genéricos, y por lo tanto influenciados por el trabajo sobre el modelado de la empresa. Un ejemplo de este trabajo es la creación del modelo CIMOSA (Proyecto ESPRIT 688/52388) el cual ha evolucionado desde el soporte de diseño e implementación de sistemas CIM hasta el desarrollo de modelo de empresa.

Aunque el ámbito de un modelo CIM es más limitado, este puede ser visto como un modelo, parte de un modelo de la empresa para una aplicación especifica en el sistema CIM.

2. Modelos de Información que representan los datos e información que describen la fábrica en términos de sus recursos y procesos o ambos, por ejemplo el Modelo de Fábrica, el Modelo de Instalación [Molina, 1992], Modelo de recursos de Manufactura (ISO TC184/SC4/WG8 /N13) y el Modelo de Manufactura [Young, 1989]

1.5 MODELOS CIM

La manufactura integrada por computadora empieza con la concepción de un nuevo diseño, donde las funciones y características del producto son concebidas. El diseño del producto determina enormemente los métodos y procesos de manufactura. En la manufactura integrada por computadora el concepto de computadora es considerado como un factor integrador para el control de la interacción del hombre y las máquinas para crear el producto desde el diseño para manufactura.

La construcción de un sistema de manufactura integrada por computadoras implica un proceso muy tedioso y largo. Esto requiere de expertos en manufactura para ligar las computadoras a varias actividades de un sistema de producción e interconectarlos como una entidad funcional. Sin embargo, si el diseñador y los ingenieros de manufactura cuentan con un modelo CIM es posible construir un sistema de manufactura particular para algún producto que se desee construir [Ulrichrembold, et. al., 1991]. Las actividades representadas por un modelo CIM deben de poner en funcionamiento varias etapas de organización que son ejecutadas en una forma paralela o secuencial. Cuando se construye un modelo es necesario considerar varios aspectos aunque no todos son incluidos dentro de él, estos son los siguientes:

1. Presentación de las funciones de la empresa

Un modelo típico CIM es el que se muestra en la figura 1.1. El énfasis en este modelo es la aplicación de la computadora en varias actividades de una fábrica. Se muestra que el Control y Seguridad de Calidad (CAQ) y el Control de Programa de Producción (PPC) son actividades superpuestas haciendo interfaces con CAD, CAPP y CAM. Pero sin mostrar detalles.

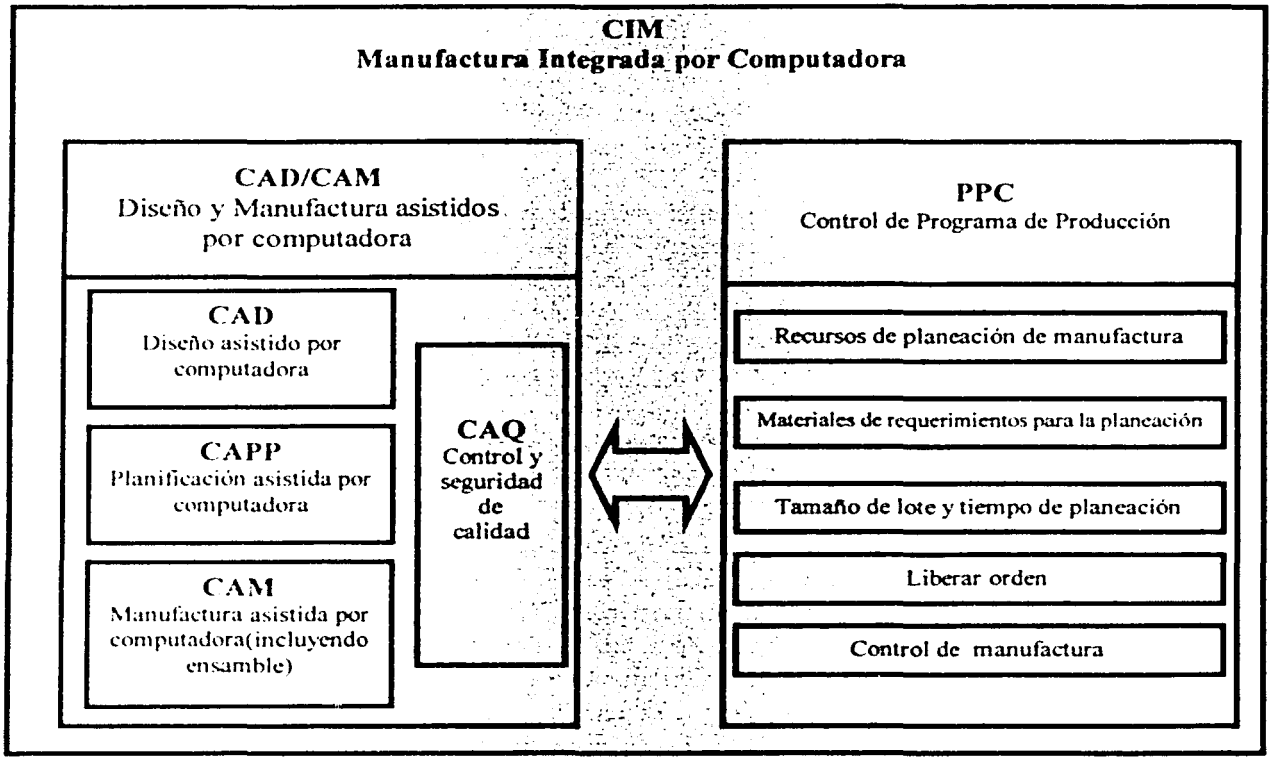


Figura 1.1 Presentación de las funciones de la empresa.

2. Integración de la administración de la base de datos con información

En este modelo las bases maestras CAD/CAM ejecutan una función de integración en una empresa de manufactura integrada por computadora. La figura 1.2 muestra la integración antes mencionada [Ulrichrembold, et. al., 1991].

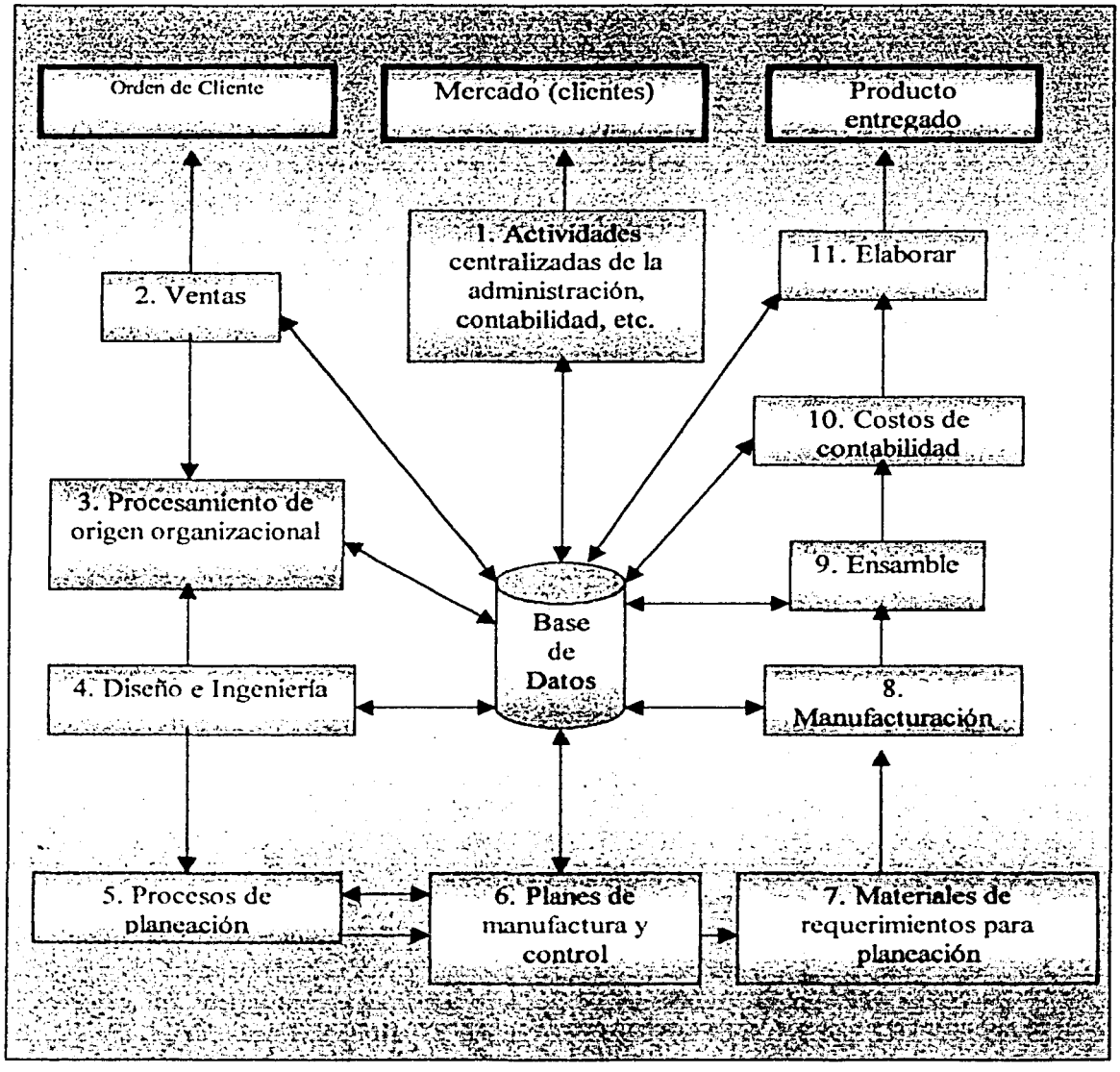


Figura 1.2 Modelo de las Bases Maestras CAD/CAM

3. Presentación del material y del flujo de productos

Este modelo es muy útil para asistir el trazado físico de una planta al estudiar el flujo de material durante su manufactura. La figura 1.3 muestra aun ejemplo de este modelo con un material típico de manufactura.

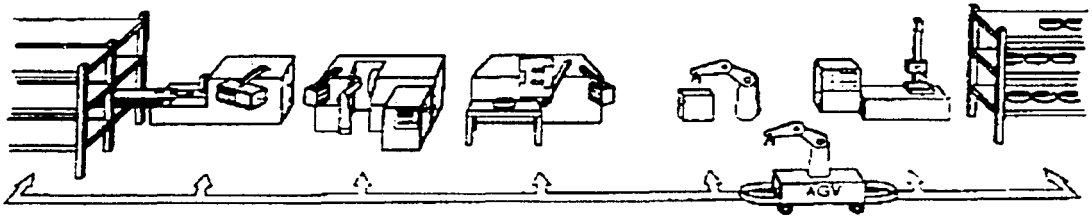


Figura 1.3 Modelo del material y flujo del producto.

4. Presentación del flujo de información

El flujo de información puede estar relacionado por la orden o por el producto. Un ejemplo del flujo de información representado por la orden se muestra en la figura 1.4.

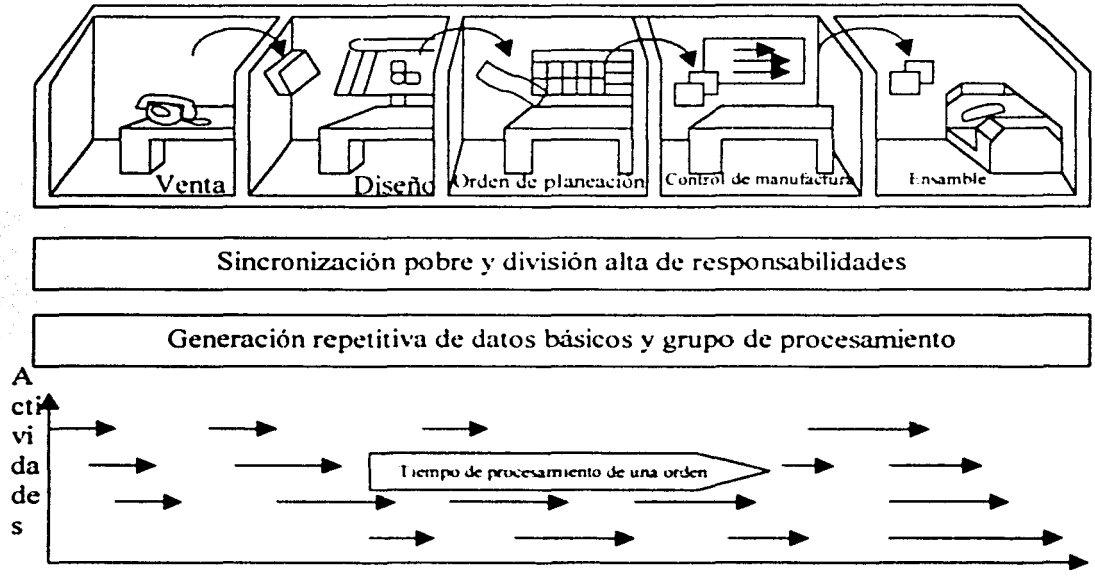


Figura 1.4 Modelo del flujo de una orden a través de una fábrica.

5. Descripción de las interfaces y los protocolos de comunicación

En esta aproximación la estandarización de interfaces y protocolos es sometido a la posibilidad de usar componentes de sistema modular.

6. La presentación de la planeación jerárquica y las funciones de control

La ventaja de este modelo es mostrar como la información para la planeación y las funciones de control están representadas de forma jerárquica desde lo abstracto hasta un nivel detallado.

7. La inclusión del tiempo

En la manufactura la entrada de los planes es muy importante, por esta razón una característica que sucede es mostrar los tiempos de manufactura sobresalientes como una gran ventaja.

A continuación se mencionan algunos modelos CIM que se han desarrollado.

1.5.1 El concepto de CIM de IBM

El compromiso de IBM en la manufactura con el procesamiento de datos empezó con la introducción de la computadora en la fábrica. Rápidamente encontró un mercado potencial para el hardware y software dentro de las industrias que empezaron a desarrollar varios módulos de procesamiento de datos para un lugar especifico o aplicaciones para los clientes. Así que un intento fue realizar productos que pudieran ser usados por otras industrias sin cambio de nombre u otro tipo de cambios. Sin embargo fue una tarea difícil porque era necesario hacer esos cambios para que la planta fuese automatizada correctamente. Así que para 1970 desarrolló una filosofía general de CIM llamada Comunicación Orientada a la Producción de Información y Sistema de Control (COPICS) la cual contiene toda la planeación y actividades de control para manufactura incluyendo:

1) Orden de servicio al cliente:
   Pronóstico
   Planes de producción y planeación
   Administración de inventario
   Planeación de actividades de manufactura
2) Orden de liberación:
   Control y monitoreo de la planta
   Mantenimiento de la planta
   Compra y recibimiento
   Tiendas de control
   Planeación de costos

El procesamiento de la información que soporta IBM es para proporcionar administración, toma de decisiones y varias aplicaciones. Para 1989 IBM puso énfasis en hacer una comunicación con la administración de la base de datos así como promover en el sistema de COPICS su propio software y hardware.

1.5.2 El modelo jerárquico AMRF de NIST

Este modelo fue propuesto por el Instituto Nacional de estándares y tecnología, aunque formalmente la organización fue conocida como Agencia Nacional de estándares NBS. El modelo AMRF (Facilidad de Investigación de Manufactura Avanzada) se hizo para operar hardware y software estándares para sistemas de manufactura controlados por computadora. Este es un diseño modularizado. Una computadora jerárquica y un sistema de sensores están para planear y controlar las operaciones de manufactura. Al igual que estos modelos facilitan la configuración de un sistema de control de manufactura. El modelo consiste de tres columnas: un sistema de información guía, un sistema de control y un sistema de planeación y diseño, como se ve en la figura 1.5.

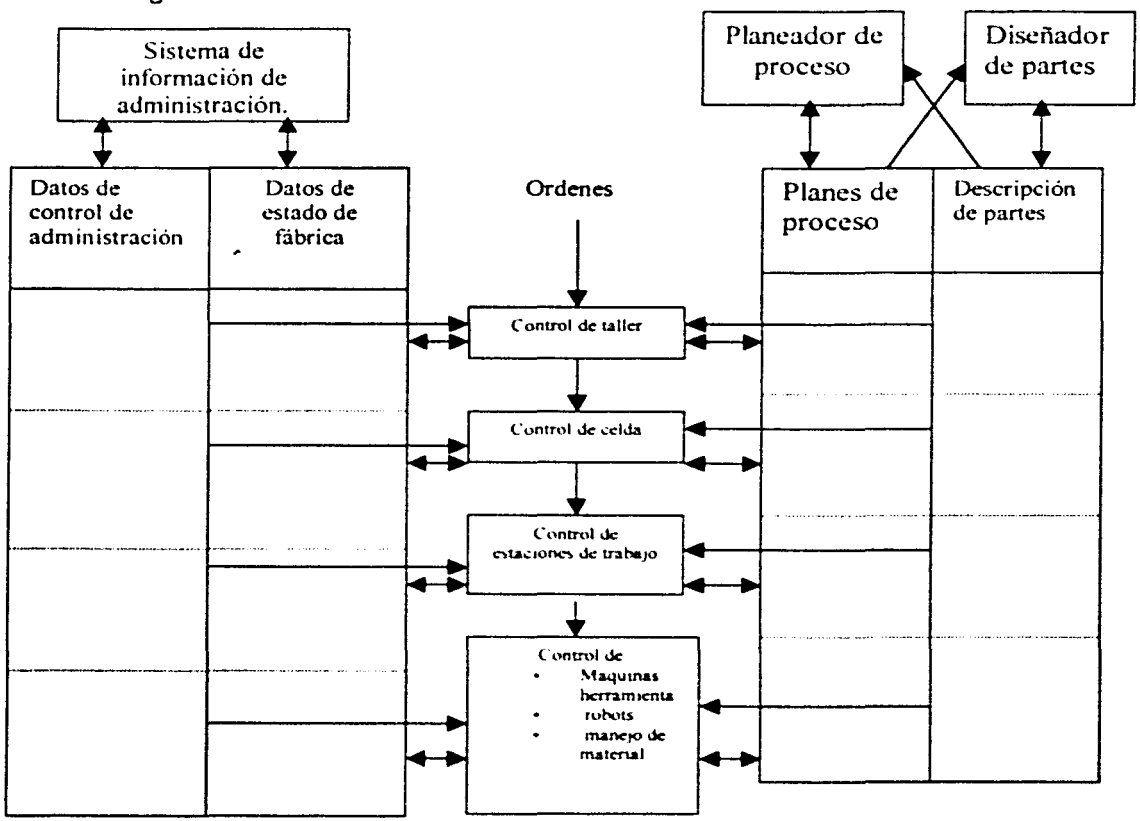


Figura 1.5 Arquitectura CIM del AMRC de NIST.

El sistema de administración de información controla y coordina la planta de manufactura completamente. El sistema de sistema de diseño y planeación prepara los documentos de manufactura. La información sobre la orden es entregada al nivel más alto de la jerarquía del sistema de control y las metas globales con las estrategias son decididas. La orden inicia la manufactura del diseño y las actividades de planeación. En cada nivel más abajo el procesamiento de la orden es redefinido exitosamente hasta el final de la jerarquía para obtener un grupo de instrucciones de control primitivas que operarán el equipo de manufactura directamente. Los comandos para una asignación de manufactura son pasados en secuencia a través de los niveles de control, estos pueden invocar operaciones paralelas o secuenciales. Los sensores sobre los niveles de la fábrica colectan información de estados sobre los procesos de manufactura y envían ésta hacia arriba para alimentar otros niveles. El sistema entero es un sistema modular de diseño incluyendo el hardware y software.

1.5.3 El concepto de CIM para Siemens AG

Siemens vende productos de procesamiento de datos y usa estos en sus numerosas actividades de manufactura. El CIM no es un producto hecho pero si una estrategia y un concepto que lleva a cabo las metas de mercado en una empresa. El modelo de Siemens comprende las principales funciones: planeación, ventas, compras, PPC, CAD, CAQ y CAM. Estas funciones son interconectadas por un flujo de información intenso. La tarea de este flujo es el procesamiento y distribución de datos en una manera rápida y concisa. Para llevar a cabo el procesamiento de datos en un sistema de manufactura se une un modelo jerárquico de empresas. Cada nivel jerárquico tiene sus propios requerimientos del procesamiento de datos y existe un flujo estable de instrucciones desde los niveles altos hasta los más bajos. Una característica del modelo de Siemens es que incorpora una organización asistida por computadora que contiene: contabilidad, personal y finanzas. Debido a esto existe una liga muy fuerte entre CAD y CAM. [Ulrichrembold, et. al., 1991].

1.5.4 El concepto CIM de la Corporación de Equipo Digital

La Corporación de equipo Digital es un productor y un usuario del hardware y software de CIM. El término CIM (Manufactura Integrada por Computadora) significa digitalizar el desarrollo de los procesos de manufactura con la asistencia de la computadora y la integración del procesamiento de la información de todas las actividades de la empresa. El modelo de Digital se parece al de Siemens y tiene muchas características idénticas. Las actividades individuales de este modelo son soportadas por un modelo de tecnología estructurado. Siendo este modelo una extensión del modelo OSI (Organización Internacional de Estándares).

El modelo asume que la manufactura integrada por computadora es una aproximación orientada al negocio para la automatización de un sistema de control jerárquico por computadora. Las metas y objetivos de este modelo son definir un sistema de manufactura integrado por computadora tan consistente como sea posible para proporcionar documentos referentes a la planeación e implementación distribuida.

Con esto el sistema de control completo es dividido en modelos funcionales que reflejan el negocio y sus datos. El diseñador de un sistema CIM empieza con el análisis de todas las actividades de manufactura y define sus funciones junto con el flujo de datos. De esta forma con la información generada el trazado del sistema físico puede ser hecho. Un modelo funcional y un modelo físico de un sistema de manufactura y sus subsistemas son el resultado. Esto da un buen sistema que modulariza las funciones para reutilizar paquetes y define claramente las interfaces para la fácil configuración del sistema.

1.5.5 Modelo ESPRIT CIM-OSA

El ESPRIT (Programa estratégico Europeo para la investigación y desarrollo en la tecnología de la información) es un proyecto que fue creado en 1981 por la unión de las industrias europeas y la comisión europea. Está orientado a levantar la competitividad industrial de la comunidad Europea. La manufactura integrada por computadora es una materia importante dentro del programa ESPRIT porque la economía Europea depende fuertemente de las industrias manufactureras.

La estrategia del ESPRIT ha sido la creación de un ambiente en el cual los sistemas de producción multi-vendedores puedan ser implementados por un costo razonable. Para este propósito una arquitectura de sistema abierto se creo para permitir la configuración de instalaciones de manufactura desde módulos genéricos.

Dentro de la estructura del concepto del CIM-OSA es posible construir arquitecturas CIM para varias industrias manufactureras así como aplicaciones desde bloques de construcción básicos de acuerdo a los productos definidos. Una empresa es un sistema complejo de combinación de las acciones de la gente y de las máquinas con la asistencia de las operaciones de procesamiento de datos. La infraestructura de integración ayuda a las compañías para organizar dinámicamente sus actividades.

Un sistema de manufactura esta compuesto por módulos de entidades funcionales como se ve en la figura 1.6. Una entidad funcional es un ideal, un objeto independiente con implementación que describe un componente conceptual de una función del negocio. Una vez que la entidad ha sido designada esta existirá permanentemente y podrá ser usada para operar parte el sistema del negocio. Estas entidades pueden actuar en secuencia o en paralelo. El intercambio de datos relacionados con las entidades funcionales es hecho por vía transacción.

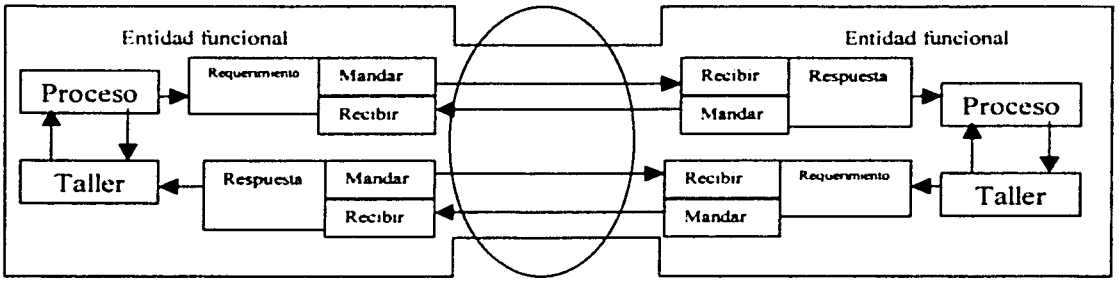


Figura 1.6 La imagen muestra las entidades funcionales y sus menús de comunicación.

Una interfaz de ejecución es usada para ligar los servicios de procesamiento de datos básicos a la estructura de control deseada. El modelo CIM-OSA también proporciona una infraestructura de integración al usuario que contiene un conjunto de funciones y da soporte para la integración de estas. La infraestructura de integración actúa como un tipo de configuración combinada con el sistema de operación, la cual liga las entidades funcionales para un sistema de control operable.

La infraestructura de integración del modelo CIM-OSA tiene los siguientes componentes:

Aplicación: El usuario define su aplicación por una vía terminal u otro tipo de dispositivo de entrada.

Comienzo y servicio: Este módulo es la interfaz de aplicación, a través de esta operación todas las funciones del proceso son definidas en una forma homogénea como entidades funcionales por los humanos, máquinas, sistemas de procesamiento de datos externos que necesiten algunos sistemas de manufactura.

Administración de los procesos del negocio: Esta función es responsable para montar la estructura de control, la cual es necesitada por las entidades funcionales externas para operar como una unidad.

Administración de la información: Esta función es responsable de proveer un servicio de información adecuado con integridad de los datos, consistencia, fiabilidad y redundancia. Cuidando el acceso correcto y las prioridades del usuario tanto del almacenamiento de datos como de su recuperación y conversión.

Sistema amplio de intercambio de servicios: Este servicio es responsable para la guía de la comunicación de los datos. Suministra los canales de datos, asegura el acceso de los datos independientemente y sincroniza y guía la transferencia de datos. Una función adicional es la grabación de los datos para un procedimiento de recuperación posible.

Dentro del concepto del modelo CIM-OSA una empresa consiste de ingeniería y funciones de operación. Las funciones de ingeniería contienen la arquitectura de referencia y los recursos son usados para estructurar y elevar una operación de empresa. El modelo asume que hay una comunidad de soporte que provee software estandarizado y modelos de hardware para manufactura. Desde que el software y los módulos de hardware son hechos de acuerdo a los estándares del CIM-OSA la configuración del sistema es muy fácil. El usuario puede tratar varias estrategias de manufactura y puede optimizar estas con ayuda del modelado y simulación de herramientas.

Los puntos fuertes de esta aproximación del CIM-OSA es el modelo CIM genérico, la librería de módulos de hardware y software y la posibilidad de configurar el sistema de manufactura con la asistencia de infraestructuras de integración [Ulrichrembold, et. al., 1991].

1.6 MODELADO DE INFORMACIÓN

En esta sección describiremos la metodología integrada, concebida para ser usada en el trabajo de modelado de información. Las dimensiones de modelado consideran necesario representar las capacidades de manufactura explicitas. La parte de los recursos de manufactura, procesos y estrategias para la representación genérica y compañía especifica de información de manufactura esta más allá. La necesidad para representar estrategias e información operacional es discutida y la estructura multinivel será definida.

1.6.1 Metodología del modelado de información de manufactura

La investigación del modelado de manufactura esta principalmente enfocada con la representación de la información de capacidad de manufactura de instalación. Para facilitar esta tarea se define la metodología del modelado de información de manufactura para ser usada y ejecutar el modelado de información de manufactura en el grupo de investigación MOSES. La metodología fue definida para:

1. Colocar un contexto y armonizar la investigación para modelar la información de manufactura.

2. Establecer un método formal para el desarrollo de modelado de manufactura.

3. Construir un consistente, completo y sólido modelo de manufactura.

4. Estructurar y acoplar otras contribuciones de investigación relacionados para la investigación del modelo de manufactura.

La idea de usar una representación multidimensional para describir la metodología de integración fue tomada desde CIMOSA. Aquí se reconoce la necesidad para representar todos los elementos envueltos en la investigación del modelo de manufactura. Esta representación multidimensional contiene todos los elementos que se consideran para ser necesario y adecuadamente describir la capacidad de manufactura de instalación. La metodología esta compuesta de:

1. Las dimensiones del modelado: datos, funciones y comportamiento.

2. Las entidades de información de manufactura: recursos, procesos y estrategias.

3. La organización de estas entidades de información de manufactura dentro de una estructura multinivel: Fábrica, Taller, Celda y Estación.

Esta representación multidimensional ayuda a identificar los siguientes resultados acerca de la investigación del modelo de manufactura.

- ✓ La necesidad para los diferentes aspectos del modelo de los recursos de manufactura, procesos y estrategias capturan las características relevantes para el soporte estratégico y decisión operacional. Estos aspectos son llamados dimensión de modelados y permite la representación de: datos (clases, composición y atributos), funciones (capacidades funcionales), y comportamiento (estatus y ejecución).

- ✓ Los recursos de manufactura y procesos describen información de manufactura genérica. Tipos similares de manufactura y procesos pueden ser utilizados por diferentes compañías. De hecho dos compañías pueden tener el mismo tipo de tecnología, ejemplo: recursos de manufactura y procesos. No obstante la instalación de manufactura de cada una de estas compañías se puede ejecutar por diferentes caminos, por la decisión de las compañías en como organizan y usan esos recursos y procesos. Este aspecto debe ser aceptado para representar verdaderamente la capacidad de manufactura de una compañía. Las estrategias de manufactura representan esta información específica de la compañía, la cual permite a una compañía especificar como los recursos y procesos son organizados y usados para soportar las funciones de manufactura de la misma.

- ✓ La relevancia de definición de la estructura multinivel para representar estrategia e información de manufactura operacional. Esta estructura multinivel permite la representación del nivel de fábrica de la estrategia de información de manufactura la cual define que la estrategia de manufactura de la compañía esta bajo los tres niveles (Taller, Celda y Estación). El nivel de fábrica es dirigido para la representación de la formulación de estrategias de manufactura. El resto de los tres niveles representa la información operacional de la instalación de manufactura para soportar las actividades del ciclo de vida del producto (ejemplo, diseño y manufactura).

1.6.2 La necesidad para el modelado de dimensiones diferentes

La necesidad para identificar el modelado de dimensiones diferentes requeridas para representar objetos reales existen en universo de discursos.

Basándose en diversas investigaciones (CIMOSA, IMPPACT, etc.) y en otras referencias de modelado, especialmente métodos orientados a objetos, se comprende que las diferentes propiedades de un objeto en el mundo real tienen que ser modeladas. Estas propiedades representan diferentes aspectos de un objeto el cual permite capturar la esencia de un objeto. Preguntas tales como: ¿Qué es un objeto, y cuál es su papel?, ¿ Cuales son las relaciones entre objetos?, ¿Qué puede hacer un objeto?, ¿Cómo evaluar el objeto en el tiempo?, y ¿ Cómo el objeto ejecuta su trabajo?, para que puedan tener una respuesta y luego modelar la información considerando un objeto, se identificaron las siguientes dimensiones de modelado:

1. Modelando los datos los cuales describen cual es el objeto y cual es su papel en el universo de discursos.

2. Modelando la información para representar las relaciones entre objetos y lo que puede hacer un objeto.

3. Modelar el comportamiento para expresar como evoluciona el objeto en el tiempo y cual es la ejecución del objeto.

Para explicar cuales son estas dimensiones de modelado en el modelo de manufactura, se usa la representación gráfica de gráficas conceptuales [Sowa, 1984]. La gráfica conceptual permite la descripción de conceptos, sus atributos y relaciones por interconexión de ellos usando semántica estática, conocida como relación conceptual.

1.6.3 Modelado de datos

Se han usado diferentes mecanismos de modelado de abstracción empleados para técnicas de modelado orientadas a objetos [Booch 1991, Rumbaugh 1991] para describir que es un objeto y cual es su papel en el universo de discursos. En esta investigación los objetos son las entidades de información de manufactura (recursos, procesos y estrategias) y el universo de discursos es la instalación de manufactura.

El uso de este mecanismo de abstracción facilita la organización de entidades de manufactura en su clasificación (taxonomía de clases) para simplificar su descripción. Atributos generales son especificados en clases (comunes) mientras que atributos específicos son definidos en las clases especificas. Porque las clases más especializadas tendrán los mismos atributos de aquellas clases parientes en adición a otras definiciones para clases particulares. Este es uno de los más poderosos mecanismos de abstracción de métodos orientados a objetos [Booch 1991].

La agregación (tener relación) permite la composición de entidades de manufactura basadas en sus componentes funcionales. La agregación es muy importante porque permite la definición completa y composición de recursos, procesos y estrategias.

1.6.4 Modelado de función

La descripción de sistemas de funciones y los procesos de descomposición funcional y de categorización de las relaciones entre funciones es proveída modelando la función [Klein y Scheer, 1990]. Se define que el modelar la función es relacionado a la representación de las capacidades funcionales de entidades de manufactura (ejemplo, que pueden hacer los recursos de manufactura, procesos y estrategias).

En el modelo de manufactura las capacidades funcionales de las entidades de manufactura son definidas y basadas en relaciones semánticas entre objetos. La característica de los métodos orientados a objetos representa relaciones semánticas simplificando su trabajo.

1.6.5 Comportamiento de modelado

El comportamiento de modelado describe la dinámica de un sistema. Todas las entidades en un sistema tienen algún estado en tiempo, ejecución y operación, y exhibición de alguna ejecución. Modelar el comportamiento permite la representación de tiempo de dependencias (secuencia, paralelismo y concurrencia), ejecución de operación y ejecución. Los métodos de modelado están bajo desarrollo, sin embargo no hay una metodología conocida que adecuadamente describa el comportamiento dinámico.

1.6.6 La descripción genérica e información de manufactura especifica de una compañía

Se han identificado dos tipos de capacidad de manufactura de información de una compañía. Información acerca de capacidades genéricas de instalación basadas en factores técnicos, e información de capacidad especifica de la compañía basada en organización y factores operacionales.

La información de capacidad genérica es relacionada a los aspectos tecnológicos de instalación. Esta información representa capacidades teóricas basadas en cuáles son los recursos de manufactura y procesos que se pueden hacer, y cual es la facilidad con que pueden producirse. Esta información esta relacionada para las limitaciones físicas de los recursos y procesos los cuales incluyen a otros: tamaño del producto, peso del producto, forma del producto, volumen, materiales, tolerancia y superficie terminada.

El segundo tipo de capacidad de manufactura es relacionado con las estrategias de decisión y reglas operacionales impuestas en uso, organización y composición de

los recursos y los procesos de manufactura. La capacidad teórica de unos recursos o procesos puede ser dimensionada dependiendo de sus estrategias impuestas.

La inclusión de las estrategias en el modelo de manufactura es algo novedoso lo cual habilita el modelo para representar la capacidad especifica de información de una compañía.

1.6.7 La representación de información estratégica y operacional

Actualmente las empresas de manufactura han adoptado cinco o cuatro niveles funcionales para organizar sus funciones. Los cuatro niveles que se han usados para representar la estructura del modelo de manufactura son los siguientes:

- ✓ Fábrica.
- ✓ Taller.
- ✓ Celda.
- ✓ Estación.

Estos niveles de abstracción proveen información de manufactura para todas las actividades funcionales dentro de una empresa de manufactura. Los diferentes niveles del modelo de manufactura representan las diferentes perspectivas que el usuario pueda tener acerca de la capacidad de información de manufactura. Estas diferentes perspectivas representan diferentes puntos de vista acerca de los recursos y procesos y su factibilidad.

1.6.8 Nivel de Fábrica

La fábrica sostendrá la información estratégica, la cual representa la estrategia de manufactura de una compañía, los siguientes niveles (taller, celda y estación) representan información operacional la cual describe como la estrategia de manufactura ha sido realizada.

La estrategia de manufactura es el set de decisiones las cuales determinan la capacidad de factibilidad de manufactura. Estas decisiones influyen en el camino a la factibilidad de su estructura composición y organización. Los tres niveles más bajos del modelo de manufactura son la representación de como estas decisiones han sido realizadas.

El nivel de fábrica describe la estrategia de manufactura en términos de decisión de estrategias, reglas operacionales y ejecución de medidas. Las reglas operacionales definen como las decisiones de estrategia son reforzadas y activadas. La ejecución de medidas son definidas para permitir la evaluación de la situación de manufactura usando los niveles más bajos del modelo de manufactura.

1.6.9 Niveles de taller, celda y estación.

La información operacional comprende la composición, estructura y organización del taller, celda y estación que es representada bajo los 3 niveles del modelo de manufactura. La información representada en estos niveles comprende la capacidad del taller, celda o estación, y procesos tecnológicos, composición de estaciones dentro de celdas y celdas dentro de talleres, y las estrategias operacionales impuestas en el uso de recursos, procesos y facilidades. Una instalación en este contexto puede ser un taller, una celda o una estación. Esta información representa como la estrategia de manufactura ha sido realizada por poner juntos un set de capacidades de manufactura que habilitan una compañía con el propósito de escoger una estrategia competitiva.

1.6.10 El uso de la metodología del modelado de información de manufactura

El principal objetivo en el modelado de información de manufactura es el de escribir y representar las entidades de manufactura en los diferentes niveles funcionales en términos de datos, función y comportamiento.

La definición de la metodología del modelado permite tener sistemáticamente una captura y representar la información manufacturada, y así construir un modelo de manufactura consistente.

1.6.11 Modelos de Información del Producto y de Manufactura

Los últimos modelos de manufactura, en otras palabras modelos de información han llegado a ser un elemento importante de los modelos de la empresa ya que estos básicamente capturan y representan la información de los recursos de manufactura de una empresa. El desarrollo de modelos de empresa pueden ser basados en modelos de referencia (por ejemplo: CIMOSA, PERA, GRAI-CIM) los cuales permiten la representación de los aspectos importantes de una compañía, en otras palabras información, procesos, recursos, etc. Estos aspectos son capturados en un modelo de empresa por el modelado de diferentes vistas de la empresa. Uno de los aspectos más importantes en una empresa es su información, la cual es modelada en la vista de información.

Desde el punto de vista de Molina, él esta de acuerdo con otros investigadores y considera que la información necesaria para soportar la realización del diseño y funciones de manufactura en una empresa, puede ser capturada y representada en dos modelos de información. De tal forma Molina propone dos modelos.

1. Modelo del Producto.
2. Modelo de Manufactura .

1.6.12 Modelo de Producto (MP)

Los MP's varían dependiendo de las aplicaciones y actividades que soportan. Así pues, los requerimientos de información son diferentes para definir especificaciones, diseño para manufactura, evaluar costos, etc. La estructura y contenido de MPS para aplicaciones especificas son temas actuales de investigación [Krause, et. al., 1993]

Los MP's dependen de las aplicaciones que los usarán y de la concepción del producto que se requiera representar en ellos. Un MP puede ser estructurado en sub-modelos y la información representada en él puede incluir: descripción del producto (estructura, geometría, funcionalidad, etc.); ciclo de vida del producto (métodos usados por el diseñador, requerimientos, etc.); ambiente del producto(datos de empresas involucradas en su desarrollo, conocimiento sobre el diseño, etc.).

Para asistir el diseño para manufactura, es necesario que los MP's representen la geometría final de los productos, la evolución de esta geometría durante su proceso de producción, los procesos de manufactura y las máquinas y herramientas necesarias para su producción [Borja, et. al., 1997].

1.6.13 Modelo de Manufactura (MM)

Un MM identifica, representa y captura los datos, información y conocimiento que describen los recursos, procesos y estrategias de manufactura de una compañía particular [Molina 1995]. Un MM provee de la información de manufactura necesaria a aplicaciones para efectuar diseño para manufactura [Borja, et. al., 1997].

Aunque varios conceptos de Modelos de Manufactura han sido desarrollados, sus méritos y estructuras están intensivamente siendo discutidas. Lo principal es desarrollar modelos genéricos para tipos específicos de industrias, desde un modelo genérico de propósito general que serán poco complejos y muy abstractos. Los presentes esfuerzos han mostrado que para un modelo genérico para ser practico, solo los recursos más estandarizados y operaciones deberían modelarse. Un problema adicional es la complejidad y la tarea de la demanda de tiempo para ejecutar el trabajo del modelado de la información. Para asistir los desarrollos en esta tarea muchas ideas de ingeniería de software han sido adoptadas tales como el uso de modelos de referencia y métodos orientados a objetos.

Los sistemas de manufactura han ido evolucionando a lo largo de las últimas décadas de los 80's y más aún principios de los 90's. Con el surgimiento de la nueva tecnología en la Manufactura se ha hecho posible construir sistemas de Manufactura desde modelos genéricos de hardware y software. Estos módulos son configurados a un sistema de manufactura especifico con la ayuda de los modelos de manufactura genéricos.

El procesamiento eficiente y la distribución de la información correcta juegan un papel importante para la obtención de fábricas programables para ciertos productos

específicos. Uno de los mejores cambios en la construcción de estas fábricas es la concepción y diseño de los modelos de manufactura genéricos que proveen una colección de herramientas de software y hardware estandarizadas para que una fábrica funcional pueda ser configurada. Una vez que el producto y sus métodos de manufactura estén definidos, el ingeniero en sistemas será capaz de presentar al modelo genérico sus ideas y un modelo específico para que pueda ser construido por la selección de una biblioteca de herramientas que puedan ser necesitadas para la elaboración de su producto [Ulrichrembold, et. al., 1991].

Dentro de los desarrollos de los modelos de manufactura más importantes podemos mencionar el modelo ESPRIT-CIMOSA, el cual da soporte a la comisión Europea. Este trabajo es una influencia muy fuerte para la concepción de los nuevos modelos de manufactura, pero que sin duda alguna son el resultado del surgimiento de la tecnología de manufactura integrada por computadora CIM.

1.7 BASE DE DATOS DENTRO DEL CIM

La eficiencia del equipo de cómputo que controla la manufactura en las máquinas herramientas CNC (Control Numérico por Computadora), y en particular el costo mínimo de operación del maquinado depende directamente de la organización y de la administración de las Bases de Datos.

La información establecida en el plan de operación es: el material de trabajo y los procesos de manufactura. La selección del material de herramientas y la geométrica herramienta, que permite calcular la condición de corte óptimo , bajo la cual la máquina es operada económicamente, esto representa una importante parte de la programación NC (Control Numérico) y de la generación del plan de proceso.

La calidad de la optimización de los parámetros del proceso para el control NC y la planeación de operaciones dependen básicamente de la calidad y fiabilidad de los datos de maquinabilidad disponibles en la correspondiente Base de Datos y del modelo aplicado en el cálculo de parámetros. Si la estrategia de control esta basada en costos mínimos/criterios parciales, es necesario seleccionar un proceso optimizado, el cual suministre en corto tiempo y en línea, el valor óptimo de las condiciones relevantes de la máquina.

Solamente estos requisitos son necesarios para poder esperar un eficiente uso y administración de la información en programación NC y en las operaciones de procesos de planeación.

La alta velocidad de acceso y la recepción cruzada de datos, etc., requiere de una selección adecuada del DBMS (Sistema de Administración de la Base de Datos). La decisión aproximada de la selección de DBMS tiene que considerar los principios, destaca la decisión general de la maquinabilidad de la base de datos.

La figura 1.7 muestra la serie de información requerida en la selección de procesos, interviene la serie de materiales {M}, procesos {P}, herramientas mecánicas {MT}, pudiendo implementar los procesos, y el módulo de herramientas {T}. Debido a los hechos eso solo parcialmente puede crear la serie de interacciones entre M, P, T y MT; el número de posibilidades y las condiciones de máquina apropiadas son limitadas. Esto es importante por que se reduce significativamente el número de datos disponibles en la máquina y hace más fácil el manejo de la base de datos.

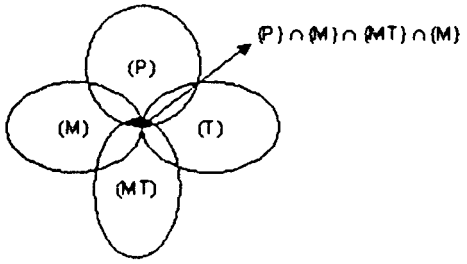


Figura 1.7 Interacción de los módulos {M}, {P}, {MT} y {T}, que participan en la operación de maquinado.

El problema de desarrollo para comprender el mecanismo de la base de datos es por demás complejo debido a que en algún momento incremente el número de material nuevo, usado en la generación de productos. Por cada uno de esos materiales, la maquinabilidad tiene que examinar el transporte de la salida ordenada de los datos reales obtenidos que provienen de la entrada de la base de datos.

Concretamente la maquinabilidad de la base de datos tiene que representar una parte de la BD general en el nivel de producción como esta indicado en la figura 1.8.

Esto quiere decir que estos datos deben estar disponibles no solamente para la programación NC y el departamento de planeación de procesos, pero también en la designación, por las personas de control de calidad, el departamento responsable de compra de materiales, herramientas y maquinas de herramientas, etc.

El objetivo de esta investigación, por consiguiente, esta enfocada también en entrar en la solución de una apropiada DBMS, y la integración de la base de datos para maquinabilidad en la técnica del sistema de información usadas para actividades CAD/ GT/CQA/CAA.

En resumen decimos que para un eficiente uso de herramientas controladas por computadora (es decir programación NC) y procesos de operación de planeación(CAPP). Esto es necesario para desarrollar una base de datos de maquinabilidad como una parte de técnica general y sistemas de información tecnológica con estructuras de datos compatibles. La investigación muestra cual criterio debe ser utilizado para establecer el flujo de información entre varias actividades como CAD, CAGT, CAPP, CAM.

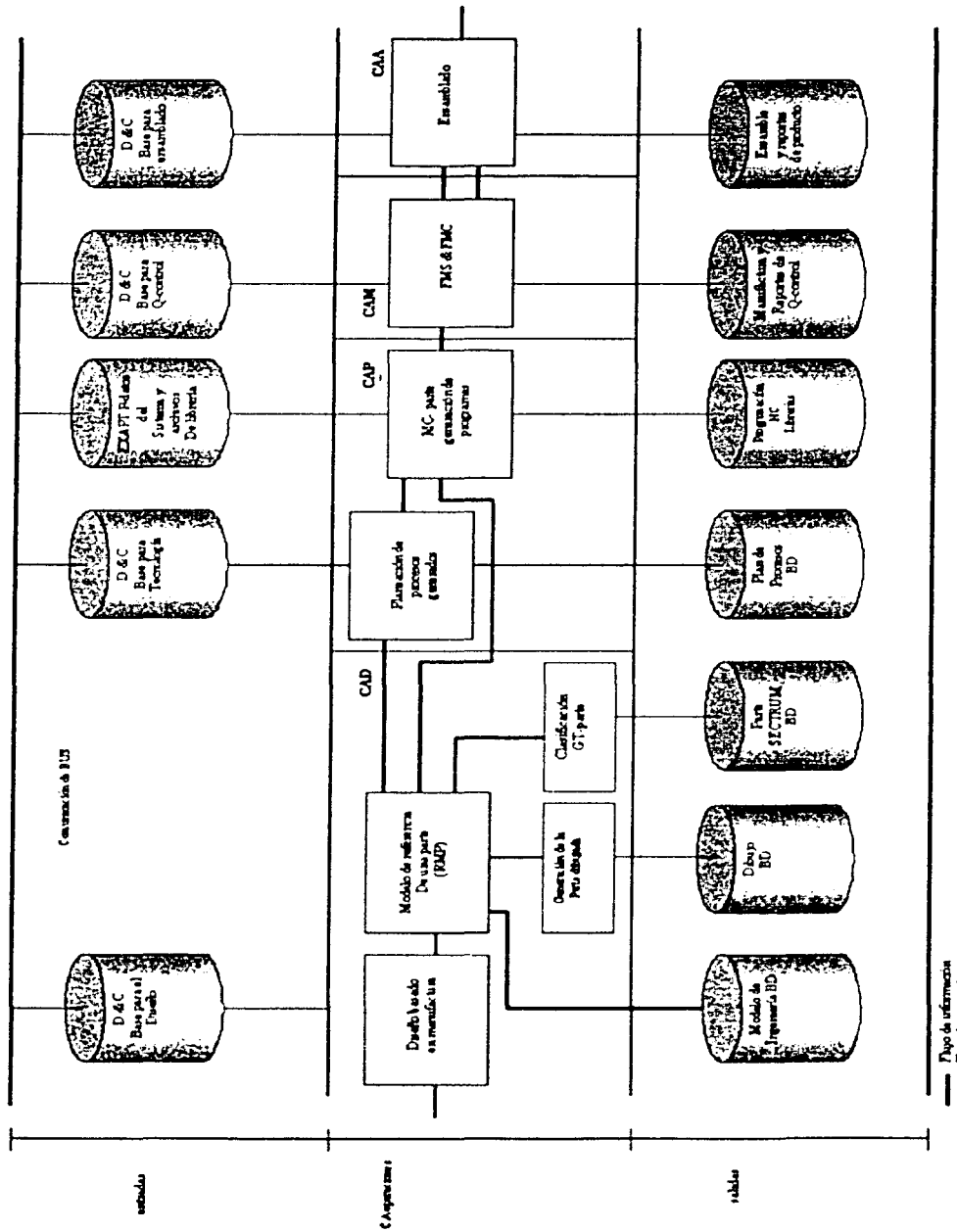


Figura 1.8 Sistema de información CAD/CAGT/CAPP/CAM/CAQ

La base de datos de maquinabilidad es una parte grande de la base de datos del producto entidad relación, conteniendo: los datos generales del producto, modelo de ingeniería de una parte de la base de datos GT y parte del plan de procesos, etc. [Kastelic, et. al., 1993]

1.8 BASES DE DATOS PARA MODELOS DE INFORMACIÓN

1.8.1 Base de datos Relacional

El modelo tradicional de bases de datos es el modelo Entidad-Relación (ER por sus iniciales en inglés - Entity-Relationship)[2], que se divide en tres partes, las cuales se ocupan de la estructura, la integridad y la manipulación de los datos respectivamente. Cada una de las tres partes tiene sus propios términos especiales, y los más importantes en relación con la estructura de los datos se muestran en la figura 1.9 [1].

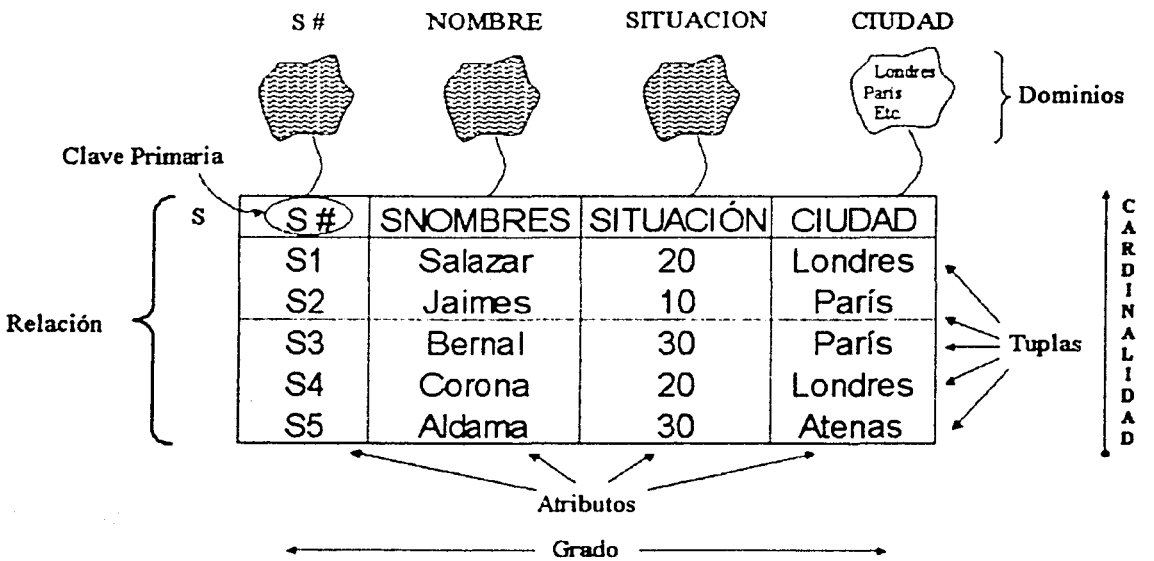


Figura1.9 Estructura de los datos Entidad-Relación.

En este modelo, los datos se organizan en filas (tuplas) y columnas (campos). A su vez, las tuplas se organizan en tablas. Las filas son instancias específicas de una entidad. Las columnas son características de esa entidad. Una tabla separada modelaría la relación entre dos entidades. En esta tabla, cada fila sería una relación, y cada columna sería el identificador único (Unique ID) de las dos entidades en la relación. Usando un Lenguaje de Consulta Estructurado (SQL por sus iniciales en inglés

Structured Query Language) las dos entidades podrían referenciarse en cruzado a través de la tabla relación. La unión, a través de la tabla, provee una relación entre las dos entidades [2].

1.8.2 Base de Datos Orientada a Objetos, BDOO

A finales de los 80's aparecieron las primeras BDOO (Bases de Datos Orientadas a Objetos), bases de datos inteligentes. Soporta el paradigma orientado a objetos almacenando datos y métodos, no sólo datos. Está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Es más segura ya que no permite tener acceso a los datos (objetos); esto debido a que para poder entrar se tiene que hacer por los métodos que haya utilizado el programador [4].

Para entender totalmente las BDOO y como difieren de los modelos relacionales, uno debe entender algunos términos, los cuales fijan las restricciones impuestas por el OODM (por sus iniciales en inglés - Object Oriented Data Model) subyacente, a saber qué es un objeto. Las sutiles diferencias que se aplican a los conceptos de bases de datos y que permiten que la BDOO sea una técnica de modelado mucho más fuerte que otras y en algunos casos más fuerte que el modelo relacional, son importantes. Así, uno necesita un nivel inicial sobre el cual pueda construir la BDOO sobre el OODM. Al nivel de abstracción más alto, de acuerdo con Rob y Coronel, un objeto está modelando una situación o entidad del mundo real al tener una identificación única, propiedades específicas a sí misma, y la habilidad de trabajar en conjunto con objetos tanto de la misma o distinta especificación [Rob, et. al., 1997]. Una entidad en un modelo ER no brinda este comportamiento.

La Identidad del Objeto (OID por sus siglas en inglés - Object IDentity) provee el primer aspecto de un objeto, una identidad única. El sistema asigna una OID, y no es capaz de ser cambiado. De esta forma, dos objetos no pueden ser creados con la misma OID. Esta idea no debería confundirse con una clave primaria, ya que una clave primaria está compuesta de valores ingresados por el usuario, mientras una OID está asignada por el sistema. Al quitar un objeto del sistema, quita una OID del sistema y por definición el sistema nunca asignará a otro objeto esa misma OID [Rob, et. al., 1997].

Un objeto está definido por los atributos que describen la entidad del mundo real que se está modelando. La terminología orientada a objetos usualmente se refiere a estos atributos como variables de instancia. Los atributos pueden ser de cualquiera de los tipos base soportados por un lenguaje de programación. Además de atributos de datos, los objetos usan atributos funcionales o métodos para cumplir con una restricción adicional impuesta por el OODM: interacción [Rob, et. at., 1997]. Los métodos permiten que los datos dentro del objeto sean accedidos y permiten acceder a estados o atributos de otros objetos.

1.8.3 Principios de orientación a objetos

En una base de datos orientada a objetos, cualquier cosa es un objeto y se manipula como tal. Un objeto es una instancia que responde a mensajes activando un método. Los objetos soportan una serie de características de los mismos [3]:

- Se agrupan en tipos denominados clases
- Contienen datos internos que definen su estado actual
- Soportan ocultación de datos
- Pueden heredar propiedades de otros objetos
- Pueden comunicarse con otros objetos enviando o pasando mensajes
- Tienen métodos que definen su comportamiento

Las *clases* son una colección de objetos con propiedades similares, comportamiento común, relaciones comunes a otras clases.

La *instancia* es un objeto con propiedades definidas en su descripción de la clase.

El *mensaje* es una clase que debe tener un método correspondiente. Un mensaje puede ser enviado a un objeto a ejecutar una acción.

El *método* es una lista de instrucciones detalladas que definen cómo responde un objeto a un mensaje en particular.

La *superclase* es la clase que deriva a otra clase.

La *subclase* es la clase derivada de una superclase.

La *liga* expresa compatibilidad de relaciones entre las clases.

Los *objetos* heredan las características de su clase y de todas las clases de nivel superior a la que pertenecen.

Estos principios y técnicas hacen que las BDOO estén adecuadas a aplicaciones que implican tipos de datos complejos, tales como documentos compuestos o de diseño asistidos por computadora que combinan texto, gráficos y hojas de cálculo.

La Bases de Datos proporciona un modo natural de representar las jerarquías que aparecen en los datos complejos. La jerarquía de clases permite a la Base de Datos seguir la pista del tipo de cada objeto en el documento. Finalmente, el mecanismo de mensajes ofrece soporte natural para una interfaz de usuarios gráfica [3].

1.8.4 Tres Enfoques de Construcción de Bases de Datos OO

Las BDOO se pueden construir mediante alguno de los tres enfoques siguientes [4]:

- *El Primero:* se puede utilizar el código actual altamente complejo de los sistemas de administración de las bases de datos, de modo que una BDOO se implante más rápido sin tener que iniciar de cero. Las técnicas orientadas a objetos se pueden utilizar como medios para el diseño sencillo de sistemas complejos. Los sistemas se construyen a partir de componentes ya probados con un formato definido para las solicitudes de las operaciones del componente.

- *El Segundo:* considera a la BDOO como una extensión de la tecnología de las bases de datos por relación. De este modo, las herramientas, técnicas, y vasta experiencia de la tecnología por relación se utilizan para construir un nuevo SGBD(Sistema de Gestión de Bases de Datos). Se pueden añadir apuntadores a las tablas de relación para ligarlas con objetos binarios de gran tamaño (BLOB : Objetos Binarios de Gran Tamaño). La base de datos también debe proporcionar a las aplicaciones clientes un acceso aleatorio y por partes a grandes objetos, con el fin de que sólo sea necesario recuperar a través de la red la parte solicitada de los datos.

- *El Tercero:* reflexiona sobre la arquitectura de los sistemas de bases de datos y produce una nueva arquitectura optimizada, que cumple las necesidades de la tecnología OO. Las compañías como Versant, Objectivity, Itasca, etc. Utilizan esté enfoque y afirman que la tecnología de relación es un subconjunto de una capacidad más general. Además que las BDOO no de relación son aproximadamente dos veces más rápidas que las bases de datos por relación para almacenar y recuperar la información compleja. Por lo tanto, son esenciales en aplicaciones como CAD y permitirían que un depósito CASE fuera una facilidad de tiempo real en vez de una facilidad por lotes.

1.8.5 Ventajas en BDOO

Está su flexibilidad y soporte para el manejo de tipos de datos complejos. Por ejemplo, en una base de datos convencional si una empresa adquiere varios clientes por referencia de clientes servicio, pero la base de datos existente que mantiene la información de clientes y sus compras no tiene un campo para registrar quién proporcionó la referencia, de qué manera fue dicho contacto, o si debe compensarse con una comisión, sería necesario reestructurar la base de datos para añadir este tipo de modificaciones. Por el contrario, en una BDOO, el usuario puede añadir una "subclase" de la clase de clientes para manejar las modificaciones que representan los clientes por referencia.

La subclase heredará todos los atributos, características de la definición original, además se especializará en especificar los nuevos campos que se requieren así como los métodos para manipular solamente estos campos. Naturalmente se generan los

espacios para almacenar la información adicional de los nuevos campos. Esto presenta la ventaja adicional que una BDOO puede ajustarse a usar siempre el espacio de los campos que son necesarios, eliminando espacio desperdiciado en registros con campos que nunca usan.

La segunda ventaja de una BDOO, es que manipula datos complejos en forma rápida y ágilmente. La estructura de la base de datos está dada por referencias (o apuntadores lógicos) entre objetos [4].

1.8.6 Posibles Desventajas en BDOOs

Al considerar la adopción de la tecnología orientada a objetos, la inmadurez del mercado de BDOO constituye una posible fuente de problemas por lo que debe analizarse con detalle la presencia en el mercado del proveedor para adoptar su producto en una línea de producción sustantiva.

El segundo problema es la falta de estándares en la industria orientada a objetos. Sin embargo, el "Grupo Manejador de Objetos" (OMG), es una Organización Internacional de proveedores de sistemas de información y usuarios dedicada a promover estándares para el desarrollo de aplicaciones y sistemas orientados a objetos en ambientes de cómputo en red. La implantación de una nueva tecnología requiere que los usuarios iniciales acepten cierto riesgo. Aquellos que esperan resultados a corto plazo y con un costo reducido quedarán desilusionados. Sin embargo, para aquellos usuarios que planean en futuro el uso de tecnología orientada a objetos, paulatinamente compensará todos los riesgos [4].

1.9 TRABAJOS REALIZADOS CON MODELOS DE MANUFACTURA

Los modelos de los productos surgen del análisis de los datos necesarios para soportar el diseño y la manufactura de productos. Estos modelos son implementados en bases de datos para proveer de información a las aplicaciones computacionales que asisten el diseño concurrente de productos.

Requerimientos de herramientas computacionales que soporten el diseño concurrente de una pieza.

Una herramienta computacional que asista el Diseño concurrente de una pieza, debe satisfacer los siguientes requerimientos [Borja y Morano, 2000]:

1.- Auxilie a la Ingeniería Concurrente permitiendo el trabajo en equipo [Borja, 1997]

2.- Exista una interacción con el diseñador proporcionándole información y proponiéndole soluciones de diseño [Borja, 1997].

3.- Que interprete tolerancias y acabados de superficies por parte del programa [Kunwoo, 1999].

4.- Considere aspectos de funcionalidad del producto y del molde, moldeabilidad del producto, y la maquinabilidad del molde [Canciglieri, et. al., 1998].

5.- Cuente con un protocolo estándar de intercambió de datos con otros sistemas CAD/CAM/CAE [Kunwoo, 1999].

Para satisfacer los requerimientos anteriores, es necesario contar con un MP que capture la información de diseño de la pieza. Para realizar lo anterior Borja y Morano generaron el MP para soportar el diseño concurrente, al cual llamaron MPPIM (Modelo de Producto de la Pieza a Inyectar y de su Molde). Tiene la característica de ser compatible con los Modelos de Información aplicados al diseño de productos metálicos (Borja y Mirón 1999) para poder representar en un futuro productos de ensamble con piezas metálicas y piezas de plástico. [Borja, Morano, 2001]

1.9.1 Diseño para manufactura asistido por computadora

Las necesidades de las empresas de reducir los tiempos y costos requeridos para el desarrollo de productos, aumentando la satisfacción de sus clientes, ha requerido de nuevas herramientas computacionales. Estas nuevas herramientas, pretenden auxiliar actividades concurrentes posibilitando el que diversos software comparta información. Para lograr lo anterior, la forma en que la información de productos y fábricas deben ser almacenadas en computadoras y la forma de usarla para asistir al diseño para manufactura, son investigados. Un ejemplo es el Agente para Torneado, el cual es un software basado en el uso de modelos de información (representaciones en computadoras de productos y fábricas) que asiste el diseño para manufactura de piezas rotacionales [Borja, et. al., 1997].

1.9.2 Sistemas CAE basados en modelos de información

Como se ha mencionado, es necesario integrar las diferente herramientas de cómputo que se usan durante la realización de productos para lograr así un soporte integral que asista eficazmente a quipos de diseño. Esto es posible si todos los programas consultan y almacenan datos empleando el mismo depósito de información o base de datos [Pasad 1996, Fowelr 1995]. Este depósito debe contener todos los datos requeridos por los programas que lo usarán y debe estar estructurado de forma conveniente para la realización de consultas y almacenaje. La estructura y contenido de las bases de datos están determinados por los modelos de información y a los programas que interactúan con ellos se les llama aplicaciones basadas en información o aplicaciones. El software que incluye modelos de información y aplicaciones es conocido como un sistema CAE basado en modelos de información.

1.9.3 Aplicaciones

Las aplicaciones que interactúan con los modelos de información en los sistemas CAE [Borja, et. al., 1997]:

a) consultan información de productos y manufactura en los MP's y MM's del sistema;
b) asisten a diseñadores en funciones específicas tales como modelado geométrico, análisis, optimización, etc.;
c) producen datos de productos, tales como descripciones, características, información de manufactura, etc.;
d) guardan los datos producidos en MP's;
e) suministran esta información al diseñador u otras aplicaciones;
f) interactúan con sistemas CAM, como por ejemplo máquinas de medición por coordenadas, centros de maquinado, etc.

1.9.4 El Agente para Torneado

El agente para torneado es una aplicación original que integra el concepto de sistemas CAE basados en modelos d información, con el uso de principios básicos de diseño para manufactura [Pahl, et. al., 1988, Corbet, et. al.,1991]. El objetivo principal del agente es explorar el uso de MP's y MM's para rediseñar componentes de tal forma que sean fáciles y económicos de producir. El Agente asiste el diseño de piezas rotacionales que se fabrican con procesos de corte de materiales.

El agente para torneado es un sistema basado en reglas que asisten las actividades esenciales del diseño para manufactura [Borja, et. al., 1997]:

a) rediseñar componentes representados en MP's para que éstos sean fáciles de fabricar;
b) verificar si el componente puede ser producido en una fábrica particular representado por un MM;
c) definir la ruta de proceso para componentes (secuencia y descripción de operaciones de manufactura incluyendo recursos necesarios), seleccionando procesos y recursos de MM's, y especificando operaciones de manufactura dentro de los MP's de las partes.

CAPÍTULO II

2. PLANTEAMIENTO DEL PROBLEMA

2.1 INTRODUCCIÓN

En este capítulo se describen los trabajos para asistir al diseño y la manufactura basados en Modelos de Información (MI), tales como los modelos desarrollados en la Universidad Loughborough del Reino Unido por Molina y Borja, así como de los trabajos que se desarrollan en la Facultad de Ingeniería de la UNAM, en el proyecto SADET (Sistema Auxiliar para el Diseño de Ejes de Transmisión).

A demás se plantea el problema del proyecto, los requerimientos y la propuesta de desarrollar un sistema que asista al diseño para la manufactura.

2.2 ANTECEDENTES

2.2.1 Modelo de Manufactura de Molina [1994]

El uso inicial del término Modelo de Manufactura (MM) fue en el contexto de la descripción del modelo de Manufactura Integrada por Computadora (CIM) (capítulo 1, sección 1.5). Para Molina, el término MM es usado para identificar un modelo de información en un sistema CAE (capítulo 1, sección 1.9.2). El cual representa la información relacionada con la capacidad de manufactura de las instalaciones que soporta a la Ingeniería Concurrente (IC). El modelo que propone integra y flexibiliza ambientes computacionales para soportar actividades de diseño y manufactura.

El MM varia dependiendo de las empresas, ya que cada una cuenta con diferentes niveles de información que son requeridos por los diversos departamentos dentro de la empresa, de tal forma que podemos encontrar dos tipos de información:

a) La información específica se refiere a la que depende de la actividad a realizar por parte de un empleado de la empresa.

b) La información genérica de las facilidades de manufactura es necesaria para definir los recursos, procesos y estrategias de manufactura basadas en sus atributos genéricos y clasificaciones de estos en términos de sus características.

El MM contiene información relacionada a tres aspectos importantes de la empresa:

a) Los recursos: se refieren a todos los elementos físicos que se encuentran en una empresa y que permiten la ejecución de la producción como: maquinaria de

producción, herramientas de producción, material, sistemas de almacenamiento, personal.

b) Los procesos: son definidos en general como un cambio en las propiedades de un objeto, incluyendo la geometría, la dureza, etc.

c) Las estrategias: las compañías necesitan de estrategias de decisión y reglas de operación llamadas estrategias de manufactura, éstas describen restricciones impuestas sobre el uso y organización de los recursos y procesos.

A continuación se muestra la estructura principal del Modelo de Manufactura propuesta por Molina, figura 2.1.

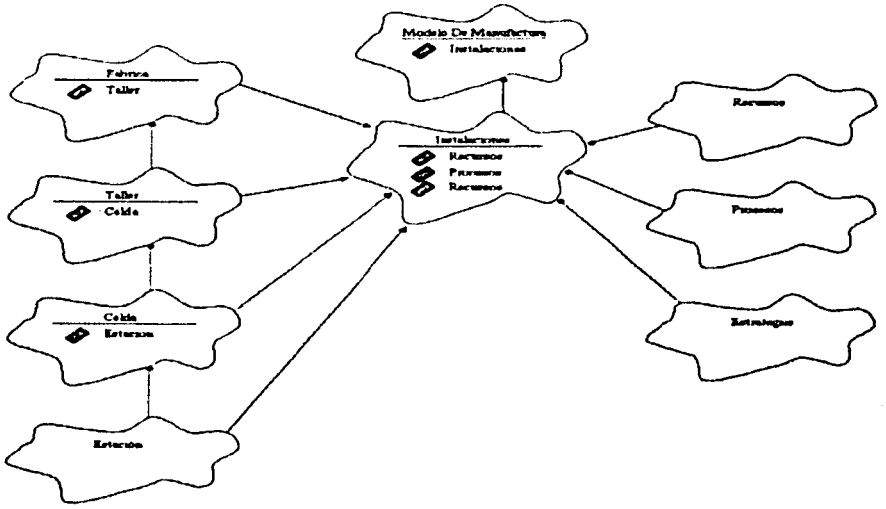


Figura 2.1 Estructura del MM de Molina

La relación entre estos tres aspectos fundamentales describe la capacidad de manufactura de una compañía para tomar decisiones estratégicas y operacionales.

Existen dos tipos de información de capacidades de manufactura en una compañía:

- ✓ Información de Manufactura

La representación de los procesos y recursos a través de los MM, permite representar la infraestructura de la empresa, la cual es necesaria en los sistemas CAE y será usada para soportar los requerimientos del ciclo de vida del producto en una compañía.

- ✓ Información estratégica y operacional.

El segundo tipo de información de capacidades de manufactura se refiere a las estrategias, de cómo manejar los recursos y procesos de la empresa. Las estrategias de manufactura son decisiones que debe tomar la organización y el uso de los recursos y/o procesos. En el MM las estrategias representan como una compañía organiza los recursos y procesos.

2.2.2 Modelo de Manufactura de Borja [1996]

La idea de desarrollar un modelo de manufactura, es el soportar las actividades de la ingeniería concurrente, dentro del ciclo de vida del producto, así es posible incluir elementos que permitan soportar actividades tan particulares como la ingeniería inversa.

El trabajo que Borja propone utiliza el modelo de manufactura, para realizar la ingeniería inversa de un componente considerando diversas fábricas, celdas y talleres de manufactura, de esta manera se pueden obtener substitutos de diversas fábricas procesos y recursos de todas las fábricas y obtener el producto óptimo.

El MM de Borja permite capturar procesos y recursos de manufactura requeridos para la fabricación requerida para la manufactura de operaciones del modelo del producto.

Los requerimientos del MM de Borja, están enfocados en soportar el proceso de ingeniería inversa utilizando el diseño para la manufactura, Borja decide utilizar el modelo de Molina, pero realizando algunos cambios en la representación de la información de manufactura.

Borja define a los procesos de manufactura como un procedimiento para: a) realizar cambios en el diseño de la geometría, tolerancia, en las propiedades del material; b)medición de las propiedades (inspección); c)manejo de las partes. En resumen el término es usado para referir a cualquiera de los procedimientos que transformen la pieza en el producto deseado.

Los recursos de manufactura son todos los elementos físicos dentro de la empresa que son necesarios para la realización del producto.

La definición que propone Borja permite la inclusión de clases como Procesos de Instalación, Procesos de manejo de materiales como procesos de manufactura dentro del MM. Borja selecciona dos tipos de instalación Celda y Estación para su investigación, estas instalaciones pueden ser identificadas con una gran variedad de fábricas.

En el modelo que usa Borja se puede observar que la taxonomía de los Procesos de manufactura que él propone, son diferentes con respecto a la que presenta Molina,

la clase Procesos por arranque por viruta tiene una liga directa con procesos de manufactura y las clases Máquinas herramientas y Herramientas de producción se convirtieron en recursos particulares. Además se puede observar una nueva clase que propone Borja, la cual es Manejo de material.

A continuación se muestra la estructura principal del Modelo de Manufactura propuesta por Borja, figura 2.2.

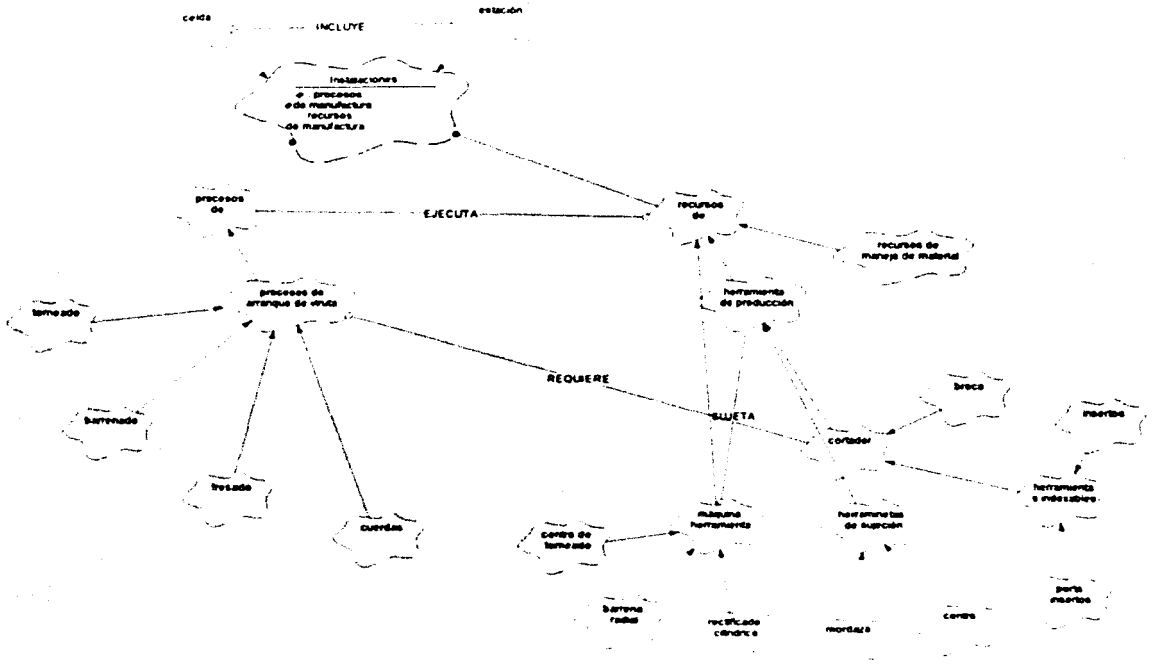
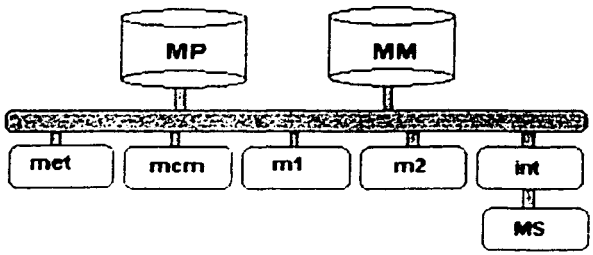


Figura 2.2 Estructura del MM de Borja

2.2.3 SADET

Actualmente en la Facultad de Ingeniería de la UNAM y con la cooperación de CONACYT (proyecto J-27775U) y empresas del área metalmecánica, se lleva a cabo el desarrollo del proyecto SADET (Sistema Auxiliar para el Diseño de Ejes de Transmisión) figura 2.3, basado en modelos del Producto y de Manufactura. Este proyecto pretende diseñar, implementar y aprobar un sistema de cómputo para asistir el diseño para manufactura de ejes de transmisión.



MP	Modelo del Producto	MM	Modelo de Manufactura
met	Modelador de ejes de transmisión	m2	Módulo 2 de diseño de manufactura
mcm	Modelador de Celdas de Manufactura	int	Interfaz a modelador de sólidos
m1	Módulo 1 de diseño para manufactura	MS	Modelador de sólidos

Figura 2.3 Arquitectura de SADET.

Enseguida se describen brevemente cada uno de los módulos de SADET:

- Modelo del Producto (MP). Esquema de información que especifica bases de datos basado en características.

- Modelo de Manufactura (MM). Esquema de información que especifica bases de datos que almacenan las capacidades de manufactura.

- Modelador de ejes de transmisión (met). Este programa crea bases de datos que representan ejes de acuerdo a la especificación del MP.

- Modelador de Celdas de Manufactura (mcm). Este programa crea bases de datos que representan celdas de manufactura de acuerdo a la especificación del MM.

- Módulo 1 de diseño para manufactura (m1). Este programa toma información del MP del sistema y revisa el eje representado considerando criterios de facilidad para manufactura.

- Módulo 2 de diseño de manufactura (m2). Este programa toma información del MP del sistema y compara los requerimientos del eje representado con las capacidades de manufactura de las celdas representados en el MM del sistema.

- Interfaz a modelador de sólidos (int.). Este programa toma información de un eje del MP, la transforma en un formato entendible por un modelador de sólidos y la salva en un archivo. Posteriormente, este archivo es leído por el modelador de sólidos para generar un modelo en tres dimensiones

del eje. Por medio de la interfaz descrita, SADET tendrá comunicación con un modelador de sólidos.

En el proyecto SADET se han desarrollado algunos de estos modelos y se han realizado en Visual C++ y usando una base de datos orientada a objetos Object Store. Los modelos se documentaron con IDEF0(ver apéndice C) y diagramas de clase UML(ver apéndice A).

2.3 PROYECTO SADET

Las actividades del proyecto SADET, son desarrolladas en la Facultad de Ingeniería por un equipo integrado por profesores y alumnos de licenciatura, maestría y doctorado de las áreas de ingeniería mecánica y en computación. Los trabajos desarrollados cumplen con el objetivo propuesto en SADET que es el de diseñar, implementar y probar sistemas de cómputo que asistan al sistema dejes de transmisión.

2.3.1 Modelo de Manufactura de una Celda de Producción, Ayala [2001].

Ayala propone un MM basado en la metodología propuesta por Molina y una herramienta computacional para Modelo de Celdas de Manufactura (MCM), para soportar el diseño para la manufactura e integrarse al proyecto SADET. El objetivo de su trabajo es "definir un modelo computacional para representar las capacidades de manufactura de fábricas específicas". Partiendo de este modelo y de los programas computacionales que interactúan con este modelo, se auxilia las actividades de diseño siguiendo los principios de IC.

Además para integrar los departamentos de diseño y manufactura, el modelo de información asiste al diseño para la manufactura y soporta las actividades relacionadas a las capacidades de manufactura de la empresa y de esta manera desarrollar productos rápidos, económicos, de alta calidad.

El modelo de Ayala usa 3 grupos o clases abstractas que funcionan como una clase principal de clases secundarias. Estas clases principales son:

- ⇨ Instalación: son las estaciones, celdas, SMF y fábricas, y que tienen recursos y procesos de manufactura.

- ⇨ Procesos de Manufactura: son las etapas en donde se realiza una transformación de la materia prima (propiedades mecánicas, geometría, cambios en sus tolerancias), manejo de material, planeación de la producción.

- ⇨ Recursos de Manufactura: son las máquinas, cortadores, accesorios, materiales y los dispositivos para el manejo del material que se utiliza para la producción.

En la figura 2.4 se muestra la estructura principal del MM que usa Ayala en el Modelo de Celdas de Manufactura.

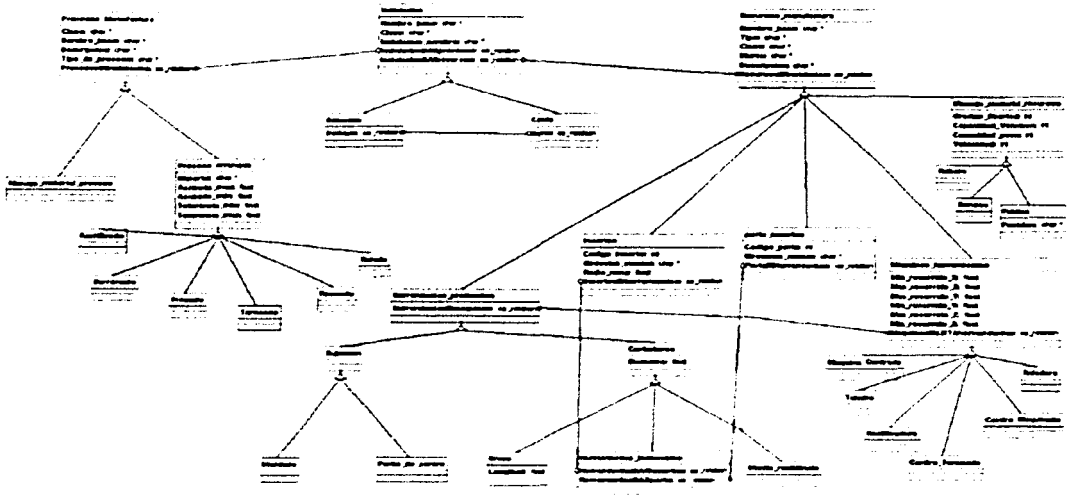


Figura 2.4 Estructura del MM usada por Ayala.

2.3.2 Diseño e Implementación de un Sistema Modelador de Ejes de Transmisión, Mirón [2001].

El trabajo de Mirón especifica la estructura básica del MP y desarrolla el Modelador de Ejes de Transmisión (MET), las cuales son dos de las partes principales del sistema SADET.

El objetivo de Mirón es desarrollar una aplicación (MET) que sea capaz de almacenar, recuperar y editar información de base de datos de MP's.

Mirón propone una modificación al MP de Borja [1997] para crear un modelo simplificado que sea capaz de capturar información necesaria para modelar ejes de transmisión, esta adaptación forma parte de un sistema CAE para asistir el proceso de diseño concurrente.

El desarrollar un modelo de información del producto es para posibilitar la creación de instancias, las cuales representan productos y sus rediseños. El modelo del producto soporta aplicaciones para asistir el diseño para manufactura, o diseño para "X".

El Modelador de Ejes de Transmisión (MET) es una aplicación desarrollada para capturar información de componentes existentes. MET asiste en la documentación de la estructura y geometría de ejes de transmisión y en general de partes rotacionales manufacturadas por procesos de torneado, creando modelos de producto y proporcionando una interfaz para poblar o llenar dichos modelos, también auxilia en el análisis de la información de productos de referencia.

En la figura 2.5 se muestra la estructura principal que utilizó Mirón.

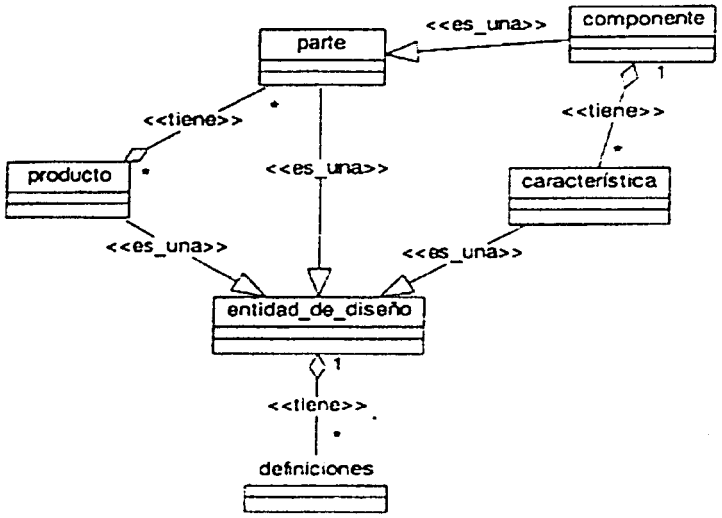


Figura 2.5 Estructura utilizada por Mirón.

2.3.3 Interfaz entre un MP y un Modelador de Sólidos, Vega [2002] .

La Interfaz propuesta por Vega, permite seleccionar un eje que pertenezca al MP, toma la información correspondiente a la estructura de dicho eje y es leída e interpretada por un modelador de sólidos para generar un modelo en tres dimensiones que describa físicamente al eje.

El proyecto permite explorar el potencial de los sistemas basados en modelos de productos y manufactura para asistir el diseño de partes. También da una visión de las tendencias actuales para el desarrollo de los futuros sistemas CAE.

El objetivo de la Interfaz es crear el modelo en tres dimensiones de los ejes de transmisión representados en el Modelo de Producto (MP) de SADET. Como primer paso toma información necesaria de un eje seleccionado del MP, a continuación esta información es llevada a un lenguaje de programación y, finalmente, le indica al

modelador de sólidos que genere un modelo en tres dimensiones de dicho eje. La información mínima para poder hacer esta interfaz debe incluir geometría y dimensiones del eje.

Empleando modelos de información (MP Y MM) no solo se logra analizar las características de los productos y de las herramientas necesarias para su manufactura, la Interfaz ha demostrado que se puede representar gráficamente dicha información. La Interfaz permite la visualización de la información del MP.

2.3.4 Diseño de Moldes de Inyección Asistido por Modelos de Información, Morano [2002].

Morano propone un proyecto que contribuye al desarrollo de herramientas computacionales basados en los modelos de información para auxiliar el diseño de moldes de inyección. La aportación principal de Morano es el establecimiento de los modelos de producto necesarios para llevar a cabo esta tarea.

El proyecto de Morano tiene como objetivo "contribuir al desarrollo de herramientas computacionales basadas en los modelos de información para auxiliar al diseño de moldes de inyección".

La aportación del proyecto de Morano propone una arquitectura, figura 2.6, de las próximas herramientas computacionales para asistir el diseño concurrente de piezas de plástico y sus moldes de inyección.

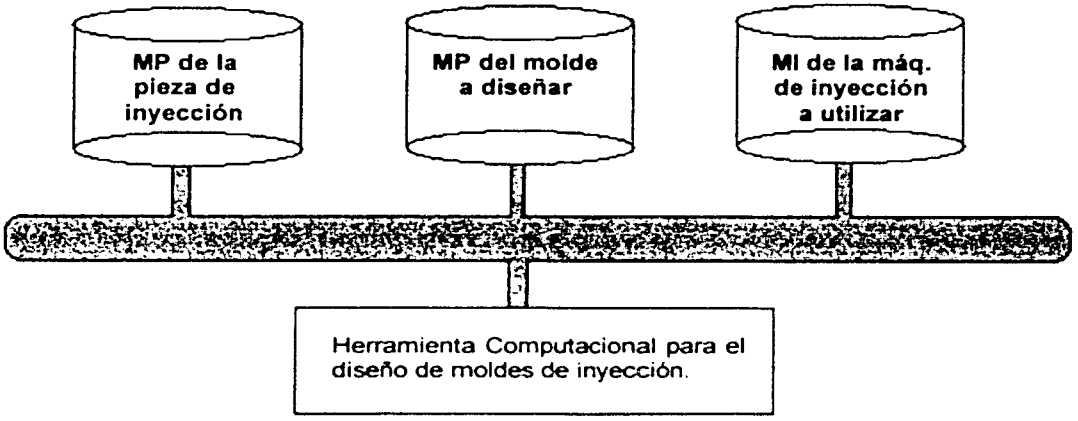


Figura 2.6 Arquitectura de Morano.

También aporta, las extensiones realizadas al MP de Borja (1997), para poder representar piezas de inyección de plástico y sus moldes, el desarrollo completo de un Modelo de Información para una máquina de inyección y la definición de un algoritmo para partir el diseño concurrente de una pieza de inyección de plástico y su molde.

Otras aportaciones de Morano son:

⇨ Identificar la información que debe ser almacenada en el MP de la pieza a inyectar y en el MP del molde a diseñar.

⇨ Definir la forma de representar la información en los MP.

⇨ Desarrollar los MP de la pieza a inyectar y del molde a diseñar.

⇨ Definir la información que debe ser representada en el MI de la máquina de inyección de plástico.

⇨ Desarrollar el MI de una máquina de inyección de plásticos.

⇨ Permitir la compatibilidad de los MP generados en esta investigación, con los MI aplicados al rediseño de productos metálicos [Borja y Mirón 1999]. Esto permitirá representar productos de ensamble con piezas metálicas (ejes de transmisión) y piezas de plástico (piezas moldeadas por inyección).

2.4 DEFINICIÓN DEL PROBLEMA Y OBJETIVO

Después de analizar los diversos trabajos realizados en el proyecto SADET (sección 2.3), podemos notar que cada programa lleva a cabo una tarea que cuenta con aplicaciones que permiten poblar los diversos Modelos de Información. Sin embargo no se cuenta con aplicaciones que integren la información para ayudar en el diseño y manufactura de productos.

Para lograr la integración que mencionamos anteriormente, se propone desarrollar el Sistema de Evaluación de Manufactura de un Producto (SEM), cuyo objetivo principal es la **evaluación de factibilidad de manufactura en componentes** utilizando el paradigma de orientado a objeto y el lenguaje de programación Java. Es una aplicación que utiliza el concepto de sistemas CAE basados en modelos de información.

Como objetivos particulares se propone:

⇨ Estudiar las aplicaciones sobre modelos de información.

⇨ Estudiar las aplicaciones de Diseño para la Manufactura (DFM) asistido por computadora.

⇨ Desarrollar una sistema que asista en el proceso de manufactura.

⇨ Definir un caso de estudio para una empresa del área metalmecánica.

Se debe contar con una herramienta computacional confiable y rápida para obtener información de requerimientos y capacidades de manufactura de un producto, para determinar si es posible o no su realización.

Por lo tanto, como parte del proyecto SADET, este sistema integra las diferentes herramientas de cómputo que se usan durante la realización de productos para lograr así un soporte integral que asista eficazmente a quipos de diseño. Se plantea así un sistema en donde se consulten y almacenen datos empleando el mismo depósito de información, en donde la estructura y contenido de las bases de datos están determinados por los modelos de información.

Con lo mencionado anteriormente se pretende:

g) Consultar información de productos y manufactura en los MP's y MM's del sistema;

h) producir datos de productos, tales como descripciones, características, información de manufactura, etc.;

i) interactúan con sistemas CAM, como por ejemplo máquinas de medición por coordenadas, centros de maquinado, etc.

Los alcances del trabajo de investigación incluye:

✓ Investigación bibliográfica sobre los modelos de información.
Investigación bibliográfica sobre las aplicaciones de DFM asistido por computadora.

✓ Investigar sobre técnicas actuales para DFM y planeación de procesos asistidos por computadora CAPP.

✓ Desarrollar un sistema que asista en el proceso de manufactura.

✓ Definir un caso de estudio para una empresa del área metalmecánica.

2.5 DEFINICIÓN DE REQUERIMIENTOS, ESPECIFICACIONES Y RESTRICCIONES DE SEM

El sistema SEM debe cumplir con los siguientes requerimientos:

- ✓ Utilizar los conceptos de Ingeniería Concurrente.
- ✓ Utilizar MI desarrollados en el proyecto SADET.
- ✓ Utilizar UML como herramienta modeladora del sistema.
- ✓ Utilizar un manejador de bases de datos orientado a objetos.(Object Store, OS)
- ✓ Utilizar Java que es un lenguaje orientado a objetos.

Además el sistema deber ser fácil de aprender y utilizar, para cualquier persona que deseé consultar los MI. Se debe contar con: un ambiente amigable como es el sistema operativo Windows 98 o Windows 2000, el equipo de cómputo con las siguientes especificaciones: procesador desde Pentium II a una velocidad 166Mhz, 64Mb de RAM y con espacio libre en disco duro mínimo de 3Gb.

2.6 MÉTODO DE DESARROLLO.

El método para desarrollar el sistema, figura 2.7, se basa en el propuesto por Mirón [2001] y se muestra a continuación:

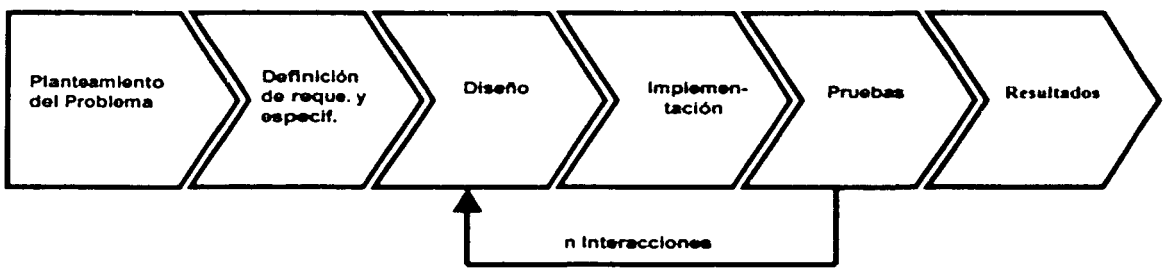


Figura 2.7 Metodología del sistema.

1.- Se define el problema y las necesidades de crear el sistema SEM para la integración de información del proyecto SADET. (sección 2.4)

2.- En este punto se definen los requerimientos, especificaciones y restricciones del sistema. (sección 2.5)

3.- En esta etapa se diseñará el sistema utilizando UML como herramienta modeladora.

4.- La implementación del sistema se realizará utilizando el lenguaje de programación Java, además se utilizara un manejador de bases de datos orientado a objetos que es Object Store.

5.- Se propone un caso de estudio para probar la integración del sistema al proyecto SADET.

CAPÍTULO III

3. DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

En este capítulo se describe la estructura del MP y MM propuestos por Mirón y Ayala respectivamente. También veremos como el sistema propuesto será diseñado con UML, que es una herramienta de modelado de sistemas que permite el análisis y diseño de proyectos de software(ver apéndice A).

3.2 ESTRUCTURA DEL MM, AYALA [2001].

La estructura principal (ver figura 2.4, capítulo 2, sección 2.3.1)del modelo de manufactura propuesto por Ayala, se basa en tres grupos de clases significativos, estos son: *Procesos de manufactura*, *Recursos de manufactura,* e *Instalaciones.* Estos grupos son clases abstractas que no generan instancias por si mismas pero que funcionan como una clase principal de clases secundarias. La asociación entre clases se da utilizando verbos.

El primer grupo es Instalación, el cual es la clase abstracta a partir de la cual se pueden crear los objetos *Estación* y *Celda*, además se establece la asociación Incluye, lo que nos dice que una celda está formada por estaciones.

El segundo grupo es el de *Recursos* de manufactura, está constituido básicamente por todos los elementos físicos que se encuentran en una instalación y que permiten la manufactura de productos.

El último grupo es el de *Procesos* de manufactura, el cual se asocia externamente con *Procesos_Manufactura* porque requiere de *Herramientas_producción* e *Instalación* tiene *Procesos*.

3.3 ESTRUCTURA DEL MP, MIRÓN [2001].

La estructura del MP que utiliza Mirón se basa es el paradigma orientado a objetos y provee el contexto para las estructuras de información de las categorías de información definidas por Borja [1997] que son:

1.- Estructura del Producto.
2.- Especificaciones.
3.- Conceptos.
4.- Características de Diseño.
5.- Información de Manufactura
6.- Propiedades.

Mirón solo utiliza dos categorías de información, que son: estructura del producto y características de diseño, estas representan la necesidad de información para soportar ingeniería inversa considerando el método de manejo de información, y en particular el uso combinado de un MP y un modelo de MM.

El MP y su estructura se desarrollaron usando el UML como una herramienta de modelado semántico para el diseño y documentación de los esquemas. La figura 2.5 de la sección 2.3.3 del CAPÍTULO 2, muestra el diagrama de clases de la estructura principal del MP y las relaciones entre las clases.

La estructura principal del MP se centra alrededor de la estructura física del producto y sobre las etapas de su ciclo de vida. Para el caso del MP simplificado solo se basa en una de las tres etapas del ciclo de vida del producto, esta es el producto como es diseñado y planeado para manufactura. Dicha etapa será modelada por la clase definiciones.

3.4 DISEÑO DEL SISTEMA SEM EN UML

3.4.1 Clases

Para iniciar el modelado del sistema se analizó el problema (capítulo 2, sección 2.4) y se encontraron las necesidades del sistema. El sistema permitirá la integración de las bases de datos de los MP y MM. El sistema será utilizado por cualquier usuario que desee información de factibilidad de manufactura de un producto. El usuario interactuará con el sistema a través de un menú principal y de diálogos, para abrir y leer las bases de datos del MP y MM.

"Una **clase** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semánticas. Gráficamente, una clase es representada con un rectángulo, usualmente incluyendo su nombre, atributos y operaciones" [1]. De esta forma obtenemos nuestras primeras clases que son todos los sustantivos que se encuentran en el análisis del problema, estos serian: el sistema, MP, MM usuario, menú y diálogo, estas clases se muestran a continuación en la figura 3.1.

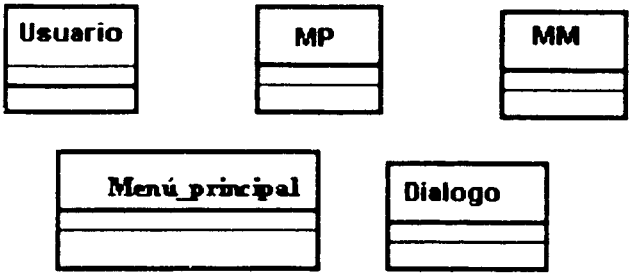


Figura 3.1 Clases del sistema SEM

Otras clases que se tienen son: una clase donde se puedan encontrar los diferentes tipos de bases de datos que vamos a utilizar, una más seria una clase que nos permita conocer la estructura de las clases de las bases de datos, por lo tanto también necesitamos las clases de estructura de clase de cada una de las bases de MP y MM. Estas nuevas clases serian las de la figura 3.2.

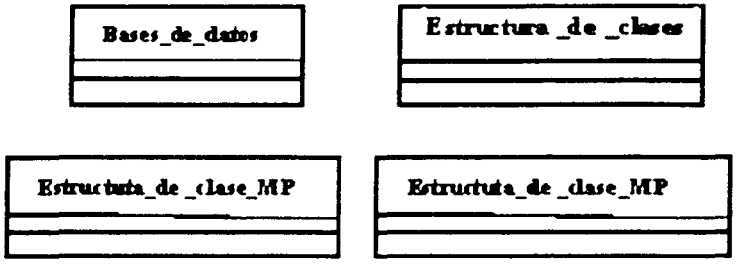


Figura 3.2 Nuevas clases del sistema SEM

A continuación se describe brevemente cada una de las clases, basado en el paradigma orientado a objetos:

- **Diálogo** es una clase base para la definición común que afecta a las clases que derivan de ella en jerarquía.
- Una de esas clases es **Menú_Principal**, esta es creada para soportar la interfaz gráfica que utilizara el usuario.
- **Estructura_de_clases**, es la clase interfaz entre la clase menú y la clase base de datos para la interpretación de la estructura de datos de las instancias de las clases Mm y Mp.
- **Estructura_de_clase_MP,** esta clase es hija de la clase **Estructura_de_clases** e interpreta la estructura de la base del MP.
- **Estructura_de_clase_MM** esta clase es hija de la clase **Estructura_de_clases** e interpreta la estructura de la base del MM.
- **Base_de_datos**, es una clase abstracta que nos ayuda a definir diferentes tipos de bases de datos.
- **MP,** esta es hija de la clase **Bases_de_datos** y es la que representa al MP.
- **MM,** esta es hija de la clase **Bases_de_datos** y es la que representa al MM.

- **Usuario**, es una clase abstracta. Es cualquier persona que haga uso del sistema.

En la clase Bases_de_datos se tendrían como hijas a las clases MP y MM, y en la clase de Estructura_de_clases se tienen como hijas a las clases Estructura_de_clase_MP y Estructura_de_clase_MM. En nuestro sistema la clase Menú_principal es un diálogo con el que el usuario interactúa, por lo tanto la clase Menú_principal hija de la clase Diálogo. Para nuestro sistema cambiaremos el nombre de la clase Diálogo por J_Diálogo para hacer referencia de que es la clase de diálogo de Java.

En el diagrama de clases siguiente de la figura 3.3, nos permite estructura de forma más clara el sistema SEM, en el se encuentran las clases definidas anteriormente.

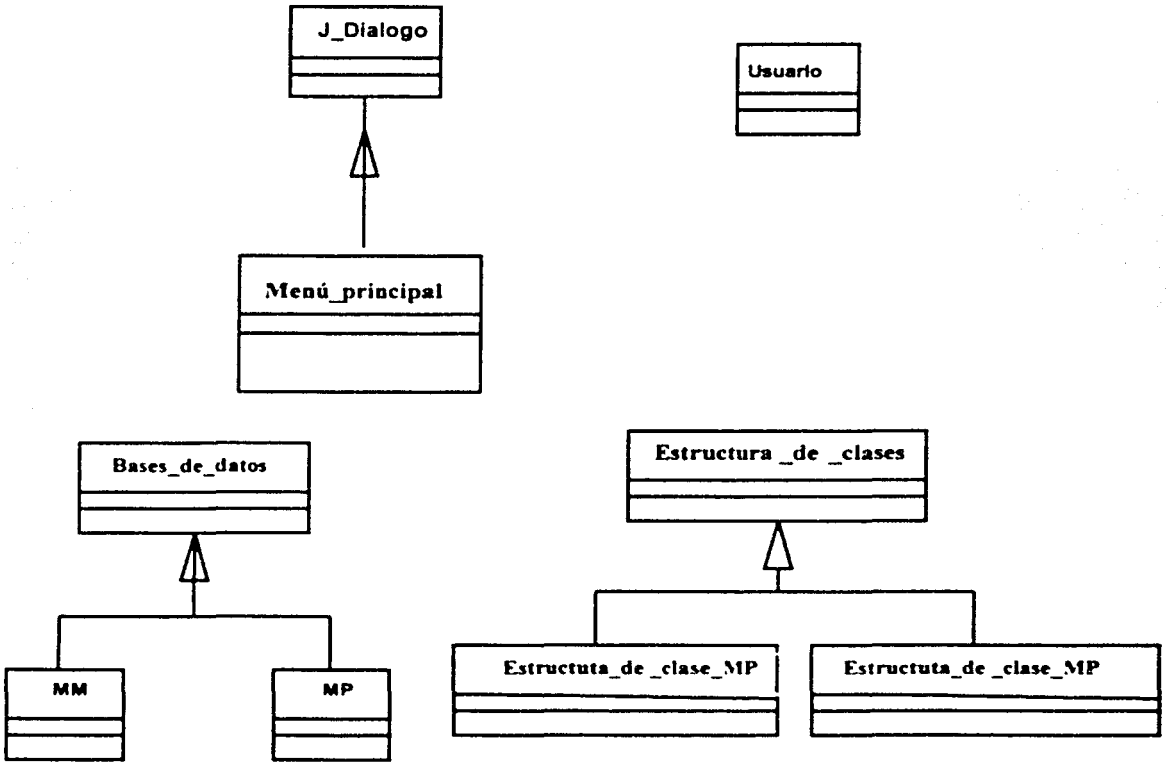


Figura 3.3 Diagrama de Clases del Sistema.

3.4.2 Caso de uso

El sistema requiere de un caso de uso que es "una descripción de un conjunto de secuencias de acciones que usado para estructurar los elementos de comportamiento de un modelo, se representa con una elipse de líneas sólidas" [1]. Un caso de uso es la evaluación que se desea hacer, por ello este es el caso de uso que se requiere para el sistema y se muestra en la figura 3.4.

Figura 3.4 Casos de Uso del sistema.

Ahora definimos a los actores, "el actor es una entidad externa al sistema que se modela y que puede interactuar con él" [Schmuller, 1999]; uno de nuestros actores es el Usuario y otros actores son nuestras bases de datos del Modelo del Producto y Modelo de Manufactura, estos se muestran en la figura 3.5.

Usuario Modelo del Producto Modelo del Producto

Figura 3.5 Actores del Sistema

Con los elementos anteriores se creo un diagrama de caso de uso, "que muestra un conjunto de casos de uso, actores y sus relaciones, sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo"[1]. Hay que aclarar que el caso de uso debe ser fácil de entender por el usuario final. [Schmuller, 1999].

Para realizar la evaluación de factibilidad de manufactura de un producto, se deben seguir los siguientes pasos:

1.- El usuario para consultar, como primer instancia debe abrir alguna de las bases de datos, ya sea del MM o MP. Una vez abierta la base, se realiza la lectura de datos deseados.

2.- Después de hacer la primera consulta, el usuario debe abrir la otra base (MM o MP) para realizar la segunda lectura.

3.- El sistema muestra al usuario información a través de diálogos. Si abre la base del MP se muestran las características primarias y secundarias del producto; en el MM se muestra el nombre de la instalación existente, sus procesos y sus recursos.

4.- Después de mostrar esta información, el sistema internamente hace una comparación de los datos obtenidos de las bases y le entrega al usuario una lista de resultados de la evaluación.

Cabe señalar que antes de abrir las bases, el sistema comprueba la estructura de clase de las bases MP y MM. En la figura 3.6 se muestra el diagrama de Caso de Uso, que es un escenario donde podemos ver como interactúan los actores con el caso de uso.

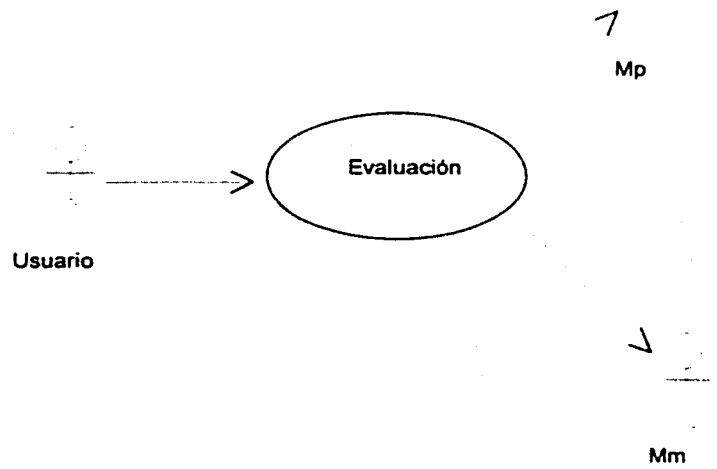


Figura 3.6 Diagrama de Caso de Uso del Sistema.

En la siguiente tabla llamada Curso Normal de los Eventos para el Caso de Uso, se muestran los pasos que sigue un actor y la respuesta del sistema dentro del Caso de Uso.

Acciones de los actores	Respuesta del sistema
1.- El usuario se abre menú principal y da seleccionar base (MM o MP).	
	2.- El sistema abre un diálogo para seleccionar la base (MM o MP)
3.- El usuario selecciona primero la base (MM o MP) que desea abrir.	
	4.- El sistema comprueba la estructura de clase de la base de datos (MM o MP) abierta.
5.- La base de datos (MM o MP) es abierta.	
	6.- El sistema muestra un diálogo para indicar la información que existe en la base de datos.
7.- El usuario obtiene la información deseada de la primer base que abrió.	
	8.- El sistema guarda temporalmente la información obtenida.
9.- El usuario selecciona la segunda base (MM o MP) que desea abrir.	
	10.- El sistema comprueba la estructura de clase de la base de datos (MM o MP) abierta.
11.- La base de datos (MM o MP) es abierta.	
	12.- El sistema muestra un diálogo para indicar la información que existe en la base de datos.
13.- El usuario obtiene la información deseada de la segunda base que abrió.	
	14.- El sistema guarda temporalmente la información obtenida.
	15.- El sistema hace una comparación de los datos de ambas bases.
	16.- El sistema entrega una lista de resultados de la evaluación al usuario en un diálogo.

Tabla 1. Curso Normal de los Eventos para el Caso de Uso.

3.4.3 Diagramas de secuencia

En la figura 3.7 y 3.8 se muestran los Diagramas de Secuencia del Modelo del Producto y del Modelo de Manufactura respectivamente, donde se muestra la secuencia de pasos de la interacción que tendrá el usuario con el sistema y las bases de datos que seguirá el sistema. Un diagrama de secuencia representa una forma de indicar el periodo durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento[1].

En el Diagrama de Secuencia del Modelo del Producto, figura 3.7, se explica el funcionamiento paso a paso del sistema.

1) El usuario abre la opción de archivo que se encuentra en el menú principal, donde se abre las opciones de manejo de archivos.

2) Aquí el usuario selecciona abrir base donde se abre un submenú.

3) Posteriormente selecciona Modelo del Producto abriéndose un diálogo, en este se selecciona la base de datos que quiere consultar el usuario.

4) El sistema se regresa a menú principal.

5) De forma interna se comprueba la estructura de la base de datos y la abre.

6) A continuación se abre un diálogo para seleccionar el producto o componente a consultar.

7) Ya seleccionado nuevamente el sistema de forma interna lee la estructura, lee el objeto, regresa información a menú principal y la guarda temporalmente.

8) Se abre un nuevo diálogo para seleccionar la base de datos del Modelo de Manufactura.

9) En este se selecciona la base de datos que quiere consultar el usuario.

10) El sistema se regresa a menú principal, de forma interna se comprueba la estructura de la base de datos y la abre.

11) A continuación se abre un diálogo para seleccionar la instalación, los recursos o los procesos a consultar.

12) Ya seleccionado nuevamente el sistema de forma interna lee la estructura, lee el objeto, regresa información a menú principal y la guarda temporalmente.

13) Por ultimo ya obtenida la información de ambas bases, internamente se comparan los datos, para mostrar al usuario finalmente si es factible o no la manufactura del producto o componente.

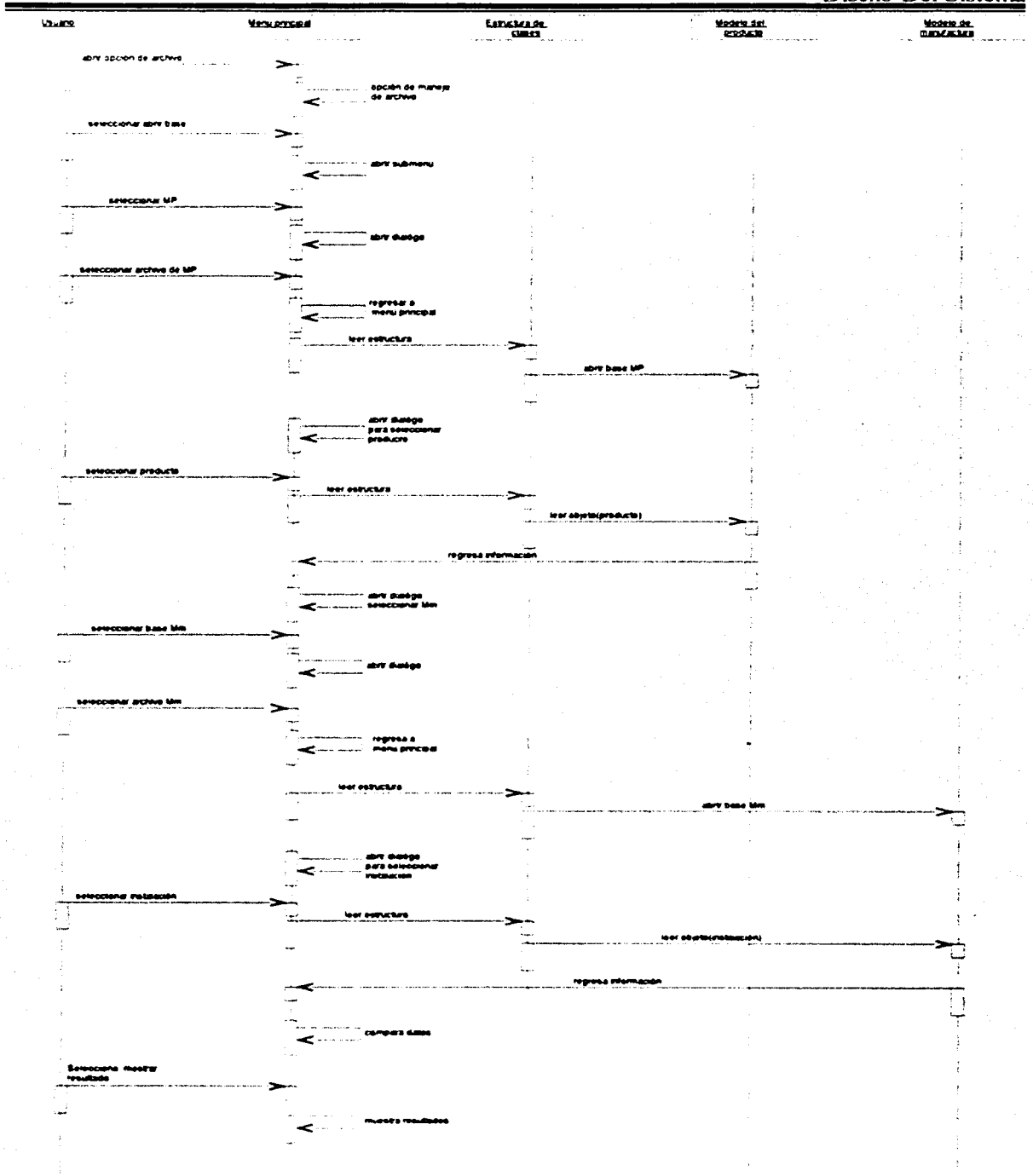


Figura 3.7 Diagrama de Secuencia del Modelo del Producto.

En el Diagrama de Secuencia del Modelo de Manufactura, figura 3.8, se explica el funcionamiento paso a paso del sistema.

1) El usuario abre la opción de archivo que se encuentra en el menú principal, donde se abre las opciones de manejo de archivos.

2) Aquí el usuario selecciona abrir base donde se abre un submenú.

3) Posteriormente selecciona Modelo de Manufactura abriéndose un diálogo.

4) En este se selecciona la base de datos que quiere consultar el usuario.

5) El sistema se regresa a menú principal, de forma interna se comprueba la estructura de la base de datos y la abre.

6) A continuación se abre un diálogo para seleccionar la instalación, los recursos o los procesos a consultar.

7) Ya seleccionado nuevamente el sistema de forma interna lee la estructura, lee el objeto, regresa información a menú principal y la guarda temporalmente.

8) Se abre un nuevo diálogo para seleccionar la base de datos del Modelo del Producto.

9) En este se selecciona la base de datos que quiere consultar el usuario.

10) El sistema se regresa a menú principal, de forma interna se comprueba la estructura de la base de datos y la abre.

11) A continuación se abre un diálogo para seleccionar el producto o componente a consultar.

12) Ya seleccionado nuevamente el sistema de forma interna lee la estructura, lee el objeto, regresa información a menú principal y la guarda temporalmente.

13) Por ultimo ya obtenida la información de ambas bases, internamente se comparan los datos, para mostrar al usuario finalmente si es factible o no la manufactura del producto o componente.

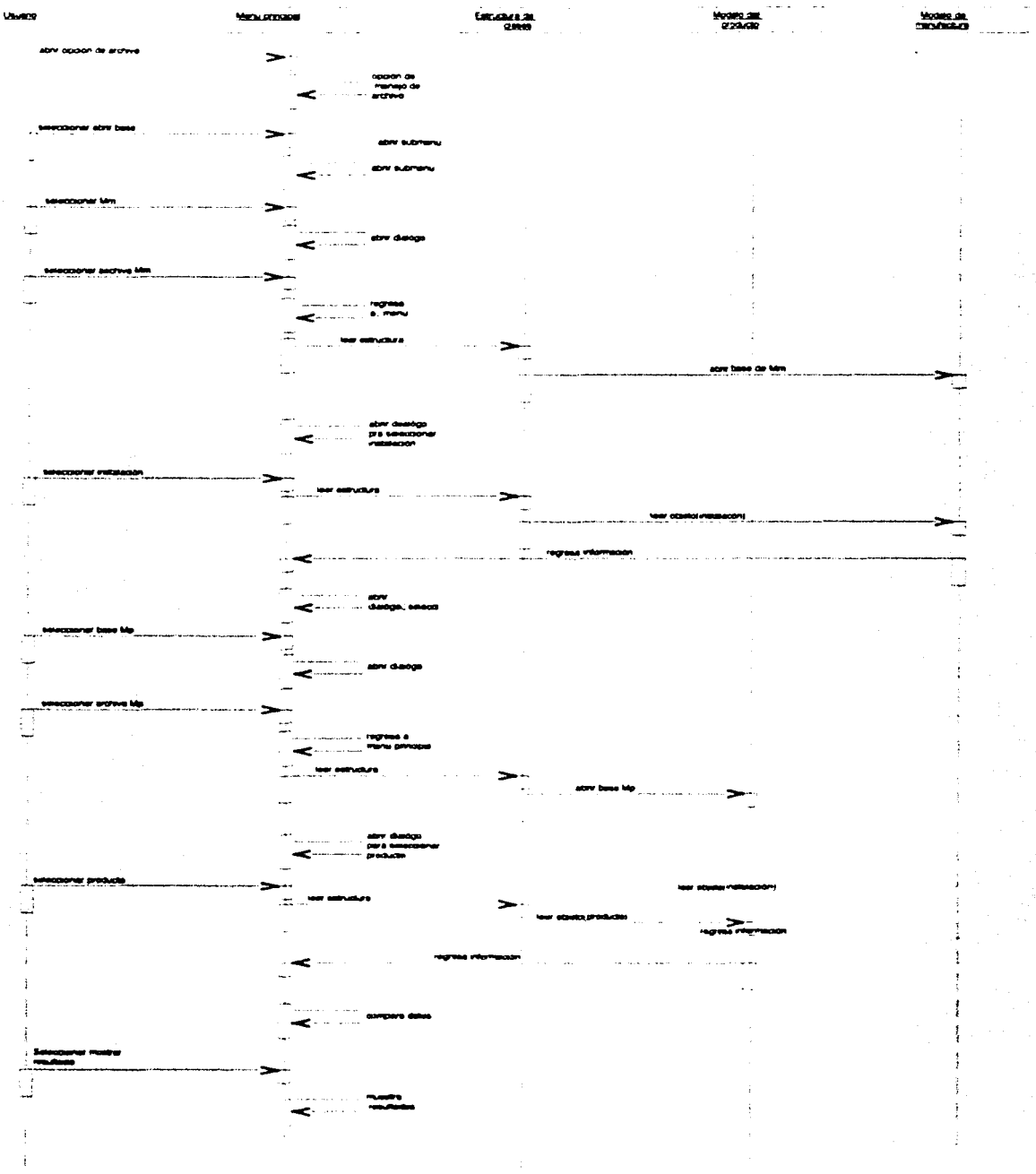


Figura 3.8 Diagrama de Secuencia del Modelo de Manufactura.

3.5 IMPLEMENTACIÓN DEL SISTEMA SEM

3.5.1 Herramientas de Programación

El Sistema SEM es implementado en el lenguaje de programación Orientado a Objetos Java. Cabe mencionarse que aplicaciones similares, se han realizado, más estas han sido hechas con C++(SADET). Por lo que se presenta la interrogante ¿Por qué Java? y no continuar con la misma línea de programación con C++.

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje, es decir, no existen punteros, las cadenas de caracteres son objetos y la administración de memoria es automática. Java trabaja con datos como objetos y con interfaces a esos objetos. Soporta características propias del paradigma de la orientación a objetos: encapsulamiento, herencia y polimorfismo, etc. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. De esta forma, Java construye y facilita la creación de entornos de desarrollo-aplicaciones de interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac. Por lo que el Sistema SEM puede ejecutarse en cualquier plataforma teniendo el mismo comportamiento.

Otro punto importante para Java es tener una interfaz directa con el manejador de Bases de Datos: Object Store facilitando el almacenamiento y recuperación de la información de las bases de Datos que necesita el sistema SEM para cumplir con sus objetivos previamente establecidos.

3.5.2 Implementación

De esta forma el código generado en Java para la implementación del sistema SEM se compone de 2 archivos principales: *"Modulo12"* y *"Dialogos1"*. Además, se hace uso de las clases que definen las estructuras del Modelo del Producto y de Manufactura (clases realizadas en Java). Dichas clases se encuentran almacenadas en paquetes (packages), los cuales son fácilmente de acceder desde *"Modulo12"*, por las características de polimorfismo y encapsulamiento (ver Apéndice B), al insertar dentro de las primeras líneas las siguientes sentencias utilizadas por Object Store (ver Apéndice D):

```
import com.odi.tutorial.tesis_MP.*;
import com.odi.tutorial.tesis_dialogo.*;
```

Estas sentencias importan las clases del paquete *tesis_MP* que se van a utilizar, así como las del paquete *tesis_dialogo* que corresponden al Modelo del Producto y al Modelo de Manufactura respectivamente.

"Modulo12" sólo permite la lectura y acceso a los datos tanto del Modelo del Producto como del Modelo de Manufactura a través de transacciones (operaciones propias de Object Store que permiten leer, modificar o agregar información en la base de datos). Un ejemplo de estas transacciones se muestra en la figura 3.9 (ver Apéndice D).

```
/// Método que guarda los procesos al hacer uso de una transacción
para el Modelo del Producto.

public Object[] guarda_procesos()
{
  trans= Transaction.begin(ObjectStore.READONLY);
   rootProce = (Set) db1.getRoot("Procesos");
   arre = rootProce.toArray();
  trans.commit(ObjectStore.RETAIN_HOLLOW);
return arre;
}
/// Método que permite poner en pantalla la informacipon correspondien al
Modelo de Manufactura al hacer uso de una transacción.

public void getInstaNames()
{ f1=true;
  trans= Transaction.begin(ObjectStore.READONLY);
  rootInsta = (Map)db.getRoot("Instalacion");
  rootProce = (Set) db.getRoot("Procesos");
 rootRecur = (Set) db.getRoot("Recursos");

        ...
trans.commit(ObjectStore.RETAIN_HOLLOW);
}
```

Figura 3.9 Uso de Transacciones.

Es importante mencionar que no se utilizan transacciones para modificar datos de las bases a utilizar, debido a que esto solo que puede hacer desde las aplicaciones que las generaron. *"Modulo12"* (ver Apéndice D) accesa y guarda temporalmente la información necesaria para llevar a cabo la evaluación. De las funciones que pueden logran esta tarea podemos mencionar: *"guarda_procesos"*, *"guarda_recursos"*, *"Compara"*, *"busca_procesos(int pr)"*, etc.

Mientras que *"Dialogos1"* tiene la tarea de crear los diálogos necesarios que permiten mostrar al usuario la información de las bases que se consultan así como de los resultados proporcionados por el sistema SEM una vez que ha realizado la evaluación correspondiente.

Para poder ver claramente esto, en el siguiente capítulo se observará el comportamiento del sistema SEM mediante un caso de estudio.

3.5.3 Descripción del Sistema SEM

Al ejecutarse el programa SEM aparece la pantalla principal (Fig. 3.10), donde se muestra el menú principal activado y diversos elementos desactivados.

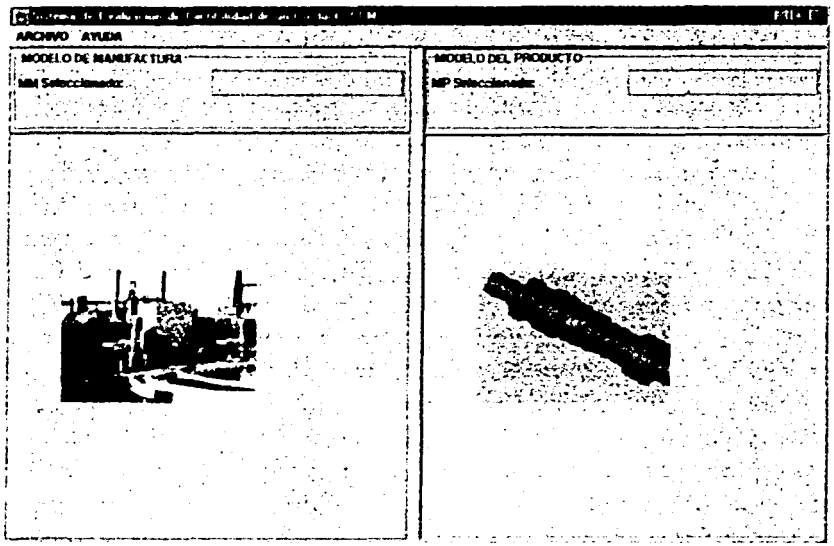


Figura 3.10 Pantalla principal

Esta pantalla contiene los siguientes elementos(Fig. 3.11):

- Un menú con las siguientes opciones:
  - ARCHIVO
    - Seleccionar Base
      - Modelo de Manufactura
      - Modelo del Producto

* Salir

- AYUDA

  * Acerca de..

❖ Una caja de texto donde se puede observar el nombre de la base de datos MP una vez que se haya abierto esta.

❖ Otra caja de texto donde se puede observar el nombre de la base de datos MM una vez que se haya abierto esta.

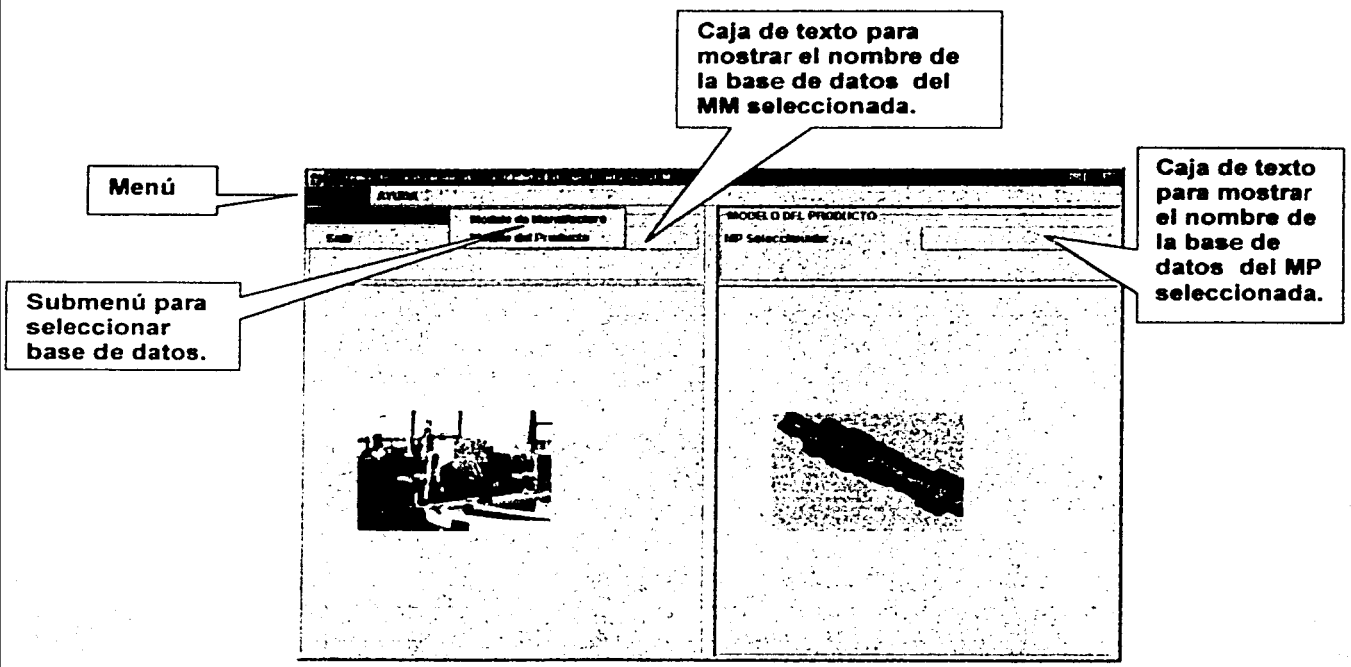


Figura 3.11 Descripción de la pantalla principal.

CAPÍTULO IV

4. CASO DE ESTUDIO

4.1 INTRODUCCIÓN

Para probar el sistema SEM y conocer el comportamiento de este, se propone la evaluación del MM y MP propuestos en el proyecto SADET.

4.2 OBJETIVOS DEL CASO DE ESTUDIO

El propósito del caso de estudio es validar y probar el sistema propuesto SEM basado en el proyecto SADET. Donde se plantean los siguientes objetivos:

- ✓ Probar y evaluar la utilidad y funcionalidad de SEM.
- ✓ Experimentar con información de un caso real.

4.3 MODELOS DE INFORMACIÓN PARA EL CASO DE ESTUDIO

Para el caso de estudio propuesto se requiere de los Modelos de Información, el MM es el propuesto por Ayala [2001] y poblado con "mcm" y para el MP propuesto por Mirón [2001] poblado con "met" (ver capitulo 3, sección 3.2 y 3.3).

Estos dos Modelos de Información servirán para realizar la evaluación de factibilidad de manufactura del producto; esta se realizará a través de SEM. Los datos que serán evaluados son los procesos y recursos existentes en MM y las características con que cuenta un componente en MP, posteriormente son comparados, dando como resultado la factibilidad o no factibilidad de manufactura del componente.

Dentro de la evaluación el sistema debe verificar que las bases cuenten con información consistente, es decir, que contenga los datos necesarios para realizar la comparación de estos. Si alguna base no contara con datos suficientes o inexistentes, la comparación no se realiza y por tanto el resultado será la no factibilidad de manufactura del componente.

El MM a utilizar modela una instalación llamada "Celda de Manufactura" esta consta de un centro de maquinado vertical VMC-100, un centro de torneado TURN120P, cuatro robots Escorbot, un sistema de alimentación por medio de banda, 20 insertos, 20 portainsertos, 10 brocas. Por lo que esta información es lo que se representa en el MM. La celda esta representada en la figura 4.1, se encuentra en los talleres de Ingeniería Mecánica de la Facultad.

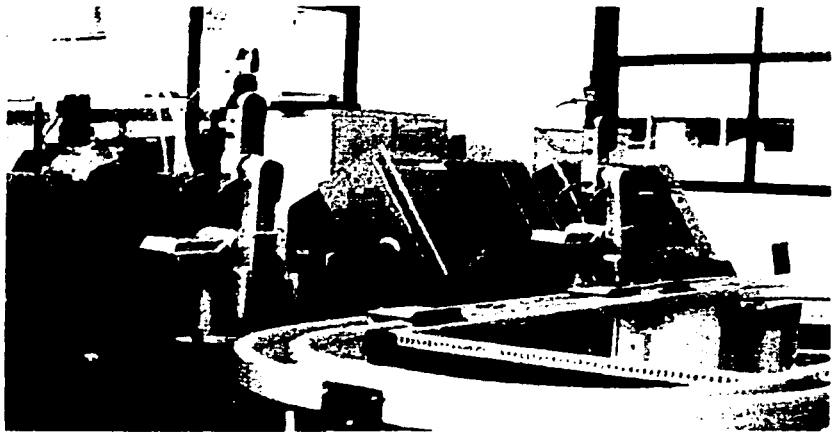


Figura 4.1 Celda de Manufactura.

Las características de los componentes del MP a utilizar son las siguientes: cuentan con sección transversal cilíndrica, las cuales pueden incluir características prismáticas como por ejemplo ranuras, manufacturadas por un proceso de maquinado, en particular por torneado así como información disponible sobre su diseño y manufactura. El componente de estudio seleccionado es un eje de transmisión, dicho eje se muestra en la figura 4.2.

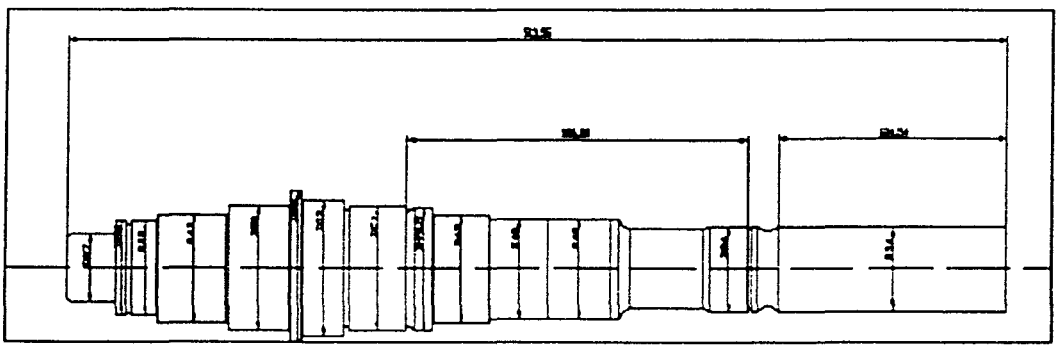


Figura 4.2 Eje para el Caso de Estudio del MP.

4.4 EVALUACIÓN DE LA CELDA Y EL EJE DE TRANSMISIÓN

Al ejecutarse el programa SEM aparece la pantalla principal (Fig. 4.3), donde se muestra el menú principal activado y diversos elementos desactivados.

Para iniciar la evaluación del caso de estudio, se debe consultar una base de datos (MP o MM) (Fig. 4.3), se selecciona del menú "Archivo", enseguida se posisiona el cursor en "Seleccionar Base", aparece un submenú donde es posible seleccionar "Modelo de Manufactura" o "Modelo del Producto".

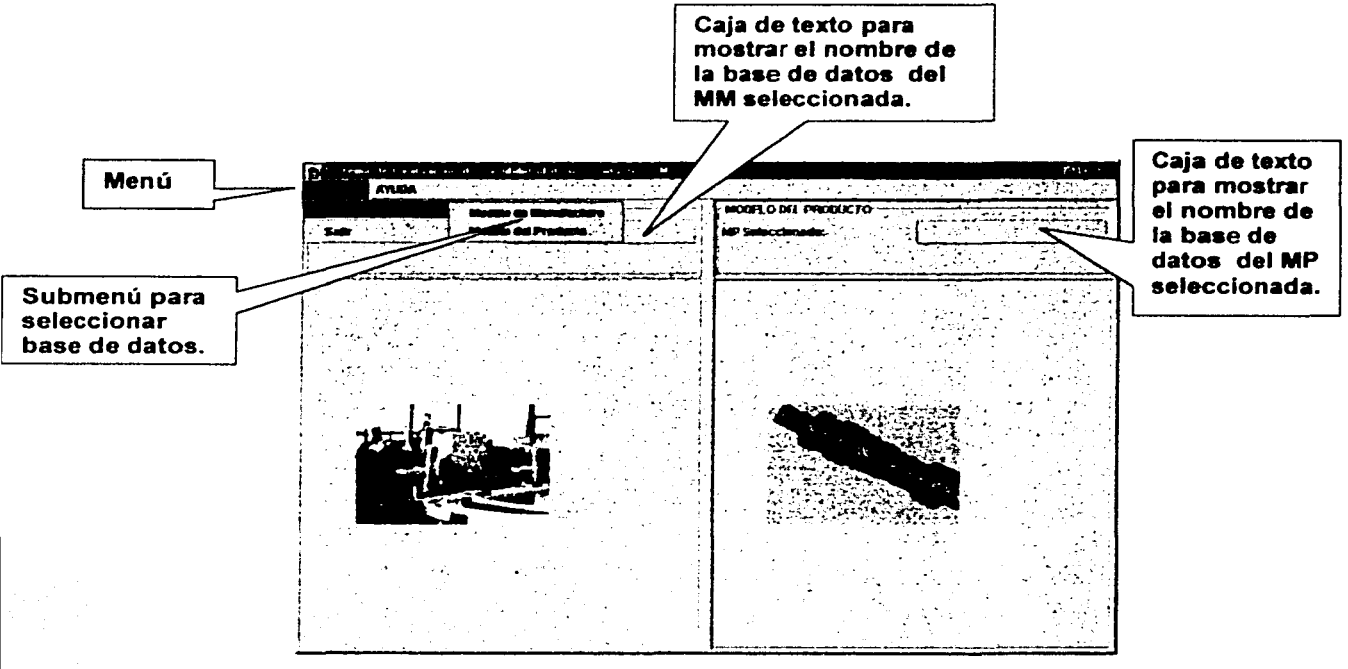


Figura 4.3 Descripción de la pantalla principal.

Enseguida se muestra el diálogo para seleccionar la base de datos deseada, en este diálogo se puede navegar a través del disco duro de la máquina. Para este caso se busca la *"Celda de Manufactura"* (Fig. 4.4). Es posible selecccionar cualquiera de las bases, MM o MP, no es necesario seguir un orden específico. Una vez seleccionada la base de MM, *"Celda de Manufactura"*, la aplicación muestra el nombre de ésta en el cuadro de texto del lado izquierdo de la pantalla principal, activándose el botón con la leyenda *"Ver Instalación"*, que permite leer y mostrar la información de la *"Celda de Manufactura"*, véase la figura 4.5.

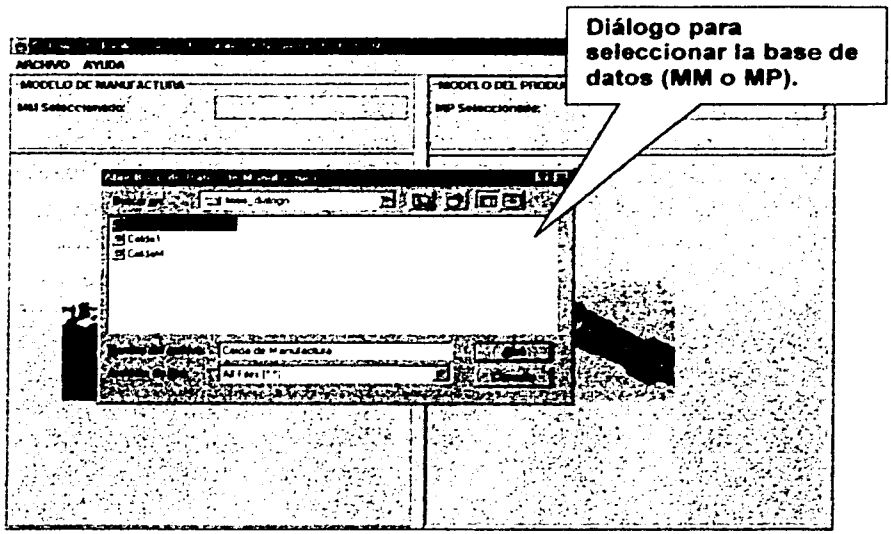


Figura 4.4 Selección de base a consultar.

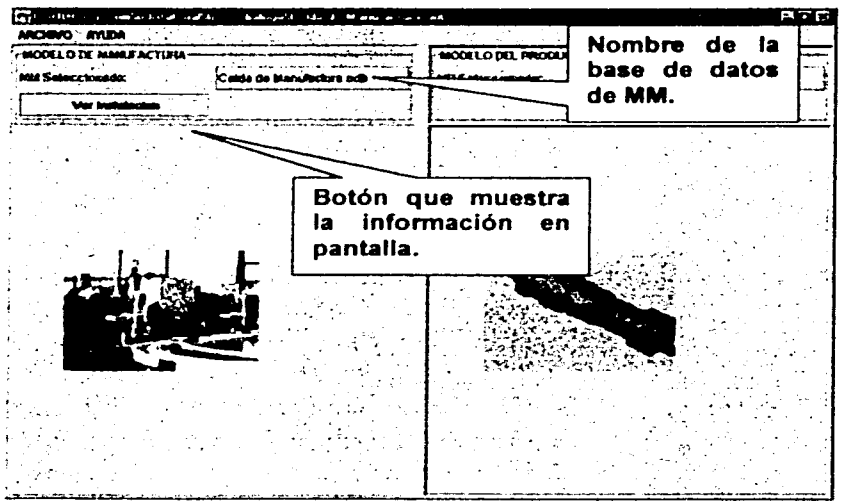


Figura 4.5 Base de Manufactura seleccionada.

La respuesta del sistema es ocultar el botón y mostrar la información existente en la *"Celda de Manufactura"*, colocándola en listas de *"Nombre Instalación"*, *"Nombre de Procesos"* y *"Nombre de Recursos"* (Fig. 4.6). El mismo procedimiento se realiza para la base de MP, *"tremec"*, esto permite que se active el botón *"Ver Ejes"*.

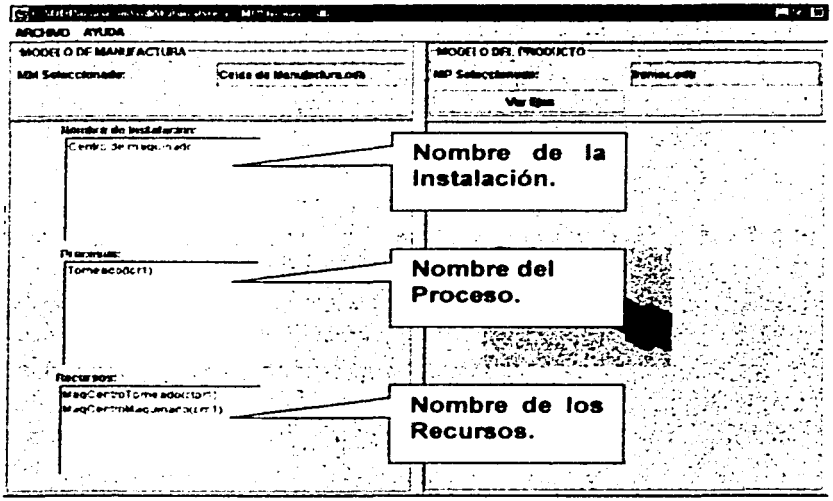


Figura 4.6 Pantalla donde se muestra la base abierta y la información de manufactura.

Al seleccionar el botón *"Ver Ejes"* el sistema SEM muestra la información contenida en *"tremec"* mostrando las listas: *"Nombre de Eje"*, *"Características Primarias" y "Características Secundarias"*, del lado derecho de la pantalla, incluyendo el nombre de *"tremec"* en el cuadro de texto. Al mismo tiempo se oculta el botón *"Ver Ejes"* y se activa la opción de *"Evaluar"*. Esta permite seleccionar de la lista *"Nombre de Eje"* alguno de los ejes existentes para su evaluación (Fig. 4.7).

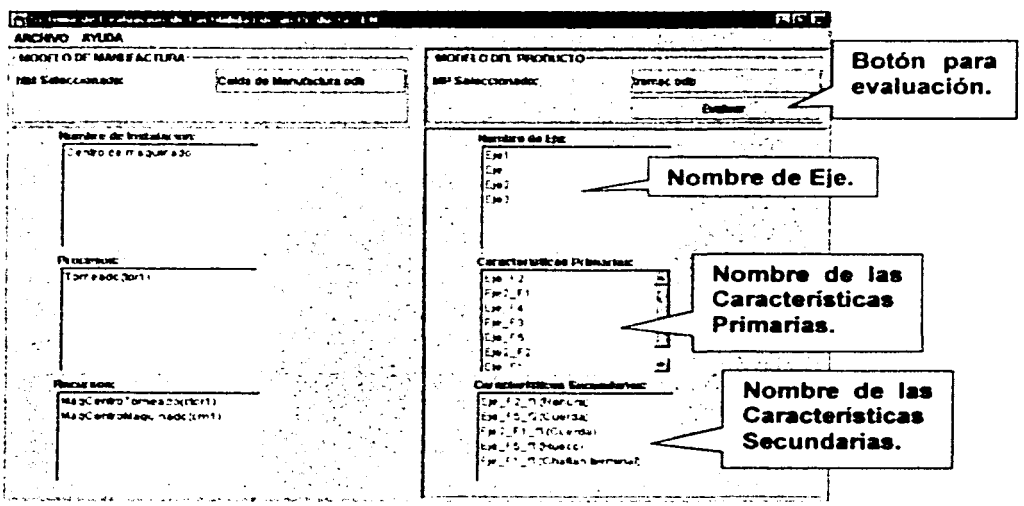


Figura 4.7 Pantalla donde se muestra la opción de "Evaluar".

A continuación se pulsa el botón *"Evaluar"* y se oculta terminando su función, en este momento es posible seleccionar el eje que se quiere evaluar, para el caso de estudio el nombre elegido es *"Eje"* (Fig. 4.8).

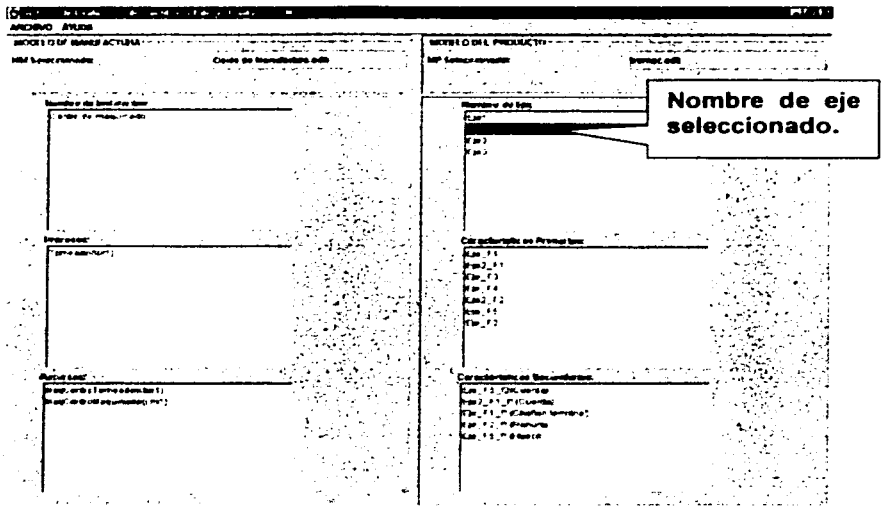


Figura 4.8 Eje seleccionado para evaluar.

Al dar doble clic en el eje seleccionado, SEM efectúa la evaluación de forma interna y envía un diálogo a la pantalla principal en donde se muestra el resultado final de factibilidad o no factibilidad, para el caso de estudio el sistema SEM encontró que es factible la manufactura del *"Eje"* de *"tremec"* en la *"Celda de Manufactura"*.

Dentro del diálogo se incluye una lista de procesos para manufacturar el eje de transmisión, en donde el orden de producción se da de izquierda a derecha respecto al producto o componente; dichos procesos son resultados de la evaluación que realiza SEM, (Fig. 4.9).

Este procedimiento es posible aplicarlo a cada uno de los componentes existentes en *"tremec"*. Si el componente no es factible de manufacturar el sistema SEM puede realizar otra evaluación con otra Instalación, siguiendo la misma mecánica que se realizó para la evaluación de este caso de estudio.

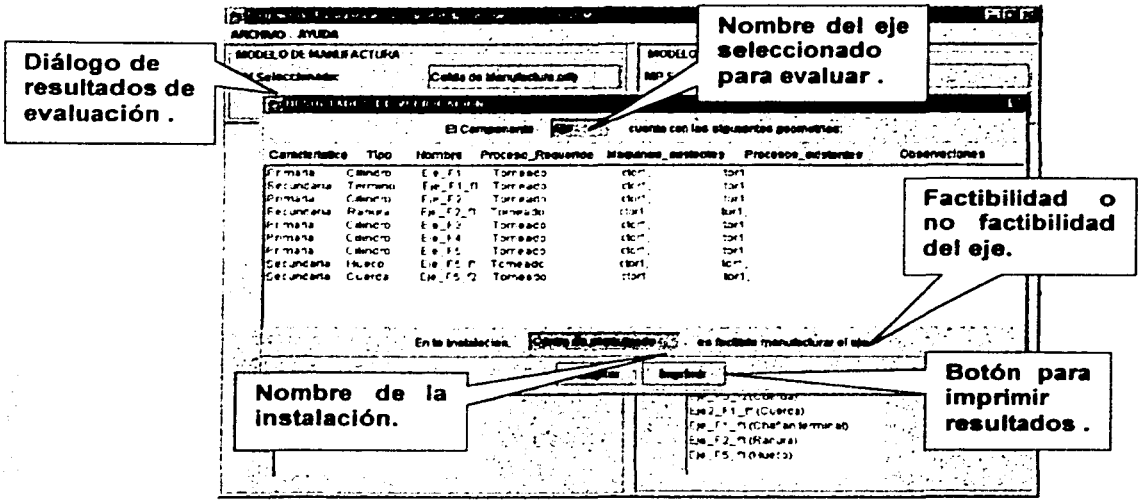


Figura 4.9 Diálogo de resultados de la evaluación del Eje.

Si se desea imprimir los resultados de la evaluación, esto se puede realizar a través de accionar el botón *"Imprimir"*, desde el diálogo de resultados de la evaluación (Fig. 4.10) donde puede seleccionar cualquier impresora local o en red.

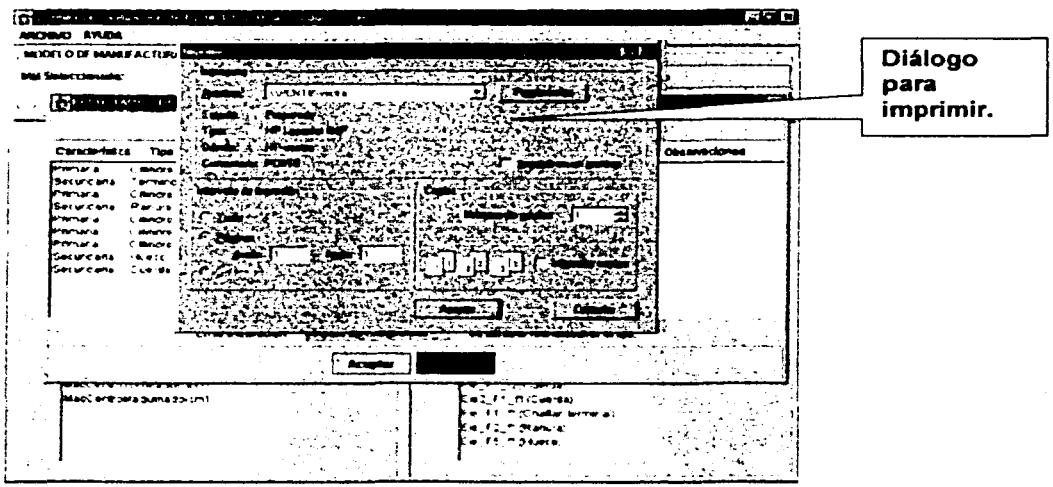


Figura 4.10 Diálogo para imprimir resultados de la evaluación del Eje.

En la figura 4.11 se muestra un ejemplo referente a la impresión de los resultados de la evaluación correspondiente al componente del caso de estudio.

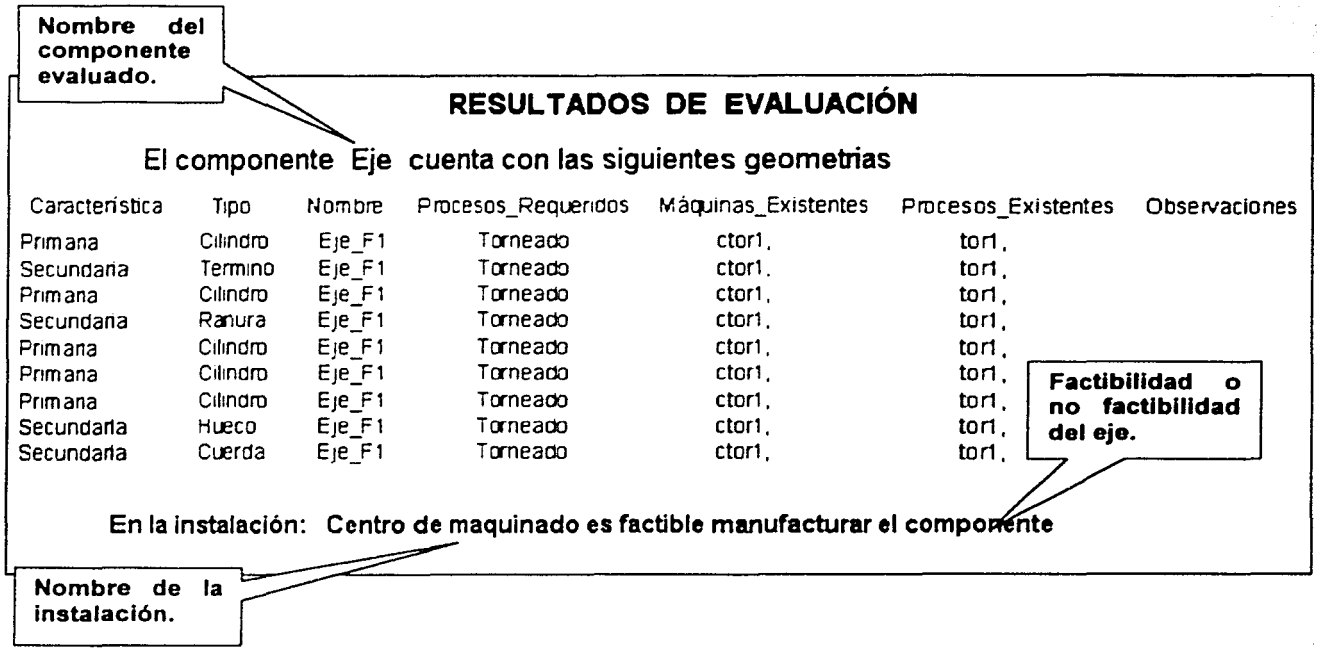


Figura 4.11 Impresión de resultados.

CONCLUSIONES

Con el sistema SEM se logró la integración de los diversos Modelos de Información que se proponen en el proyecto SADET.

Se probó satisfactoriamente que los modelos de información propuestos por Ayala y Mirón asisten al diseño para la manufactura; por lo que el sistema SEM cumple con su objetivo principal al evaluar la factibilidad de manufactura de componentes utilizando un lenguaje orientado a objetos y una base orientada a objetos.

Se realizó el estudio de los diferentes trabajos sobre modelos de información, también se estudiaron las aplicaciones DFM; lo que permitió la mejor implementación del sistema SEM. Para el desarrollo de éste se planteó utilizar las clases que se definieron para el Modelo del Producto propuestas por Mirón, las cuales se implementaron en la aplicación *"met"* la cual se realizó en C++. Sin embargo el desarrollo de SEM estaba enfocado a ser desarrollado en un lenguaje como Java y utilizando Object Store.

Por lo que para poder interactuar con la base del MP en C++ era necesario desarrollar una interfaz. Esta interfaz se intentó desarrollar pero los resultados no fueron los esperados por falta de experiencia en Object Store y la premura en el tiempo de entregar el proyecto SADET, el cual está patrocinado con el proyecto J-27775U de CONACYT. Por lo tanto se decidió emigrar la aplicación de Mirón a Java, apoyándose sólo en una parte de su estructura de clases que define él para su aplicación. Con lo anterior se logró tener una base de datos con el formato de Java, con lo que se pudo interactuar con el sistema SEM.

En cuanto al MM no existió ningún conflicto para accesar a la información, ya que ésta fue realizada en Java.

Como trabajos a futuro SEM puede mejorar las capacidades y requerimientos de manufactura del componente que se da como resultado, como es la generación de información para manufactura para cada componente que incluye una lista de operaciones que va desde el material en bruto hasta el acabado final, estas operaciones deben contar con la siguiente información: velocidades de corte, de avance, tiempos y profundidades de corte.

Este trabajo nos permitió aplicar los conocimientos y herramientas adquiridas durante la carrera, dentro de las cuales podemos mencionar: técnicas de investigación y razonamiento, metodologías de análisis y diseño de software. También nos aporto conocer y aprender UML, como modelador de sistemas; Object Store, como una herramienta de manejo de bases de datos orientadas a objetos. De igual forma se adquirieron nuevos concepto sobre Ingeniería Mecánica, como es la manufactura de componentes y sistemas CAE.

Apéndice A

UML (Unified Modeling Languaje)

UML es una herramienta de modelado de sistemas que permite el análisis y diseño de proyectos de software. UML es muy expresivo, porque abarca todos los panoramas necesarios para desarrollar y estructurar tales sistemas. A través de la notación UML se puede visualizar, especificar y documentar los componentes del sistema. Para aprender a usar UML adecuadamente se requiere aprender tres elementos principalmente: los bloques de construcción básicos, las reglas que dictan como esos bloques deben ser relacionados y algunos mecanismos que aplica todo el tiempo[5].

Hay que tener en cuenta un aspecto importante: UML no pretende definir un modelo estándar de desarrollo como otros métodos de modelaje, tales como OMT (Object Modeling Technique) o Booch, que si definen procesos concretos. En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo. Las diferencias son muy marcadas y afectan a todas las fases del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos[6].

Hoy en día, es necesario contar con un plan bien analizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tienen que ser capaz de señalar cambios si no han captado claramente sus necesidades (o sí cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución en general).

Conforme aumenta la complejidad del mundo, los sistemas informáticos también deberán crecer en complejidad. En ellos se encuentran diversas piezas de hardware y software que se comunican a grandes distancias mediante una red, misma que está vinculada a bases de datos que, a su vez, contienen enormes cantidades de información.

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

La necesidad de diseños sólidos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica) El UML es una misma notación.

Sus Inicios

A partir del año 1994, Grady Booch y Jim Rumbaugh (creador de OMT) se unen en una empresa común, Rational Software Corporation, y comienzan a unificar sus dos métodos. Un año más tarde, en octubre de 1995, aparece UML (Unified Modeling Language) 0.8, la que se considera como la primera versión del UML[6]. A finales de ese mismo año, Ivar Jacobson, creador de OOSE (Object Oriented Software Engineer) se añade al grupo.

Grady Booch

Jim Rumbaugh

Ivar Jacobson

Como objetivos principales de la consecución de un nuevo método que aunara los mejores aspectos de sus predecesores. Lo que se intenta es lograr con esto que los lenguajes que se aplican siguiendo los métodos más utilizados sigan evolucionando en conjunto y no por separado. Y además, unificar las perspectivas entre diferentes tipos de sistemas (no sólo software, sino también en el ámbito de los negocios), al aclarar las fases de desarrollo, los requerimientos de análisis, el diseño, la implementación y los conceptos internos de la OO[6].

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software[6].

Modelado de objetos

UML es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de una especificación. Un modelo es una abstracción de algo, que se elabora para comprender ese algo antes de construirlo. El

modelo omite detalles que no resultan esenciales para la comprensión del original y por lo tanto facilita dicha comprensión.

Con UML sé abstraer cualquier tipo de sistema, sea informático o no, mediante los diagramas, es decir, mediante representaciones gráficas que contienen toda la información relevante del sistema. Un **diagrama** es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las **relaciones** entre los objetos y los vértices se corresponden con los elementos del modelo. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibujan de forma que se resaltan los detalles necesarios para entender el sistema[6].

El Proceso de Desarrollo

UML no define un proceso concreto que determine las fases de desarrollo de un sistema, las empresas pueden utilizar UML como el lenguaje para definir sus propios procesos y lo único que tendrán en común con otras organizaciones que utilicen UML serán los tipos de diagramas[5].

Para entender UML se necesita formar un modelo conceptual del lenguaje y este requiere aprender tres elementos principalmente. Los bloques de construcción básicos, las reglas que dictan como esos bloques pueden ser combinados y algunos mecanismos que se aplican todo el tiempo en UML[5].

✓ Bloques de construcción de UML

Dentro de los bloques de construcción tenemos tres tipos:

1. Elementos de modelo o símbolos
2. Relaciones
3. Diagramas

Así que partiremos hablando con los **elementos de modelo o símbolos**. Los conceptos utilizados en los diagramas son los elementos de modelo que representan conceptos comunes orientados a objetos tales como clases, objetos y mensajes, las relaciones entre estos conceptos incluyendo asociación, dependencia y generalización. Un elemento de modelo es utilizado en varios diagramas diferentes, pero siempre tiene el mismo significado y símbolo. Hay cuatro tipos de Elementos de modelo en UML: estructurales, de comportamiento, agrupamiento y de notación[5].

a) ***Estructurales:*** Estos son los sustantivos de los modelos de UML. La mayoría partes estáticas de un modelo, representando elementos conceptuales o físicos. Hay siete tipos de elementos estructurales[5].

- Una **clase** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semánticas. Gráficamente, una clase es

representada con un rectángulo, usualmente incluyendo su nombre, atributos y operaciones, como se ve en la figura 1.

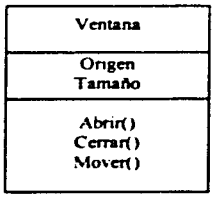


Figura 1. Clase
Representación del elemento Clase en notación UML.

- Una **interfaz** es una colección de operaciones que especifican un servicio de una clase o componente. Una interfaz puede representar el funcionamiento completo de una clase o componente o solo una parte de ese desempeño. Gráficamente una interfaz se representa con un círculo junto con su nombre. Una interfaz raramente es única. Mejor dicho, esta es tipicamente agregada a las clases o componentes que realizan la interfaz como en la figura 2.

Deletrear

Figura 2. Interfaz
Representación del elemento Interfaz en notación UML.

- Una **colaboración** define una interacción y es un conjunto de otros elementos que trabajan a la vez para proporcionar algunas funciones cooperativas que son mayores que la suma de todos los elementos de modelo. Gráficamente, una colaboración se representa con una elipse líneas punteadas, usualmente incluyendo sólo su nombre, como en la figura 3.

Figura 3. Colaboraciones
Representación de Colaboraciones en notación UML.

- Un **caso de uso** es una descripción de un conjunto de secuencias de acciones que un sistema desempeña para permitir un resultado de valor observable para un actor particular. Un caso de uso es usado para estructurar los elementos de comportamiento de un modelo. Gráficamente, un caso de uso se representa con una elipse de líneas sólidas, usualmente incluyendo sólo su nombre como en la figura 4.

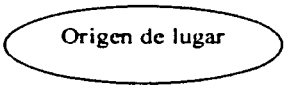


Figura 4. Casos de uso
Representación del elemento de Casos de uso en notación UML.

- Una **clase activa** es una clase cuyos objetos reconocen uno o más procesos o hilos y por lo tanto pueden iniciar una actividad de control. Una clase activa es semejante a una clase excepto que sus objetos representan elementos cuya función es concurrente con otros elementos. Gráficamente una clase activa se representa semejante a una clase pero con líneas más anchas, usualmente incluyendo su nombre, atributos y operaciones, como en la figura 5.

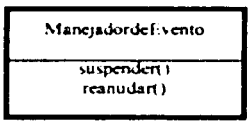


Figura 5. Clases activas
Representación del elemento Interfaz en notación UML.

- Un **componente** es un una parte física y reemplazable de un sistema que conforma y proporciona la realización de un conjunto de interfaces. Un componente típicamente representa el empaquetado físico de diferentes elementos lógicos tal como clases, interfaces, y colaboraciones. Gráficamente, un componente es representado por un rectángulo con pestañas (tabuladores), usualmente incluyendo sólo su nombre como en la figura 6.

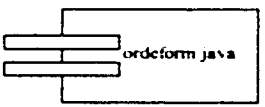


Figura 6. Componentes
Representación del elemento Componentes en notación UML.

- Un **nodo** es un elemento físico que existe al tiempo de ejecución y representa un recurso computacional, generalmente tiene al menos una memoria y frecuentemente capacidad de procesamiento. Un conjunto de componentes puede residir en un nodo y puede también emigrar de un nodo a otro[5]. Gráficamente un nodo es representado por un cubo incluyendo usualmente sólo su nombre como en la figura 7.

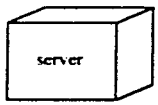


Figura 7. Nodos
Representación del elemento Nodo en notación UML.

b) ***Elementos de comportamiento***: Son las partes dinámicas de los modelos UML. Estos son los verbos de un modelo que representan la función sobre tiempo y espacio. De hecho, hay dos tipos principales de Elementos de comportamiento[5].

- Una **interacción** es una función que comprende un conjunto de intercambio de mensajes entre un conjunto de objetos con un contexto particular para lograr un propósito especifico. Una interacción involucra un número de otros elementos incluyendo mensajes, secuencias de acción (la función invocada por un mensaje), ligas (la conexión entre objetos). Gráficamente un mensaje se representa con una línea dirigida incluyendo sólo el nombre de su operación, tal como en la figura 8.

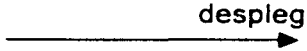


Figura 8. Mensajes
Representación del elemento Mensaje notación UML.

- Una **máquina de estado** es una función que especifica la secuencia de estados de un objeto o una interacción dada durante su tiempo de vida en respuesta a eventos, junto con las respuestas de esos eventos. Una máquina de estados involucra otros elementos incluyendo estados, transiciones(el flujo de un estado a otro), eventos y actividades. Gráficamente se representa con un rectángulo redondeado, incluyendo usualmente su nombre y sus subestados si hay alguno, como en la figura 9.

Figura 9. Estados.
Representación del elemento Estado en notación UML.

c) ***Elementos de agrupamiento***: Son las partes de organización de los modelos UML. Estos son cajas dentro de las cuales un modelo puede ser descompuesto. Hay un tipo principal de Elementos de agrupamiento nombrados paquetes[5].

- Un **paquete** es un mecanismo de propósito general para la organización de elementos en grupos. Los Elementos estructurales, funcionales y aun los de agrupación pueden estar situados dentro de un paquete. A diferencia de los componentes (los cuales existen al tiempo de ejecución) un paquete es puramente conceptual (significa que existe durante el tiempo de desarrollo). Gráficamente un paquete se representa con un fólder tabulado incluyendo usualmente su nombre y en ocasiones su contenido, como en la figura 10.

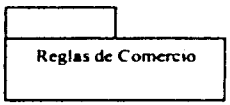


Figura 10. Paquetes.
Representación del elemento Paquete en notación UML.

d) ***Elementos de notación***: Son las partes explicativas de los modelos de UML. Son los comentarios que se pueden aplicar para describir, iluminar y remarcar algunos elementos de un modelo. Hay un tipo principal de Elementos de notación llamado nota[5].

- Una **nota** es simplemente un símbolo para representar las limitaciones y comentarios asociados a un elemento o una colección de elementos. Gráficamente una nota se representa con un rectángulo con una esquina doblada, junto con un comentario textual o gráfico, como la figura 11.

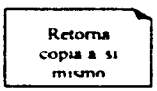


Figura 11. Notas.
Representación del elemento Nota en notación UML.

2. Relaciones

Las relaciones se usan ligar los elementos de modelo. Hay cuatro tipos de relaciones en UML:

a) *Dependencias*. Una dependencia es una relación semántica entre dos Elementos tal que un cambio en uno(el independiente) puede afectar al otro(el dependiente). Gráficamente una dependencia se representa con una línea punteada posiblemente dirigida y ocasionalmente incluye una etiqueta tal como en la figura 12.

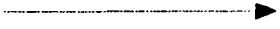


Figura 12. Dependencias
Representación del elemento Dependencia en notación UML.

b) *Asociación.* Una asociación es una relación estructural que describe un conjunto de ligas. Una liga es una conexión entre objetos.

- Una **agregación** es un tipo especial de asociación que representa una relación estructural entre un todo y sus partes. Gráficamente una asociación es representada con una línea sólida posiblemente dirigida, ocasionalmente incluye una etiqueta y frecuentemente contiene otros adornos tal como multiplicidad y nombres de roles como en la figura 13.

Empleado empleo
Figura 13. Asociaciones

c) Generalización.

Una generalización es una relación especialización/generalización en la cual los objetos del elemento especializado(el hijo) son sustituidos por elementos del elemento generalizado(el padre). De esta forma, el hijo comparte la estructura y función del padre. Gráficamente una relación de generalización es representada con una línea sólida con una flecha vacía hacia el padre como en la figura14.

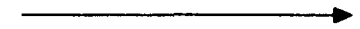


Figura 14. Generalizaciones
Representación del elemento Generalización notación UML.

d) Realización.

Una realización es una relación semántica entre clasificadores(classifiers) donde un clasificador especifica un contrato que otro clasificador garantiza llevar a cabo. Las relaciones de realización se encuentran en dos partes: entre interfaces y las clases componentes que las realizan y entre casos de uso y las colaboraciones que las

realizan. Gráficamente una relación de realización se representa por un híbrido entre una relación de generalización y una de dependencia como en la figura 15.

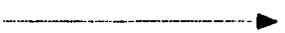


Figura 15. Realización
Representación del elemento Realización en notación UML.

3. Diagramas.

Un diagrama es la representación gráfica de un conjunto de elementos, más frecuentemente representados como una gráfica conectada de vértices(objetos) y arcos(relaciones). Los diagramas se utilizan para visualizar un sistema desde diferentes perspectivas, representa un panorama de los elementos que integran un sistema. Los mismos elementos pueden aparecer en todos los diagramas, sólo en una parte de los diagramas o en ninguno(raramente sucede)[5].

En teoría un diagrama puede contener alguna combinación de objetos y relaciones. UML incluye nueve tipos de diagramas:

1. Diagramas de clases
2. Diagramas de objetos
3. Diagramas de casos de uso
4. Diagramas de secuencia
5. Diagramas de colaboración
6. Diagramas de estado
7. Diagramas de actividad
8. Diagramas de componente
9. Diagrama de distribución

- Un **diagrama de clase** muestra un conjunto de clases, interfaces y colaboraciones y sus relaciones. Estos diagramas son los más comunes del modelado de sistemas orientados a objetos. Proporciona la Vista de diseño estático del sistema[5].

Los componentes de una clase son:

Atributo. Se corresponde con las propiedades de una clase o un tipo. Se identifica mediante un nombre. Existen atributos simples y complejos.

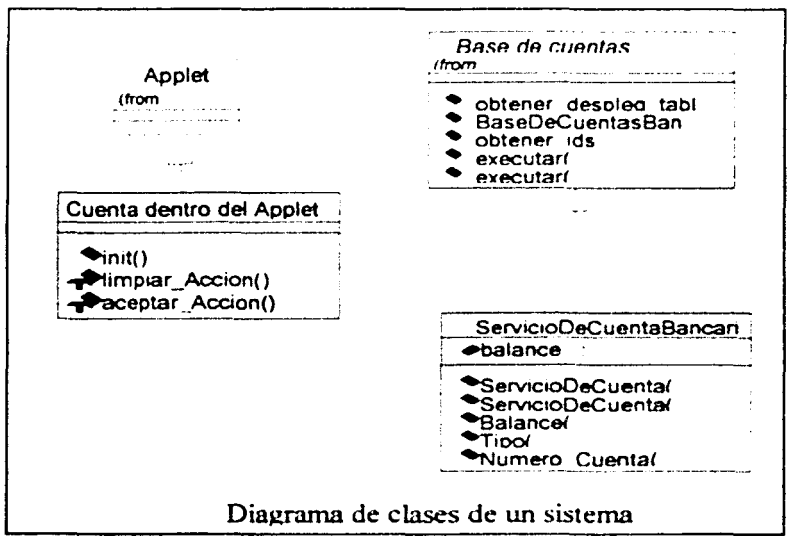
Operación. También conocido como método, es un servicio proporcionado por la clase que puede ser solicitado por otras clases y que produce un comportamiento en ellas cuando se realiza.

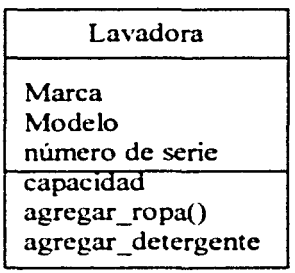
Las clases pueden tener varios parámetros formales, son las clases denominadas plantillas. Sus atributos y operaciones vendrán definidas según sus parámetros formales. Las plantillas pueden tener especificados los valores reales para los parámetros formales, entonces reciben el nombre de clase parametrizada instanciada. Se puede usar en cualquier lugar en el que se podría aparecer su plantilla[6].

Tipos. Un tipo establece una especificación de comportamiento para las clases.

Gráficamente, las clases se representan en una caja rectangular dividida en 3 compartimentos: en la parte superior se encuentra el nombre de la clase en la parte media se encuentran los atributos y en la parte inferior se encuentran las operaciones o métodos de la clase.

Las clases se relacionan entre sí a través de las relaciones que son las líneas rectas entre dos clases. Como se ve en el siguiente diagrama en la figura 16,donde se pone un ejemplo del diagrama de clases de un applet que muestra el estado de cuenta de un usuario. Otro ejemplo es si consideramos que tenemos una clase de Lavadoras que tiene atributos como: la marca, el modelo, el número de serie y la capacidad. Dentro de los métodos de la clase se encuentran: "agregar_ropa", "agregar_ detergente", "activarse" y "sacar_ropa"[Schmuller J., 1999].





Símbolo UML de una clase

Figura 16. Diagrama de clases
Representación del elemento Diagrama de clases en notación UML.

- Un **diagrama de objeto** muestra un conjunto de objetos y relaciones. Representan instancias de los objetos encontrados dentro de diagramas de clase. Estos diagramas direccionan Vista de diseño estático o Vista de proceso estático de un sistema tal como los diagramas de clase pero desde la perspectiva de casos reales o prototípica[5]. En la figura 17 se muestra la forma en que UML representa a un objeto, considerando el ejemplo de la clase Lavadoras, es decir, el símbolo es un rectángulo, como en una clase, pero el nombre está subrayado. El nombre de la instancia específica se encuentra a la izquierda de Los dos puntos(:), y el nombre de la clase a la derecha [Schmuller J., 1999].

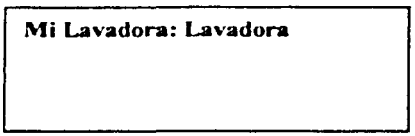


Figura 17. Objeto
Representación del elemento Objeto en notación UML.

- Un **diagrama de caso de uso** muestra un conjunto de casos de uso, actores y sus relaciones. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo[5].

  En este tipo de diagrama intervienen algunos conceptos nuevos:

  - Un **actor** es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

    Un actor se comunica con un caso de uso.
    Un caso de uso extiende otro caso de uso.
    Un caso de uso usa otro caso de uso.

Hay que aclarar que el caso de uso debe ser fácil de entender por el usuario final. Siguiendo el ejemplo de la lavadora, un caso de uso es obviamente lavar ropa, así que un usuario requiriendo esto, se representaría en un diagrama de casos de uso como en la figura 18 [Schmuller J., 1999].

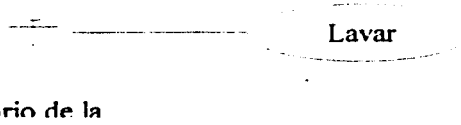


Figura 18.Diagrama de caso de uso.
Representación del Diagrama de caso de uso en notación UML.

- Un **diagrama de secuencia** muestra las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al de los objetos representados en los diagramas de colaboración, es decir son instancias concretas de una clase que participa en la interacción. El objeto puede existir sólo durante la ejecución de la interacción, se puede crear o puede ser destruido durante la ejecución de la interacción. Un diagrama de secuencia representa una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento[5].

En este tipo de diagramas también intervienen los mensajes, que son la forma en que se comunican los objetos: el objeto origen solicita (llama a) una operación del objeto destino. Existen distintos tipos de mensajes según cómo se producen en el tiempo: simples, síncronos, y asíncronos.

Los diagramas de secuencia permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el tiempo de transición, que se especifica en el diagrama. Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran: una manguera de agua(para obtener agua fresca), un tambor(donde se coloca la ropa) y un sistema de drenaje. Por supuesto, estos también son objetos. Para el caso de uso Lavar ropa damos por hecho que completó las operaciones "agregar ropa", "agregar detergente" y "activar", la secuencia sería más o menos la siguiente:

1. El agua empezará a llenar el tambor mediante una manguera.
2. El tambor permanecerá inactivo durante cinco minutos.
3. La manguera dejará de abastecer agua.
4. El tambor girará de un lado a otro durante quince minutos.
5. El agua jabonosa saldrá por el drenaje.
6. Comenzará nuevamente el abastecimiento de agua.
7. El tambor continuará girando.
8. El abastecimiento de agua se detendrá.
9. El agua del enjuague saldrá por el drenaje.
10. El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
11. El tambor dejará de girar y el proceso de lavado habrá finalizado.

Obsérvese que en la figura 19, la captura de las interacciones(representadas por las flechas) que se realizan a través del tiempo (representado por rectángulos verticales) entre el abastecimiento de agua, el tambor y el drenaje(que son los objetos representados con rectángulos en la parte superior del diagrama). En este diagrama el tiempo se da de arriba hacia abajo [Schmuller J., 1999] .

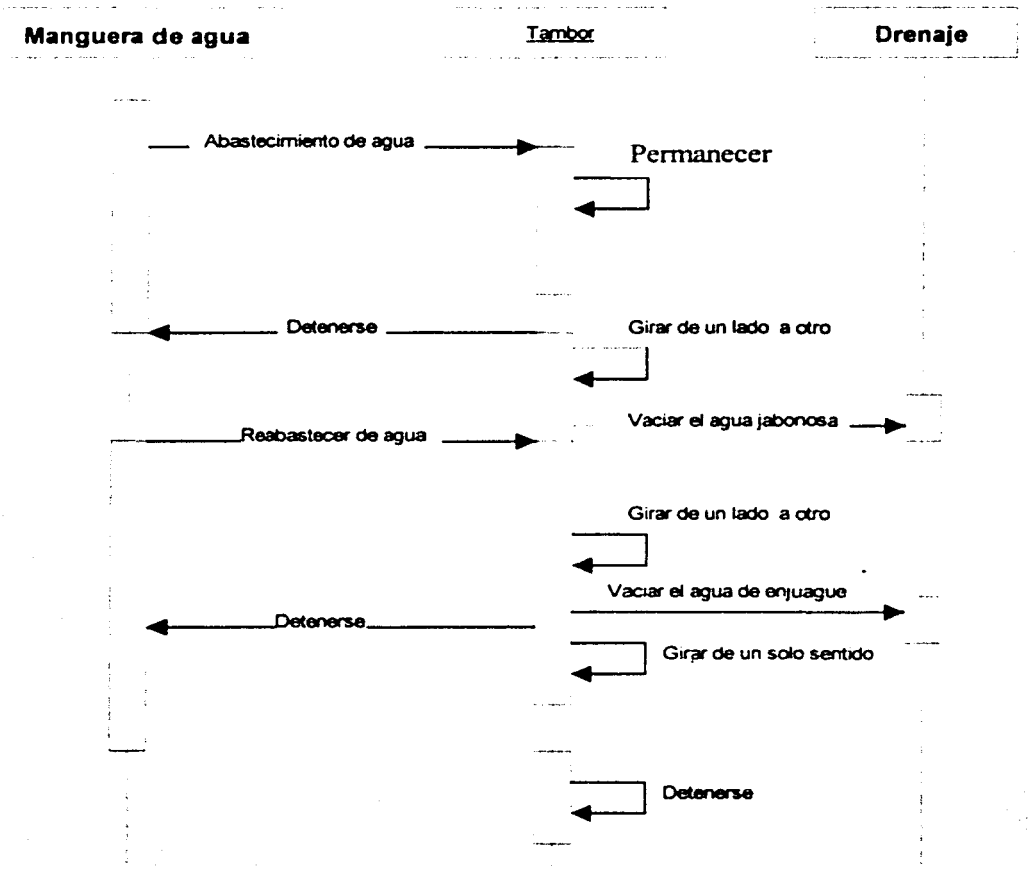


Figura 19. Diagrama de secuencia.
Representación del Diagrama de secuencia en notación UML.

- Un **diagrama de colaboración** muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente[5].

Formando parte de los diagramas de colaboración nos encontramos con objetos, enlaces y mensajes. Tanto los diagramas de secuencia y colaboración son tipos de diagramas de interacción e indican el flujo de mensajes entre elementos del modelo, el flujo de mensajes representa el envío de un mensaje desde un objeto a otro si entre ellos existe un enlace.

Los mensajes que se envían entre objetos pueden ser de distintos tipos, también según como se producen en el tiempo; existen mensajes simples, sincrónicos, balking, timeout y asíncronos. Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces. Los diagramas de secuencia y colaboración son isomórficos, lo que significa que puedes tomar uno y transformarlo en el otro[5]. Para ejemplificarlo podemos decir que se agrega un cronómetro interno al conjunto de clases que constituyen a una lavadora. Luego de cierto tiempo, el cronómetro detendrá el flujo de agua y el tambor comenzará a girar de un lado a otro[Schmuller J., 1999]. Obsérvese en la figura 20.

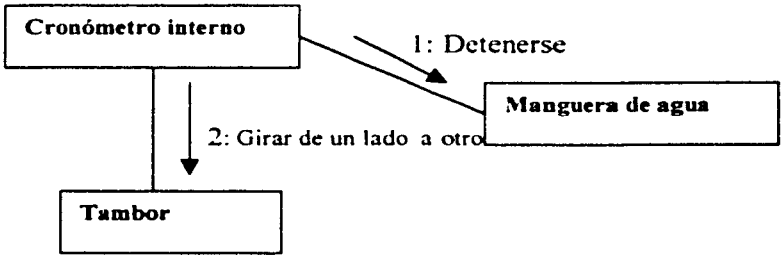


Figura 20. Diagrama de colaboración.
Representación del Diagrama de colaboración en notación UML.

- Un **diagrama de estado** muestra una máquina de estados que consiste de estados, transiciones, eventos y actividades. Son importantes para modelar el desempeño de una interfaz, clase o colaboración reactivos de modelado y enfatiza el desempeño ordenado de eventos de un objeto el cual es especialmente usado en modelado de sistemas reactivos.

Representan la secuencia de estados por los que un objeto o una interacción entre objetos pasa durante su tiempo de vida en respuesta a estímulos (eventos) recibidos. Un estado en UML es cuando un objeto o una interacción satisface una condición, desarrolla alguna acción o se encuentra esperando un evento[5].

Cuando un objeto o una interacción pasa de un estado a otro por la ocurrencia de un evento se dice que ha sufrido una transición, existen varios tipos de transiciones entre objetos: simples (normales y reflexivas) y complejas.

Gráficamente, los estados se representan en rectángulos con esquinas redondeadas y las líneas entre sus estados se llaman transiciones. Las transiciones constan de una sintaxis, donde los parámetros van separados por comas. La condición es una expresión que tiene que considerarse para que el evento se genere y la acción o consecuencia es una expresión que se debe efectuar después del evento puede ser un incremento o un decremento[5].

En cualquier momento, un objeto se encuentra en un estado en particular. Una lavadora podrá estar en la fase de remojo, lavado, enjuague, centrifugado o apagada. El diagrama de la figura 21 muestra las transiciones de la lavadora de un estado al otro el símbolo que está en la parte superior de la figura representa el estado inicial y el de la parte inferior el estado final[Schmuller J., 1999].

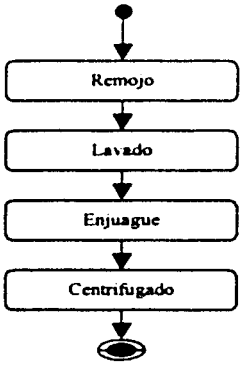


Figura 21. Diagrama de estados
Representación del Diagrama de estados en notación UML.

- Un **diagrama de actividad** es un tipo especial de un diagrama de estado que muestra el flujo de actividad de un sistema. Son importantes para modelar la función de un sistema y enfatiza el flujo de control de los objetos.

Son similares a los diagramas de flujo de otras metodologías OO. En realidad se corresponden con un caso especial de los diagramas de estado donde los estados son estados de acción (estados con una acción interna y una o más transiciones que

suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen.

Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier otra metodología OO). Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema[5]. La figura 22 muestra la forma en que el diagrama de actividades representa los pasos del 4 al 6 (El tambor girara de un lado a otro durante 15 minutos, el agua jabonosa saldrá por el drenaje, Comenzará nuevamente el abastecimiento de agua) de la secuencia que se lleva a cabo en el ejemplo de la lavadora para el caso de uso: Lavar ropa, siguiendo el diagrama de secuencias[Schmuller J., 1999].

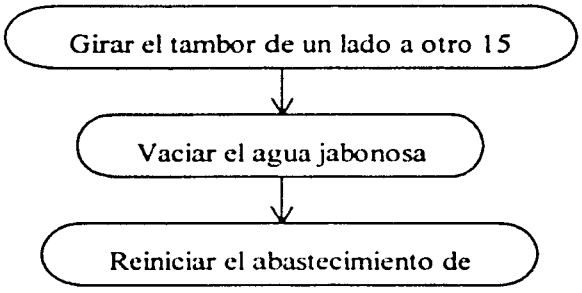
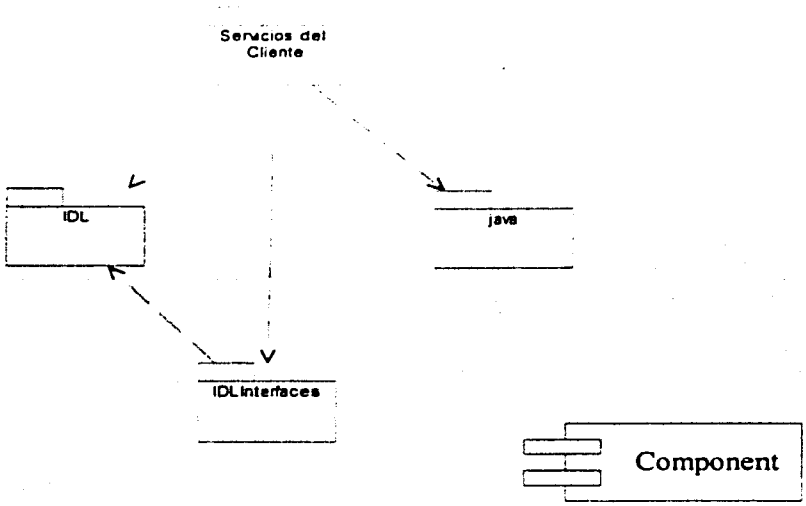


Figura 22. Diagrama de actividades
Representación del Diagrama de actividades en notación UML.

- Un **diagrama de componente** muestra las organizaciones y dependencias de un conjunto de componentes. Estos diagramas direccionan la Vista de implementación estático. Están relacionados a diagramas de clases en donde un componente normalmente mapea uno o más clases, interfaces y colaboraciones.

Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software (una fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación[5].

Cada paquete debe tener un diagrama de componentes para representar las clases que contiene internamente similar a hacer un acercamiento o "zoom" al interior de cada uno de los paquetes[5]. En la figura 23 b), se muestra la representación de un componente. La figura 23 a) muestra la representación de un paquete con un rectángulo con pestaña que en el interior contiene el diagrama de componentes.



a) b)

Figura 23. Diagrama de Componente
Representación del Diagrama de Componentes en notación UML.

- Un **diagrama de distribución.** Este diagrama muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora está representada por un cubo y las interacciones entre las computadoras están representadas por líneas que conectan a los cubos. La figura 24 muestra esta representación[Schmuller J., 1999].

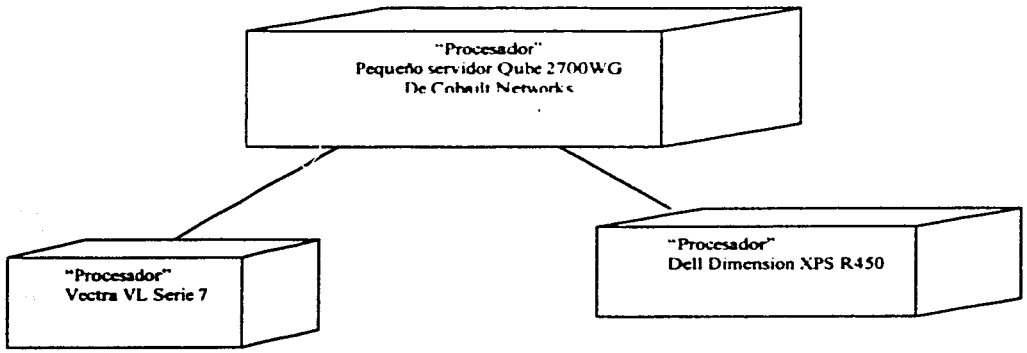


Figura 24. Diagrama de distribución
Representación del Diagrama de distribución en notación UML.

✓ Reglas de UML

Los bloques de construcción no pueden ser modelados a la vez en forma aleatoria. UML tiene un conjunto de reglas que un modelo bien formado debe contemplar. Un modelo bien formado es aquel que es semánticamente consistente en sí mismo y en armonía con sus modelos relacionados.

UML tiene reglas de semántica para:

Nombres: Cómo llamar a los Elementos de modelo, relaciones y diagramas.

Alcance : El contexto que da un significado específico a un nombre.

Visibilidad. Cómo esos nombres pueden ser vistos y usados por otros.

Integridad: Cómo los Elementos se relacionan adecuadamente y consistentemente con otros

Ejecución: Qué significa correr y simular un modelo dinámico.

Las reglas de UML alientan, pero no forzan a direccionar los aspectos más importantes de análisis, diseño e implementación para que los modelos lleguen a ser bien formados con respecto al tiempo.

✓ Mecanismos comunes de UML

UML cuenta con cuatro mecanismos comunes que se aplican todo el tiempo en el lenguaje.

1.Especificaciones

2.Adornos

3.Divisiones comunes

4.Mecanismos de extensibilidad.

- **Especificaciones**: UML es más que un lenguaje gráfico. Mejor dicho, detrás de cada parte de su notación gráfica hay una especificación que proporciona una declaración textual de la sintaxis y semántica de los bloques de construcción. Por ejemplo, detrás de un icono de clase esta una especificación que proporciona el conjunto de operaciones, atributos y función que contempla la clase.

Las especificaciones de UML proporcionan una semántica regresiva que contiene todas las partes de todos los modelos de un sistema, cada parte relacionada a otra en forma consistente[5].

- **Adornos**: La mayoría de los elementos de UML tienen una notación gráfica dirigida y única que proporciona una representación visual de los aspectos más importantes de los elementos. Por ejemplo la figura 20 muestra una clase adornada para indicar que es una clase abstracta con dos operaciones públicas, una protegida y una privada.

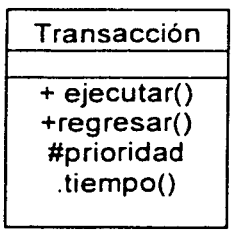


Figura 20. Adornos.
Representación de adornos (+, #, .) en notación UML.

- **Divisiones comunes**: En el modelado de sistemas orientados a objetos al menos se tienen dos formas de división.

Primeramente, la división de clases y objetos. Una clase es una abstracción; un objeto es una manifestación concreta de una abstracción. En UML puedes modelar clases tan bien como objetos. Cada bloque de construcción en UML tiene el mismo tipo de clase/objeto. Por ejemplo, se pueden tener casos de usos y sus instancias de casos de uso, componentes e instancias de componentes, nodos e instancias de nodo y así[5].

Gráficamente, el UML distingue un objeto usando el mismo símbolo que su clase y subrayando el nombre del objeto como en la figura 21.

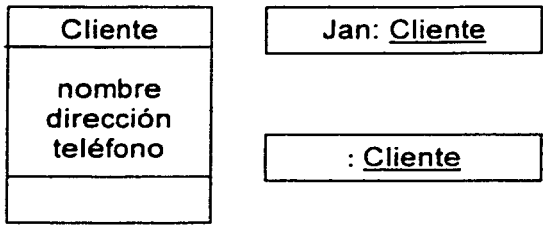


Figura 21. a) Clases y b) Objetos
Representación de la distinción entre clase y objeto en notación UML.

Segundo, existe la separación de interfaz e implementación. Una interfaz declara un contrato(de agregación) y una implementación representa una realización concreta de ese contrato, responsable de llevar a cabo fácilmente la semántica completa de la interfaz.

- **Mecanismos de extensibilidad**: UML es un lenguaje abierto que hace posible que uno pueda extender en forma controlada. Los mecanismos de extensión incluyen:

  - ***Estereotipos***

Un **estereotipo** extiende el vocabulario de UML permitiendo crear nuevos bloques de construcción que son derivados de unos existentes pero que son específicos para el problema. Por ejemplo si estás trabajando en un lenguaje de programación como Java frecuentemente se modelarán las excepciones. En este lenguaje las excepciones son clases y son tratadas en forma muy especial en los modelos que son tratados como bloques de construcción básicos y se pueden hacer con un apropiado estereotipo[5].

  - ***Valores de etiquetado***

Un valor de etiquetado extiende las propiedades de un bloque de construcción de UML, permitiendo crear nueva información de la especificación de ese elemento.

  - ***Restricciones(constraints)***

Una restricción extiende la semántica de un bloque de construcción de UML, permitiendo adicionar nuevas reglas o modificar algunas existentes

Apéndice B

Programación Orientada a Objetos.

Introducción.

La mayoría de los lenguajes experimentales que se han producido en los últimos 10 años son orientados a objetos.

La programación orientada a objetos puede describirse como un conjunto de disciplinas que desarrollan programas que facilitan la construcción de sistemas complejos a partir de componentes.

El atractivo de esta orientación de programación es que proporciona conceptos y herramientas con las cuales se representa y modela el mundo real tan fielmente como sea posible.[8]

Las ventajas claves que usualmente prometen los promotores de la orientación a objetos son la reutilizabilidad y la extensibilidad. Es decir, los sistemas orientados a objetos se tienen que ensamblar a partir de componentes previamente escritos con un esfuerzo mínimo, y el sistema ensamblado será fácil de ampliar sin necesidad de retocar los componentes reutilizados.

La frase "métodos orientados a objetos", comprende una filosofía completa de desarrollo de sistemas que abarca la programación, el análisis de requisitos, el diseño de sistemas, el diseño de bases de datos y muchos otros asuntos relacionados. Los beneficios de la reutilizabilidad y la extensibilidad, por ejemplo, no están limitados a la reutilización o ampliación de fragmentos de código. En teoría, el diseño y los documentos de análisis se pueden almacenar en bibliotecas y se pueden reutilizar o ampliar una y otra vez, siempre y cuando, claro, los usuarios potenciales tengan los medios para conocer fácilmente su existencia. [Graham I., 1996]

¿Qué son los métodos orientados a objetos?

La frase "métodos orientados a objetos" se refiere a varias cosas, como sucede con los términos "orientado a objetos" y "tecnología de objetos". En particular, la frase se refiere a la programación, el diseño, el análisis y las bases de datos orientados a objetos: se refiere, de hecho, a una filosofía completa de desarrollo de sistemas basada en una potente metáfora. [Graham I., 1996]

Terminología y conceptos básicos.

A continuación se induce la terminología de los métodos orientados a objetos. El glosario resume esa terminología e incluye definiciones de algunos términos que

podrían resultar poco familiares. Por fortuna, se está creando una terminología estándar, en gran parte, gracias a los esfuerzos del Grupo de Gestión de Objetos.

La ingeniería de software orientada a objetos se caracteriza por los siguientes términos y conceptos.

- **Objetos.** Son las unidades básicas de construcción, para conceptualización, diseño o programación, son instancias organizadas en clases con características comunes. Estas características comprenden los atributos y procedimientos, denominados operaciones o métodos.

Estos objetos deben estar basados, hasta donde sea posible, en entidades del mundo real y en conceptos de la aplicación o dominio. Los objetos pueden ser clases o instancias, aunque algunas autoridades en la materia utilizan el término objeto como sinónimo de instancia. El estándar OMG es denominar las descripciones de clase como tipos de objetos, y se emplea el término clase para referirse a su realización (implementación). [Graham I., 1996]

> *Un objeto es cualquier cosa real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos. [Martin J., 1994]*

Métodos. Los métodos especifican la forma en que se controlan los datos de un objeto. Los métodos en un tipo de objeto sólo hacen referencia a la estructura de datos de este tipo de objetos. No deben tener acceso directo a las estructuras de datos de otros objetos. Para utilizar la estructura de datos de otros objetos, deben enviar un mensaje a éste. [Martin J., 1994]

Encapsulamiento. El empaque conjunto de datos y métodos se llama encapsulado. El objeto esconde sus datos de los demás objetos y permite el acceso a los datos mediante sus propios métodos. Esto recibe el nombre de ocultamiento de la información. El encapsulado evita la corrupción de datos de un objeto. Si todos los programas pudieran tener acceso a los datos en cualquier forma que quisieran los usuarios, los datos se podrían corromper o utilizar de mala manera. El encapsulado protege los datos del uso arbitrario y no pretendido.

> *Encapsulado es el resultado (o acto) de ocultar los detalles de implantación de un objeto respecto de su usuario. [Martin J., 1994]*

El encapsulado es importante por que separa el comportamiento del objeto de su implantación. Esto permite la modificación de la implantación de un objeto sin que tengan que modificar las aplicaciones que lo utilizan. [Martin J., 1994]

Mensajes. Los objetos, las clases y sus instancias se comunican a través del paso de mensajes. Esto elimina la duplicación de datos y garantiza que no se propaguen los efectos de los cambios en las estructuras de datos encapsuladas dentro de objetos sobre otras partes del sistema. A menudo, los mensajes se realizan como llamadas a funciones. [Graham I., 1996]

Para que un objeto haga algo, le enviamos una solicitud. Esta hace que se produzca una operación. La operación ejecuta el método apropiado y, de manera opcional, produce una respuesta. El mensaje que constituye la solicitud contiene el nombre del objeto, el nombre de una operación, a veces, un grupo de parámetros. [Martìn J., 1994]

- **Clase.** El término clase se refiere a la implantación en software de un tipo de objetos.

> *Una clase es una implementación de un tipo de objetos. Especifica una estructura de datos y los métodos operativos permisibles que se*

La clase especifica la estructura de datos de cada uno de sus objetos y las operaciones que se utilizan para tener acceso a los objetos. La especificación de cómo se lleva a cabo las funciones de una clase se llama método. Los objetos se pueden utilizar exclusivamente con métodos especificos. [Martìn J., 1994]

Herencia. Una clase implanta el tipo de objetos. Una subclase hereda propiedades de su clase padre; una sub-subclase hereda propiedades de las subclases; etc. Una subclase puede heredar la estructura de datos y los métodos, o algunos de los métodos, de su superclase. También tiene sus métodos e incluso tipos de datos propios [Martìn J., 1994].

Polimorfismo. La posibilidad de emplear la misma expresión para denotar operaciones diferentes se conoce como polimorfismo. Con frecuencia, el polimorfismo se realiza por medio de una ligadura dinámica. La herencia es un tipo especial de polimorfismo que caracteriza a los sistemas orientados a objetos. [Graham I., 1996]

Beneficios de la tecnología orientada a objetos.

Mucho de los beneficios sólo se alcanzan cuando el análisis y el diseño orientado a objetos se utiliza con las herramientas OO de CASE, generadora de código basada en depósitos.

- **Reutilización.** Las clases están diseñadas para que se reutilicen en muchos sistemas. Para maximizar la reutilización, las clases se construyen de modo que se puedan adaptar. Un depósito debe estar poblado de una creciente colección de

clases reutilizables. Es probable que las bibliotecas de clases crezcan rápidamente. Un objeto fundamental de las técnicas de OO es lograr la reutilización masiva al construir un software.

- **Estabilidad.** Las clases diseñadas para una reutilización repetida se vuelven estables, de la misma manera que los microprocesadores y otros chips se hacen estables. Las aplicaciones se construyen a partir de chips de software cuando sea posible.

- **El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel.** El encapsulado oculta los detalles y hace que las clases complejas sean fáciles de utilizar. Las clases son como cajas negras; el investigador utiliza las cajas negras y no ve hacia el interior de ésta. Sólo debe entender el comportamiento de la caja negra y cómo comunicarse con ella.

- **Se construyen clases cada vez más complejas.** Se construyen clases a partir de otras clases, las cuales a su vez se integran mediante clases. Así como los bienes manufacturados se fabrican a partir de una serie de materiales de partes y subpartes ya existen, también el software se crea mediante una serie de materiales de clases ya existentes y probados. Esto permite construir componentes complejos de software, que a su vez se convierten en bloques de construcción de software más complejos.

- **Confiabilidad.** Es probable que el software construido a partir de clases estables ya probadas tengan menos fallas que el software elaborado a partir de cero.

- **Nuevos mercados para el software.** Las compañías de software deben proporcionar bibliotecas de software para áreas especificas, que se adapten con facilidad a las necesidades de la organización que las utiliza. La era de los paquetes monolíticos viene siendo remplazada por el software que incorpora clases y paquetes encapsulados de muchos proveedores distintos.

- **Un diseño más rápido.** Las aplicaciones se crean a partir de componentes ya existentes. Mucho de los componentes están construidos de modo que se puedan adaptar para un diseño particular. Los componentes se pueden ver, adaptar y ligar entre sí en la pantalla de herramientas del CASE.

- **Diseño de mayor calidad.** Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.

- **Integridad.** Las estructuras de datos sólo se pueden utilizar con métodos específicos. Estos tienen particular importancia en los sistemas cliente-despachador y los sistemas distribuidos, en los que usuarios desconocidos podrían intentar el acceso al sistema.

- **Programación más sencilla.** Los programas se conjuntan a partir de piezas pequeñas, cada una de las cuales, en general, se puede crear fácilmente. El programador crea un método para una clase a la vez. El método cambia el estado de los objetos en forma que suelen ser sencillas cuando se les considera en sí mismas.

- **Mantenimiento más sencillo.** El programador encargado del mantenimiento cambia un método de clases a la vez. Cada clase efectúa sus funciones independientemente de las demás.

- **Invención.** Los implantadores eficientes encuentran que se pueden generar ideas con rapidez con las herramientas OO más poderosas del CASE, ejecutándose en una estación de trabajo. Las herramientas los anima a investigar e implantar con rapidez sus ideas. Las personas brillantes pueden ser mucho más creativas.

- **Ciclo vital dinámico.** El objetivo del desarrollo de un sistema cambia con frecuencia durante la implantación. Las herramientas OO del CASE facilitan los cambios a la mitad del ciclo vital. Esto permite a los implantadores de software conocer mejor a los usuarios finales, adaptarse a los cambios en las empresas, afinar los objetivos conforme el sistema se consolida y mejorar de manera constante el diseño durante la implementación.

- **Esmero durante la construcción.** Las personas creativas, como los escritores y los novelistas, modifican de manera constante su trabajo durante la implantación. Esto lleva a muchos mejores resultados finales. Las mejores obras creativas se afinan una y otra vez. Las herramientas OO del CASE proporcionan a los constructores de software la capacidad de refinar el diseño conforme éste se implanta.

- **Modelado más realista.** El análisis OO modela la empresa o área de aplicación de manera que sea lo más cercana posible a la realidad que se logra con el análisis convencional. El análisis se traduce de manera directa en el diseño y la implementación. En las técnicas convencionales, el paradigma cambia cuando pasamos del análisis al diseño y del diseño a la programación. Con las técnicas OO, el análisis, el diseño e implementación utilizan el mismo paradigma y lo afinan de manera sucesiva.

- **Mejor comunicación entre los profesionales de los sistemas de información y los empresarios.** Los empresarios comprenden más fácilmente el paradigma OO. Piensan en términos de eventos, objetos y políticas empresariales que describen el comportamiento de los objetos. Las metodologías OO apoyan una mejor comprensión, ya que los usuarios finales y los creadores de software comparten un modelo común.

- **Modelos empresariales inteligentes.** Los modelos empresariales deben describir las reglas con las que los ejecutivos desean controlar su empresa. Estas deben

expresar en términos de eventos y la forma en que éstos deben modificar el estado de los objetos en la empresa. A partir del modelo de empresa se deben deducir diseños de aplicación con la mayor automatización posible.

- **Especificaciones declarativas y diseño.** Las especificaciones y el diseño construidos mediante los formalismos de las herramientas CASE, deben ser declarativos, en la medida de lo posible(que establezcan explícitamente lo que se necesita). Esto permite al diseñador pensar como un usuario final en vez de pensar como una computadora.

- **Una interfaz de pantalla sugestiva para el usuario.** Hay que utilizar una interfaz usuario gráfica, como la de Macintosh, de modo que el usuario apunte a iconos o elementos de un menú desplegado, relacionados con los objetos. En determinadas ocasiones, el usuario puede, de hecho, ver un objeto en la pantalla, Ver y apuntar es más fácil que recordar y escribir.

- **Imágenes, video y expresión.** Se almacenan objetos binarios de gran tamaño (BLOB) que representan imágenes, video, expresiones, texto sin formato o algunos otros flujos de bits de gran tamaño. Con el objeto se utilizan métodos como compresión o descompresión, cifrado o descifrado y técnicas de presentación.

- **Independencia del diseño.** Las clases están diseñadas para ser independientes del ambiente de plataforma, hardware y software. Utilizan solicitudes y respuestas con formato estándar. Esto les permite ser utilizadas en múltiples sistemas operativos, controladores de bases de datos, controladores de redes, interfaces usuario gráficas, etc. El creador de software no tiene que preocuparse por el ambiente o esperar a que éste se especifique.

- **Interacción.** El software de varios proveedores puede funcionar como conjunto. Un proveedor utiliza clases de otros. Existe una forma estándar de localizar clases e interactuar con ellas. La interacción del software de varios orígenes es uno de los objetivos más importantes de los estándares OO. El software desarrollado de manera independiente en lugares ajenos debe poder funcionar en forma conjunta y aparecer como una sola unidad ante el usuario.

- **Computación cliente-despachador.** En los sistemas cliente-despachador, las clases en el software cliente deben enviar solicitudes a las clases en el software despachador y recibir repuestas. Una clase despachador puede ser utilizada por clientes diferentes. Estos clientes sólo pueden tener acceso a los datos del despachador a través de los métodos de la clase. Por lo tanto, los datos están protegidos contra su corrupción.

- **Computación de distribución masiva.** Las redes a nivel mundial utilizarán directorios de software de objetos accesibles. El diseño orientado a objetos es la clave para la computación de distribución masiva. Las clases de una máquina

interactuarán con las de algún otro lugar sin saber donde residen tales clases. Ellas envían y reciben mensajes OO en formato estándar.

- **Computación paralela.** La rapidez de las máquinas mejorará en gran medida mediante la construcción de computadoras paralelas. El procesamiento concurrente se efectuará al mismo tiempo en varios chips de procesadores. Los objetos de procesadores distintos se ejecutarán de manera simultánea, actuando cada uno en forma independiente. Un Object Request Broker estándar permitirá que las clases de procesadores independientes envíen respuestas a otras.

- **Mayor nivel de automatización de las bases de datos.** Las estructuras de datos en las bases de datos OO están ligadas a métodos que llevan a cabo acciones automáticas. Una base de datos OO tiene integrada una inteligencia, en forma de métodos, en tanto que una base de datos de relación básica no.

- **Eficacia de la máquina.** Ha quedado demostrado que las bases de datos orientadas a objetos tienen un rendimiento mucho mejor que las bases de datos de relación para ciertas aplicaciones con estructuras de datos muy complejas. Esto, aunado al cómputo concurrente con el diseño OO, promete un salto enorme en el desempeño de las máquinas. Los sistemas cliente-despachador basado en LAN utilizarán máquinas despachadoras con concurrencia y base de datos orientadas a objetos.

- **Migración.** Las aplicaciones ya existentes, sean OO o no, pueden preservarse si se ajustan a un contador OO, de modo que la comunicación con ella sea a través de mensajes estándar OO.

- **Mejores herramientas CSE.** Las herramientas CASE utilizarán las técnicas gráficas para el diseño de las clases y de la interacción entre ellas, para el uso de los objetos existentes adaptados a nuevas aplicaciones. Las herramientas deben facilitar el modelado en términos de eventos, formas de activación, estados de objetos, etc. Las herramientas OO del CASE deben generar un código tan pronto se definan las clases y permitir al diseñador utilizar y probar los métodos recién creados. Las herramientas se deben diseñar de forma que apoyen el máximo de creatividad y una continua afinación del diseño durante la construcción.

- **Bibliotecas de clases para las industrias.** Las compañías de software venden bibliotecas para diversas áreas de aplicación. Las bibliotecas de clases independientes de la aplicación también son importantes y se proporcionan mejores como una capacidad de las herramientas CASE.

Bibliotecas de clases para las empresas. Las empresas deben crear sus propias bibliotecas de clases, que reflejen sus estándares internos y necesidades de aplicación. La identificación descendente de los objetos en la empresa es un aspecto importante de la ingeniería de la información. [Graham I., 1996]

Lenguajes de Programación.

Existe una amplia gama de lenguajes que están conectador con la idea de la orientación a objetos. Estos van desde lenguajes de programación orientados a objetos "puros" como Smalltalk, hasta lenguajes basados en objetos como Ada. No es posible realizar ninguna clasificación bien definida. Hemos hecho una clasificación como sigue:

- Lenguajes orientados a objetos puros
  - CLOS
  - Eiffel
  - Simula
  - Smalltallk
  -Prolog++ y DLP

- Lenguajes convencionales ampliados
  -C++
  -Objective-C
  -Object Pascal y Turbo Pascal
  -Modula-3 y Classic Ada
  -Object COBOL

- Entornos LISP y IA ampliados
  -KEE, Joshua y ART
  -KBMS y ADS
  -Nexpert Object y Level 5 Object
  -ProKappa, Kappa, ObjectIQ y Xshell

- Lenguaje basados en objetos
  -Ada
  -Modula-2
  -Ellie

- Lenguajes basados en clases
  -CLU

Los diseños orientados a objetos se pueden construir en lenguajes convencionales, pero su dificultad y muchos de los beneficios podrán resultar perdidos.

Se pueden utilizar paquetes de objetos para migrar a la programación orientada a objetos, y seguir protegiendo las inversiones en código convencional.

Hasta hace poco parecía que C++ llegaría a ser el lenguaje de programación orientado a objetos de propósito general con más éxito y más práctico, al menos en el mundo de las estaciones de trabajo y de graduación de escala. Cada vez más, el interés se vuelve hacia Smalltalk y, quizá, hacia Eiffel.

Los lenguajes de programación orientados a objetos son potentes, pero inmaduros, en el sentido de que están adquiriendo nuevas características rápidamente, y siguen emergiendo nuevos lenguajes. Para ciertas aplicaciones, tales como el desarrollo de GUI, son suficientemente maduros y absolutamente indispensables. [Graham I., 1996]

Modelos OO

Modelos de la realidad.

El modelo representa un aspecto de la realidad y se construye de modo que nos ayude a comprender a ésta. El modelo es mocho más sencillo que la realidad. Podemos manejar el modelo y esto nos ayudará a idear sistemas o rediseñar áreas.

Con el análisis orientado a objetos, la forma de modelar la realidad difiere del análisis convencionales. Modelamos el mundo en términos de tipo de objetos y lo que les ocurre a éstos. Esto implica también diseñar y programar sistemas de forma orientada a objetos.

El modelo se convierte en un diseño y más adelante en un código. En la medida de lo posible, este modelo debe tener una forma comprensible para los usuarios y ayudarlos a ser creativos con respecto a sus necesidades.

Los modelos que construimos en el análisis OO reflejan la realidad de modo más natural que los del análisis tradicional de sistemas. Después de todo, la realidad consta de objetos y eventos que cambian el estado de dicho objeto. Quisiéramos capturar el punto de vista de los usuarios con respecto al mundo y traducirlo en software de la manera más automática posible. Entonces, cuando cambien las necesidades de los usuarios, el software cambiará con ellas. [Martín J., 1994]

Herramientas.

Se necesitan las herramientas para el diseño del software. Mediante la información de los modelos, se crea un diseño y se construyen los métodos.

Con las herramientas OO, el diseño y código avanzan de manera natural a parir del modelo de alto nivel. Todo se relaciona con, los tipos de objetos, los métodos que utilizan los objetos y los eventos que activan las operaciones con objetos. El depósito almacena muchas clases y nos ayuda a localizar las que sean útiles en el diseño. Cada vez que hablemos a una herramienta avanzada CASE acerca de las clases, herencia, etc., debe generar un código. Se reutiliza el código de las clases o de los métodos existente, El código de los nuevos métodos se puede generar a partir de enunciados declarativos, colecciones de reglas, tablas de decisión, enunciados SQL, lógica representada mediante diagramas de acción y otros medios. Lo ideal sería que el

código se creara en forma interpretativa, de modo que lo pudiéramos ejecutar siempre que el sistema se modifique. [Martín J., 1994]

Dos tipos de modelos.

En el análisis OO, construimos dos tipos de modelos estrechamente relacionados: un modelo de los tipos de objeto y sus estructuras; y un modelo de lo que ocurre a los objetos.

El análisis y el diseño OO tienen dos aspectos. El primer aspecto se refiere a los tipos de objetos, clases, relaciones entre los objetos y herencia; se le conoce como análisis de la estructura de objetos (AEO) y diseño de la estructura de objetos (DEO). El otro aspecto se refiere al comportamiento de los objetos y lo que les ocurre al paso del tiempo; se conoce como análisis del comportamiento de objetos (ACO) y diseño del comportamiento de objetos (DCO). [Martín J., 1994]

Analogía de Análisis y Diseño.

La transición del análisis al diseño es tan natural, que a veces es difícil especificar el punto final del análisis y el punto inicial del diseño. Esto ocurre sobre todo cuando una herramienta CASE utiliza el mismo paradigma para el análisis y el diseño para generar el código.

El empleo de un modelo conceptual único junto con una herramienta I-CASE para ese modelo tiene como consecuencias:

- Una mayor productividad
- Menos errores
- Una mejor comunicación entre los usuarios, analistas, diseñadores y técnicos.
- Resultados de mejor calidad
- Más flexibilidad
- Mayor inventiva

El análisis de la estructura de objetos se ocupa de definir las categorías de objetos y la forma en que los asociamos.

Cuando el análisis para a la etapa del diseño de la estructura de objetos, identificamos las clases. Se definen las superclases, subclases, ruta de herencia y los métodos a utilizar y se lleva a cabo el diseño en detalle de las estructuras de datos.

En el análisis del comportamiento de objetos se ocupa de modelar lo que ocurre a los objetos al paso del tiempo.

Después sigue la etapa de diseño. En ésta nos preocupamos por el diseño detallado de métodos, ya sea con técnicas por procedimientos o de no por procedimiento. Se crea la entrada para los generadores de código. Se diseña la pantalla y se diseña y generan los diálogos. Finalmente, se construyen y desarrollan los prototipos. Las herramientas CASE que se utilizan deben integrar fuertemente estos aspectos del análisis y del diseño. [Martín J., 1994]

Ingeniería de la Información.

Las empresas modernas necesitan un alto grado de automatización. Se requiere que mochos sistemas computarizados trabajen juntos de la forma más eficiente posible. Los diferentes sistemas que soportan esta cadena de eventos deben ajustarse entre ellos, de modo que la cadena sea lo más eficiente posible, con costos mínimos. Para crear este conglomerado de sistemas, hay que construir un modelo de la empresa. Este modelo se diseña con todo detalle al analizar las áreas de la empresa. Al construir los sistemas independientes, éstos se relacionan con el mismo modelo y, por lo tanto, deben funcionar bien en conjunto.

El proceso de creación de un modelo de empresa, su desarrollo hacia los modelos de las áreas de la empresa y la construcción de sistemas relacionados col los modelos, recibe el nombre de ingeniería de la información. La práctica de la ingeniería de la información ha evolucionado de modelos de entes a modelos de objetos. En el primer caso, los procedimientos se separan de los datos. En el segundo, los procedimientos se empacan con los datos para formar clases.

En la ingeniería de la información, la mayoría de modelos detallados de tipos, subtipos y supertipos de entes, se convierten en tipos de objetos, los procesos se convierten en métodos empaquetados junto con las estructuras de datos.

La ingeniería de la información orientada a objetos aplica el análisis de la estructura de objetos y el análisis del comportamiento de objetos en varios niveles.

Uno de los objetivos principales de la ingeniería de la información es lograr la reutilización del diseño y el código en toda la empresa. Esto es fácil de conseguir, y de modo más eficaz, con las estructuras formales de clases de análisis y diseño orientado a objetos. [Martìn J., 1994]

Adecuación de sistemas anteriores.

Mediante las técnicas OO, podemos poner una envoltura en torno a las aplicaciones antiguas, que les permita comunicarse con las nuevas clases a través de los mensajes estándar de l mundo OO.

Al pasar al nuevo ambiente OO, con todos sus beneficios, no podemos tirar a la basura el ambiente anterior. Durante muchos años, tal vez décadas, los nuevos sistemas deberán coexistir con los sistemas pasados. En los casos donde sea muy caro

mantener los sistemas heredados, se reconstruirán lentamente usando técnicas OO basadas en depósitos. [Martín J., 1994]

HERRAMIENTAS CASE

Introducción.

En los últimos años se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software, y hoy en día la tecnología CASE (Computer Aided Software Engineering) reemplaza al papel y al lápiz por la computadora para transformar la actividad de desarrollar software en un proceso automatizado.

La idea básica de la tecnología CASE es proporcionar un conjunto integrado de herramientas que automaticen el desarrollo de software y el trabajo de mantenimiento, siendo así CASE una combinación de herramientas de software y metodologías de software estructurado. [9]

¿Que es CASE?

"CASE es la automatización del software"

Michael Lucas Gibson

"La creación de sistemas software utilizando técnicas de diseño y metodologías de desarrollo bien definidas, soportadas por herramientas automatizadas operativas en el ordenador"

Etapas en el método CASE.

La metodología CASE se basa en el análisis y desarrollo del tipo descendiente ("top-down") en que el ciclo de vida de un sistema se compone de las siguientes etapas [10]:

- Estrategia
- Análisis
- Diseño
- Construcción
- Documentación
- Transición
- Producción

Apéndice C

IDEF0

La Ingeniería de sistemas es el proceso ordenado para hacer realidad un sistema. Un sistema es una combinación de medios (como personas, materiales, equipos, software, instalaciones, datos, etc.), integrados de tal forma que puedan desarrollar una determinada función en respuesta a una necesidad concreta.

El proceso de ingeniería de sistemas tiene por finalidad la obtención del adecuado equilibrio entre los factores operativos (es decir, prestaciones), económicos y logísticos. Además de las características de "prestaciones" tradicionales, debe prestarse una especial consideración en el diseño a factores como fiabilidad, mantenimiento, factores humanos, capacidad de supervivencia, manufacturabilidad, calidad, desechabilidad, coste de su ciclo de vida y otros afines. La ingeniería de sistemas ayuda a asegurar que estos factores sean adecuadamente integrados de forma concurrente en el diseño, desarrollo y producción de nuevos sistemas, y/o la modificación de los existentes[1].

Partiendo de esto SADT (Structured Análisis and Design Technique) es uno de los métodos conocidos que es usado en ingeniería de sistemas.

SADT es una abreviación para el Análisis Estructurado y Técnica de Diseño, que es una notación gráfica y una aproximación para la descripción de un sistema. Douglas T. Ross es el creador. Desde entonces los analistas en sistemas de Softech han usado y refinado el SADT para dar solución a una gran variedad de problemas. Algunas áreas en las que ha sido útil el SADT son: en Software de Telefonía, diagnósticos y mantenimiento de sistemas, CAD, CAM, configuración de sistemas de cómputo, entrenamiento de personal, software implantado en sistemas de defensa, etc. Así que el programa integrado de manufactura asistida por computadora del Departamento de Defensa de los Estados Unidos reconoció la utilidad del SADT e hizo un estándar dominio público, al que llamó IDEF0.

Bajo el nombre de IDEF0, SADT ha sido usado por miles de personas en organizaciones industriales y militares. En el mundo comercial el SADT es usado típicamente para la definición de requerimientos. Inclusive compite con los métodos orientados al flujo de datos.

EL SADT ha sido usado ampliamente porque es la metodología capaz de representar la características de los sistemas tales como control, alimentación y mecanismos. Esto es porque SADT fue inicialmente desarrollado desde una perspectiva de ingeniería en sistemas más general tan opuesta a todos los otros métodos estructurados que tenían su origen en el diseño de software. También complementa los conceptos existentes al desarrollo de sistemas y los estándares de tiempo, tiene procedimientos explícitos para soportar un grupo de trabajo y la ventaja de ser aplicable a las primeras etapas del desarrollo del sistema.

Las características principales de esta metodología son las siguientes:

- Es un método de modelación basado en análisis estructurado de sistemas.
- El sistema es modelado como una red de actividades interconectadas.
- Permite concentrarse en los aspectos relevantes del proceso analizado.

Cada modelo debe tener un punto de vista, el cual es único para cada caso, el que permite guiar al autor sobre los aspectos relevantes a considerar, el alcance y la profundidad del modelo. Por ejemplo, considerar para el análisis el punto de vista de la empresa que va a desarrollar un proyecto de transformación. De la misma forma, cada modelo debe responder a un propósito u objetivo específico para el cual es desarrollado.

Las unidades básicas dentro de un modelo son las actividades, las que se definen cómo "el componente de un sistema que desarrolla una acción, transformando sus entradas en salidas", y se representan como cajas. En las actividades podemos distinguir:

1.Controles o reglas del proceso: determina cómo o cuándo ocurre una actividad
2.Mecanismos: es una persona, máquina u otro recurso, que desarrolla la actividad.
3.Entradas: es algo que es transformado por una actividad
4.Salidas: es el resultado, o producto, de una actividad.

En el siguiente diagrama se muestra la forma de representar cada uno de éstos:

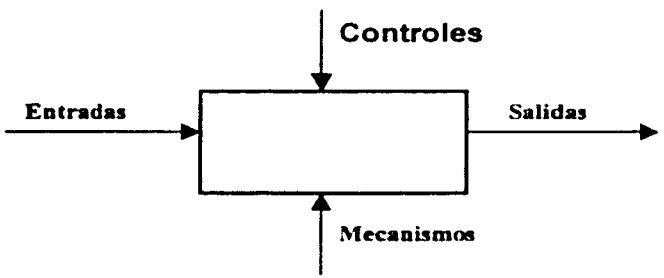


Figura 1. Actividades IDEF0

Cada actividad es descompuesta en un nivel jerárquico inferior, desglosándose en varias subactividades que también tienen controles, mecanismos, entradas y salidas. Dichas subactividades pueden ser desagregadas de igual manera, hasta alcanzar el nivel de detalle que el modelador desea [Marca, David A., Mc Gowan, Clement L., 1987].

Modelos IDEF0

Los modelos colectan y organizan diagramas en una estructura de jerarquía, en donde los diagramas más altos son menos detallados que los que se encuentran más abajo.

Un modelo es una descripción de un sistema que tiene una simple materia(caja), un propósito y una vista. El propósito es un grupo de preguntas para ser contestadas por el modelo. La vista es la perspectiva de cómo el sistema es descrito. El propósito y la vista son conceptos profundos en SADT.

Cada caja puede ser vista como una materia bien definida. La división de dicha materia (el nombre de una caja) es en sus piezas estructuradas (las cajas y flechas que hacen un diagrama), es llamada "descomposición".

En la parte superior de una caja y de varias flechas es usada para definir la frontera de un modelo entero. Una caja describe sobre todo las tareas ejecutadas por el sistema. Las flechas que tocan la caja describen los controles , entradas y salidas del sistema. El diagrama que consiste de esta caja y sus flechas define la frontera del sistema: todo lo de adentro de la caja es una parte del sistema que esta siendo descrito, todo lo de afuera de la caja constituye el ambiente del sistema

El proceso de descomposición estructurado empieza por la frontera de una caja del modelo dentro de una diagrama teniendo desde tres hasta seis cajas, luego una o más de estas en otro diagrama con otras tres o seis cajas y así sucesivamente. El título de cada diagrama es tomado literalmente de la caja que ésta descompone. El resultado de este proceso es un modelo que el diagrama más alto describe a un sistema en general denominado "caja-negra" y que los diagramas de menor nivel describen aspectos más detallados así como operaciones del sistema.

Cada diagrama es por lo tanto una pieza integral de un modelo entero. SADT identifica cada diagrama en un modelo particular por el "número nodo" , este es para el diagrama de contexto que tiene la forma: nombre del modelo o abreviación, diagonal, la letra mayúscula A (para el diagrama de actividad), un guión, y un cero.

Apéndice D

CÓDIGO DE SEM

Código del Archivo "Modulo12.java"

```
/*****************************************************************************************/
package com.odi.tutorial.tesis_final;

import com.odi.*;
import com.odi.util.*;
import com.odi.tutorial.tesis_MP.*;
import com.odi.tutorial.tesis_dialogo.*;
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.awt.List;
import java.util.*;
import javax.swing.*;
import javax.swing.border.*;

/** Intefaz para la Evaluación de Factibilidad de manufactura de un Producto
* <H3>FI-UNAM 2001-2002 by Maria Sofia Martínez A., Gabriela Avila Z.,Lizbeth
Villarruel F. </H3>
* ObjectStore Java Pro.**/

public class Modulo12 extends JApplet implements ObjectStoreConstants {
  public void init() {
    MainFrame frame = new MainFrame(this);
    frame.show();
  }
}
class MainFrame extends JFrame implements WindowListener,
ActionListener,ObjectStoreConstants {

  private JApplet parent;
  private Session session;
  private Database db,db1,db2;
  private static Transaction trans;
  private String fname,dbname1,dbname2;
  private boolean f1=false,f2=false,ba1,ba2,ba3,ba4,ba5,ba6;
  private boolean ba7,ba8,ba9,ba10,ba11,ba12;
  private static Map rootEnti,rootInsta;
  private static Set primarias,secundarias,rootProce,rootRecur;
  private Object[] arre,arre1;
  private int process;
  private static final String exitLabel = "Salir";
  private static final String MMLabel = "Modelo de Manufactura";
```

```
  private static final String MPLabel = "Modelo del Producto";
  private static final String acercadALabel = "Acerca de ..";
  private Dialogos1 box1;
  private JPanel
pa1,pa2,pa3,pa4,pa5,pa6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18;
  private JTextField t1,t2;
  private JButton b1,b2,b3,b4;
  private JLabel l1,l3,l7,l10,l13,l16,l19,im1,im2;
  private List lis1,lis2,lis3,lis4,lis5,lis6;

/* ******* MainFrame()******** */
  public MainFrame(JApplet parent) {
    super("Sistema de Evaluación de Factibilidad de un Producto SEM ");
         this.parent = parent;
       getContentPane().setLayout(new BorderLayout());
   db = null, db1 = null, db2 = null;
    trans = null;
    session = null;
    process = NO_PROCESS;
    backColor = Color.lightGray;
    setBackground(backColor);
    setSize(800, 570);
    setLocation(0, 0);
         Font textFont = new Font("Arial", Font.PLAIN, 14);
         setFont(textFont);
         box1 = new Dialogos1();
         box1.registerFontMetrics(getFontMetrics(textFont));
         box1.initializeMessageBox(this,90,70);
         box1.initializeAcer(this,15,150,100);
       box1.initializeResul(this,20,20,100);
         addWindowListener(this);
         buildMenuBar();
         ventana1(this);
         ventana2(this);
       b1.addActionListener(this);
       b2.addActionListener(this);
       b3.addActionListener(this);
}
/* ---------- ABRIR BASES DE DATOS ---------- */
 private void openBaseMM() {
   if (db != null) {
     showMessageBox("Ver Instalacion antes de abrir MP");
     return;
    }
         maybeInitialize();
```

```
        FileDialog dlg = new FileDialog(this, "Abrir Base de Datos de Manufactura",
FileDialog.LOAD);
        dlg.setFile("*.odb");
        dlg.setVisible(true);
   if (dlg.getFile() == null) {
       return;
        }
        if((lis1 !=null) &&(lis2 !=null) &&(lis3 !=null)){
        lis1.removeAll();
        lis2.removeAll();
        lis3.removeAll();
        t1.setText("");
        b3.setVisible(false);
        ba1=ba2=ba3=ba4=ba5=ba6=ba7=ba8=ba9=ba10=ba11=ba12=false;}
      fname=dlg.getFile();
        dbname1 = dlg.getDirectory() + fname;
        setTitle(dbname1);
  try {
        db = Database.open(dbname1, UPDATE);
        }catch (DatabaseNotFoundException e){
     setTitle("Empty");
     showMessageBox("No se encuentra las base de datos.");
     return;
      }
       if(db!=null){
       trans = Transaction.begin(UPDATE);
       try{
         if(db.getRoot("Instalacion")==null){
            db.close();
              db = null;
            return;
           }
                t1.setText(fname);
                    b1.setVisible(true);
            }catch (DatabaseException e){
          showMessageBox("No es una base de Manufactura");
          setTitle("Empty");
          db = null;
          System.out.println(db);
            return;   }
          trans.commit(ObjectStore.RETAIN_HOLLOW);
       }
}
 private void openBaseMP() {
.....
}
```

```
/* ----------CERRAR BASES DE DATOS---------- */
  private void closeDatabase() {
    ...
 }
/*----------ESTO ES PARA EL MODELO DE MANUFACTURA----------*/
public void getInstaNames()
{ f1=true;
  trans= Transaction.begin(ObjectStore.READONLY);
  rootInsta = (Map)db.getRoot("Instalacion");
  rootProce = (Set) db.getRoot("Procesos");
  rootRecur = (Set) db.getRoot("Recursos");
 String[] names = new String[rootInsta.size()];
 Iterator I1 = rootInsta.values().iterator();
 Iterator P1 = rootProce.iterator();
 Iterator R1 = rootRecur.iterator();
 int m=0;
if(!I1.hasNext())
{System.out.println("No hay instalacion");}
else{
do {
   names[m++]=((Fabrica)I1.next()).getNomInstalacion();
  }while(I1.hasNext());
   for(int j=0; j<names.length;j++)
   {
    lis1.add(names[j]);
   }
 }
if(!P1.hasNext())
{System.out.println("No hay procesos");}
else{
do{
  Object obj = P1.next();
    if(obj instanceof Rolado){
         Rolado Rol = (Rolado)obj;
        String nom1 = Rol.getNomProce();
        String nom = "Rolado(" + nom1 + ")";
        lis2.add(nom);
        }
        else if(obj instanceof Roscado){
         Roscado Ros = (Roscado)obj;
        String nom1 = Ros.getNomProce();
        String nom = "Roscado(" + nom1 + ")";
        lis2.add(nom);
        }
.............
}
```

```
 trans.commit(ObjectStore.RETAIN_HOLLOW);
}
/* ----------ESTO ES PARA EL MODELO DEL PRODUCTO----------*/
//Esto es para poner los componentes al panel de MP
void lee_Comp()
{
f2=true;
 trans= Transaction.begin(ObjectStore.READONLY);
        rootEnti = (Map) db.getRoot("Entidad");
        primarias = (Set) db.getRoot("Primarias");
        secundarias = (Set) db.getRoot("Secundarias");
        String Ejes[] = new String [rootEnti.size()];
        String Cil[] = new String [primarias.size()];
        ....
trans.commit(ObjectStore.RETAIN_HOLLOW);
}
//---Este metodo es para ver cuantos cilindros hay asi como sus secundarias
void makeDialogoCil(String nombresito)
{int w=0;
 Object[] pri_ = lee_primarias_nom(nombresito);
     box1.seteti(nombresito);
  for(w=0; w<pri_.length;w++)
    if(pri_[w] instanceof CCilindro){
      trans= Transaction.begin(ObjectStore.READONLY);
      CCilindro ci= (CCilindro)pri_[w];
       String
sum="Primaria"+esp+"Cilindro"+esp+ci.getNomEnti()+esp+ci.getNomProce();
        trans.commit(ObjectStore.RETAIN_HOLLOW);
      String proc=busca_procesos(6);
      String rec=busca_recursos(1);
      String com=Compara(ba6,ba7,sum,proc,rec);
      box1.addListres1(com);
     seconds(nombresito,w);
    }
}
```

Código del Archivo "Dialogos1.java"

```
/*****************************************************************************************/
package com.odi.tutorial.tesis_final;
import java.awt.*;
import java.awt.event.*;
import java.awt.print.PrinterJob;
import java.awt.print.*;
import javax.swing.*;
import javax.swing.border.*;
```

```
// Maneja varias cajas de Dialogos1.
public class Dialogos1 implements Printable {

  private FontMetrics metrics;
  private DialogBox simpleBox;
  private Label simpleLabel;
  private DialogBox inputBoxAcer;
  private int inputCols1;
  private DialogBox inputBoxResul;
  private int inputColsResul;
  private Label siLabel,comLabel,geomLabel,datoLabel,instLabel;
  private TextField ancla,ancla2;
  private JButton btimpri;
  private List listares;
  private Font fnt,fnt2;
  private int margsupy=110,margsupy2=115;
  private int paso=13;
  private String esp="       ";

  Modulo12 mod=null;
   MainFrame newMainFrame;

  public Dialogos1() {
    metrics = null;
    mod= new Modulo12();
}
  public void registerFontMetrics(FontMetrics metrics) {
    this.metrics = metrics;
  }
  public FontMetrics getFontMetrics() {
    return metrics;
  }
  /*initilizeMessageBox()*/
  public void initializeMessageBox(JFrame parent, int x, int y) {
        ...
  }
  public void showMessageBox(String message) {
        ...
  }
  public Object getMessageBoxSource(){
        ...
  }
/*-----------------PARA EL DIALOGO DE RESULTADOS -----------------*/
public void initializeResul(JFrame parent, int inputCols, int x, int y)
{
  inputBoxResul = new DialogBox(parent, true, x, y);
  inputColsResul=inputCols;
```

```
 inputBoxResul.setTitle("RESULTADOS DE VERIFICACION");
 fnt= new Font("Helvetica-Bold",Font.PLAIN,12);
 fnt2= new Font("Serif",Font.BOLD,16);
 JPanel res = new JPanel();
 JPanel res2 = new JPanel(new BorderLayout());
 JPanel res3 = new JPanel(new BorderLayout());
 JPanel res4 = new JPanel(new FlowLayout());
 JPanel res5 = new JPanel();
 ancla = new  TextField(5);
 ancla.setEditable(false);
 ancla2 = new  TextField(18);
 ancla2.setEditable(false);
 listares= new List(6);
 listares.setMultipleMode(true);
 siLabel = new Label("",Label.CENTER);
 comLabel=new Label("El Componente", Label.LEFT);
 geomLabel=new Label("cuenta con las siguientes geometrias:", Label.RIGHT);
 datoLabel=new Label(" Caracteristica      Tipo      Nombre      Proceso_Requerido
Maquinas_existentes      Procesos_existentes      Observaciones");
 instLabel=new Label("En la instalacion:",Label.CENTER);
  res.setBorder(new EtchedBorder());
 res.add(comLabel);
 res.add(ancla);
 res.add(geomLabel);
 inputBoxResul.getContentPane().add("North", res);
 res2.add(datoLabel, BorderLayout.NORTH);
 res2.add(listares,BorderLayout.CENTER);
 res4.add(instLabel);
 res4.add(ancla2);
 res4.add(siLabel);
 res3.add(res2,BorderLayout.CENTER);
 res3.add(res4,BorderLayout.SOUTH);
 inputBoxResul.getContentPane().add("Center",res3);
  JButton btacep = new JButton("Aceptar");
  btimpri = new JButton("Imprimir");
  res5.setBorder(new EtchedBorder());
  res5.add(btacep);
  res5.add(btimpri);
  btacep.addActionListener(inputBoxResul);
  inputBoxResul.getContentPane().add("South", res5);
  BoxesActions resAction = new BoxesActions();
  btimpri.addActionListener(resAction);
  listares.addActionListener(resAction);
 inputBoxResul.setLocation(x, y);
}

        ...
```

GLOSARIO DE TÉRMINOS

AMRF: Facilidad de Investigación de Manufactura Avanzada.
BDOO: Base de Datos Orientada a Objetos.
CAD: Diseño Asistido por Computadora.
CAE: Ingeniería Asistida por Computadora.
CAQ: Control y Seguridad de Calidad
CAM: Manufactura Asistida por Computadora.
CAPP: Planificación asistida por computadora
CIM: Manufactura Integrada por Computadora.
CNC: Control Numérico por Computadora.
DFM: Diseño para la Manufactura.
ESPRIT: Programa Estratégico Europeo para la Investigación y Desarrollo en la Tecnología de la Información
IGES: Sistemas de Intercambio de Gráficas Iniciales.
IC: Ingeniería Concurrente .
MCM o mcm: Modelador de Celdas de Manufactura.
MET o met: Modelador de Ejes de Transmisión.
MI: Modelos de Información.
MP: Modelo del Producto.
MM: Modelo de Manufactura.
NC: Control Numérico.
OSI: Organización Internacional de Estándares.
PPC: Control de Programa de Producción.
SADET: Sistema Auxiliar para el Diseño de Ejes de Transmisión.
SEM: Sistema de Evaluación de Manufactura.

BIBLIOGRAFÍA

Al-Ashaab, A. H., 1994, *"A manufacturing model to capture injection moulding process capabilities to support design for manufacture"*, PhD Tesis, Departament of Manufacturing Engineering, Loughborough University.

Al-Ashaab, A. H. S., Young, R., 1994, *"Design for Injection moulding in a Manufacturing Model Environment"*, Submitted to the Journal of Design and Manufacturing, the Inter. Journal of Concurrent Engineering.

Ayala, R. A., 2001,Tesis de Maestría *"Modelo de Manufactura de una Celda de Producción"*, DEPFI, UNAM.

Booch, G., 1991,*"Object-oriented design with applications"*, Benjamin/Cummings Inc.

Borja, V., 1997, *"Redesing Supported by Data Moldels with Particular Reference to Reverse Engineering"*, PhD Thesis, Department of Manufacturing Engineering, Loughborough University.

Borja, V., López, M., Valeriano, G., González, L., Santillán, S., 1997, *"Diseño para manufactura asistido por computadora: el Agente para Torneado"*, Memorias del segundo congreso de la sociedad Mexicana de Ingenieros Mecánicos, SOMIM. Centro de Diseño y Manufactura, Circuito Exterior, Talleres del Anexo Facultad de Ingeniería, Cd. Universitaria, D.F., México.

Borja, R. H., Morano, O., 2000, *" Análisis de modelos de producto para diseñar moldes de inyección"*, Centro de diseño y Manufactura, Facultad de Ingeniería, Universidad Nacional Autónoma de México, VI Congreso Anual de la Sociedad Mexicana de Ingeniería Mecánica, A. C., Colima, México.

Borja, R. H., Morano, O., 2001, *" Modelo de producto para el diseño concurrente de una pieza de inyección y su molde"*. Centro de diseño y Manufactura, Facultad de Ingeniería, Universidad Nacional Autónoma de México, Circuito Exterior, Cd. Universitaria, D.F., México.

Canciglieri, O., Young, R., 1998, *"Information interactions in data model driven design for manufacture"*, Departament of manufacturing Engineering, Loughborough, UK.

Ellis, T. I. A., 1995, Untitled PhD Thesis, Manufacturing Engineering Department, Loughborough University of Technology.

Graham I., *"Métodos Orientados a Objetos"*, Editorial Addison Woley / Díaz de Santos, 2° Edición 1996.

Kastelic, S., Kopac,J., Peklenik, J., 1993, " *Conceptual Design of Relational Data Base for Manufacturing Processes*", Annais of the CIRP.

Klein, J., and Scheer, A. D., 1990, *"Information modelling – the basis for integrated information systems"*. Workshop Implementing CIM, Saabrucken, Germany.
Krause, F. L., Kimura, F., Kjellberg, T., Lu, S.C. Y., 1993,*"Product Modelling"*, Annals of the CIRP Vol. 42.

Kunwoo, L., 1999, *"Principles of CAD CAM CAE Systems"*, Addison-Wesley.

Lee, R. J. V., 1993, *"Design for manufacture of injection moulded products"*, PhD Report, Manufacturing Engineering Department, Loughborough University of Technology.

Marca, David A., Mc Gowan, Clement L., 1987, *"SADT Structured Análisis Design Technique."*, Mc Graw-Hill.

Martín J, *"Análisis y Diseño Orientado a Objetos"*, Editorial Prentice-Hall, 1994, México.

Mirón G. Hugo G, 2001. Tema de Maestría *"Diseño e Implementación de un Sistema Modelador de ejes de Transmisión"*.DEPFI, UNAM.

Molina, A., 1995, *"A Manufacturing Model of support Data-Driven Applications for Design and Manufacture"*, PhD Tesis, Departament of Manufacturing Engineering, Loughborough University.

Morano O Héctor Rafael, 2001. Tesis de Doctorado *"Diseño de Moldes de Inyección Asistido por Modelos de Información"*. DEPFI, UNAM.

Prasad, B. 1996, *"Toward a Functional Design of a Concurrent Information Modeling System"*, Computing Modeling and Simulation in Engineering, Vol. 1.

Rob, Peter, Carlo, Coronel, 1997, *"Database Systems: Design, Implementation, and Management"*. International Thomson Publishing, Cambridge, MA.

Rumbaugh, J., Blaha, M., Premerla, W., Eddy, F., and Lorensen, W., 1991, *"Object-oriented modeling and design"*, Prentice Hall Inc.

Schmuller Joseph, 1999. Aprendiendo UML en 24 horas. Prentice-Hall.

Sohlenius, G., 1992,*"Concurrent Engineering"*, Annals of CIRP, Royal Institute of Technology, Stockholm.

Ulrichrembold, Bartholonew, Nnjf.,1991, *"The role of manufacturing models for the information technology of the factory of the 1990s"*,Journal of Design and Manufacturing, 1, 67-87.

Vega G. F., 2002. Tesis de Licenciatura *"Interfase entre un Modelo del Producto y un Modelador de Sólidos"*. Facultad de Ingeniería, UNAM.

Young, B., 1989, *"Integrated Design and Manufacture Systems"*, Department of Manufacturing Engineering, Loughborough University of Technology, Loughborough Leics., UK.

Referencias de Internet:

[1] http://www.ur.mx/ur/faciya/carreras/cursos/sis/mod-dat1/prosist.HTM, Revista soluciones avanzadas, 1994.

[2] http://www.acm.org/crossroads/espanol/xrds7-3/objects.html Todd R. Manion, traducido por Rogina, Pablo J., 2001, *"bjetos Objetos en Todos Lados."*

[3] http://www.itlp.edu.mx/publica/revistas/revista_iscanteriores/mzo99/bdoo.html Corral, Clarissa, Flores, Brenda, Moroyoqui, Guadalupe. 1999, Traducción de *"Fundamentals of Object-Oriented Database" Fundamentos de Bases de Datos Orientadas a Objetos.).*

[4] http://www.monografias.com
Trabajo enviado por: Ruiz Tijerina, Martha Esthela, 2000, *"Bases de Datos"*.

[5] http://uxmcc1.iimas.unam.mx/~cursos/objetos/temario/temario.html Oktaba Hanna,, Tecnología Orientada a objetos, versión 1999.

[6] http://gidis.ing.unlpam.edu.ar/ingles/personas/glafuente/uml/uml.html

[7] http://www.isdefe.es/maqueta_isdefe/isdefe_3a/publicaciones/mono1.htm Benjamín S. Blanchard, 2001, "Ingeniería de Sistemas".

[8] http://w3.mor.itesm.mx/~rdec/node92.html

[9] http:// docencia.dgsca.UNAM.mx/cursos/cursos/temarios/sistdinfor/SIO3.html

[10] http:// www.monografias.com/trabajos/vica/vica.html