UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
"CUAUTITLAN"

"EVALUACION COMPARATIVA DE METODOLOGIAS DE ANALISIS Y DISEÑO ORIENTADAS A OBJETOS"

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADA EN INFORMATICA

P R E S E N T A N :

MARIA DE LOURDES MONTIEL GARCIA
MARIA ISABEL PEREZ ARREDONDO

ASESOR: ING. LAURA SANDOVAL

CUAUTITLAN IZCALLI, EDO. DE MEX. 1997

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

ASUNTO: VOTOS APROBATORIOS

DR. JAIME KELLER TORRES
DIRECTOR DE LA FES-CUAUTITLAN
P R E S E N T E .

AT'N: Ing. Rafael Rodriguez Ceballos
Jefe del Departamento de Exámenes
Profesionales de la F.E.S. - C.

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS: "Evaluación Comparativa de Metodologías de Análisis y Diseño Orientadas a Objetos".

que presenta la pasante: María Isabel Pérez Arredondo

con número de cuenta: 8738206-8 para obtener el TITULO de: Licenciada en Informática.

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E .
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Edo. de Méx., a 9 de Diciembre de 1996.

PRESIDENTE	Ing. Sergio P. Acosta Torres	6/Dic/97
VOCAL	Ing. Laura Sandoval Montaño	10/12/97
SECRETARIO	Lic. Armando Carmona Bonilla	08/01-97
PRIMER SUPLENTE	Ing. Manuel Jauregui Renault	30/01/97
SEGUNDO SUPLENTE	Lic. Conrado Camacho Arteaga	28/1/97

UAE/DEP/VAP/01

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

DR. JAIME KELLER TORRES
DIRECTOR DE LA FES-CUAUTITLAN
P R E S E N T E .

AT'N: Ing. Rafael Rodriguez Ceballos
Jefe del Departamento de Exámenes
Profesionales de la F.E.S. - C.

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

"Evaluación Comparativa de Metodologías de Análisis y Diseño Orientadas a Objetos".

que presenta la pasante: María de Lourdes Montiel García

con número de cuenta: 8739322-6 para obtener el TITULO de:

Licenciada en Informática .

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E .
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlan Izcalli, Edo. de Méx., a 9 de Diciembre de 1996.

PRESIDENTE	Ing. Sergio P. Acosta Torres	6/12/97
VOCAL	Ing. Laura Sandoval Montaño	10/12/97
SECRETARIO	Lic. Armando Carmona Bonilla	08-01-97
PRIMER SUPLENTE	Ing. Manuel Jauregui Rennault	7/01/97
SEGUNDO SUPLENTE	Lic. Conrado Camacho Arteaga	28/1/97

UAE/DEP/VAP 01

A mis padres, hermanas y hermanos, por el apoyo brindado y por la paciencia que me han tenido durante los días y las noches de trabajo que dedique para desarrollar esta tesis.

A mi asesora, Ing. Laura Sandoval Montaño, que desinteresadamente acepto guiarnos en el desarrollo de esta tesis, de la misma forma agradesco su paciencia, aliento y apoyo en mi vida profesional.

A Isabel Pérez Arredondo, amiga y compañera de universidad, que acepto colaborar conmigo para realizar esta tesis.

A mis amigos, quienes comparten conmigo la felicidad y la satisfacción de haber finalizado este tan esperado trabajo.

A Dios por darme la posibilidad de realizar uno de mis objetivos profesionales.

Dedico esta tesis, así como el arduo trabajo a lo largo de muchos años persiguiendo el noble fin de ser cada día mejor

A MI MADRE

C.P. MA. DE LA LUZ ARREDONDO MUÑOZ

con todo mi amor y respeto, porque ha sido la luz e inspiración de mi vida.

Quiero dejar constancia de mi gratitud a aquéllos que me alentaron a continuar en la lucha por lo que vale la pena en la vida y a no rendirme ante los pequeños tropiezos, así como a los que colaboraron incondicionalmente para que este anhelo se hiciera realidad; todos ellos: mi familia, mis amigos y mis profesores, son personas muy queridas que estarán siempre en mi corazón.

Por último, quiero expresar mi reconocimiento a la labor de Laura Sandoval por su tiempo y dedicación durante el desarrollo de esta tesis, así como a Lourdes Montiel por su amistad, compañerismo y trabajo incansable.

María Isabel Pérez
Arredondo

HIPOTESIS:

- Si se estudian las características de los sistemas estructurados y de los orientados a objetos entonces se pueden comparar para obtener una visión de las ventajas y desventajas que cada uno ofrece.

- Si se aplican las Metodologías Orientadas a Objetos entonces la forma en que se realiza el modelado de un problema es cambiada en cuanto a la abstracción de los elementos integrantes del sistema y su interacción.

- Si se estudian varias Metodologías Orientadas a Objetos entonces se cuenta con un mayor número de herramientas para llevar a cabo el análisis y diseño de un sistema en particular.

- Si se aplican las Metodologías Orientadas a Objetos entonces se logra incrementar el capital de software mediante componentes reutilizables.

- Si se aplican las Metodologías Orientadas a Objetos entonces se tiene a disposición una amplia notación para la documentación del análisis y diseño de los Sistemas de Información.

OBJETIVOS GENERALES:

- Analizar y evaluar comparativamente tres Metodologías de Análisis y Diseño Orientadas a Objetos utilizadas en la Ingeniería de Software, a fin de concluír si los objetivos pretendidos por ellas son funcionales.

- Aplicar las Metodologías Orientadas a Objetos analizadas en un Sistema de registro de información de personal docente de una institución que imparte clases de nivel medio superior.

OBJETIVOS ESPECIFICOS:

- Definir y comprender los conceptos básicos de las técnicas tradicionales de análisis y diseño de los sistemas procedimentales o estructurados.
- Comprender las causas que originaron el surgimiento de las Metodologías Orientadas a Objetos.
- Definir y comprender los conceptos básicos de la Tecnología Orientada a Objetos.
- Comparar los sistemas estructurados y los orientados a objetos, resaltando beneficios y riesgos que presentan cada uno de ellos.
- Estudiar y comprender las Metodologías Orientadas a Objetos de Rebeca Wirfs-Brock, Edward Yourdon y Grady Booch.
- Aplicar los conocimientos adquiridos a través del estudio de los puntos anteriores, en el Diseño de un Sistema de registro de información de personal docente de una institución que imparte clases de nivel medio superior, siguiendo las etapas definidas en cada una de las Metodologías Orientadas a Objetos estudiadas.

INTRODUCCION

A lo largo del tiempo, las organizaciones han reconocido la importancia de administrar los recursos claves, tales como el trabajo y la materia prima. Actualmente, la información ha sido colocada en su merecido lugar como un recurso clave. Los tomadores de decisiones han comenzado a comprender que la información no sólo es un resultado de manejar empresas, sino la base para su organización y crecimiento.

Para maximizar la utilidad de la información en una empresa, ésta debe ser administrada correctamente, de la misma forma que otros recursos lo son. El uso de las computadoras ha servido para lograr su buena administración y su manejo.

El desarrollar software para un sistema de información resulta una tarea ardua y de alta complejidad, ya que involucra la definición de objetivos, alcances y procesos que se deben cumplir, abarcar y realizar. Esta tarea implica el seguimiento de metodologías, medidas de control para el desarrollo del software.

Debido a la complejidad y magnitud de los sistemas el software se desarrolla en periodos mayores a los planeados y consumiendo mayores recursos económicos y humanos. A consecuencia el software creado tiene defectos, que en su mayoría se deben a la falta de visión a futuro en su diseño y conceptualización, ya que al ser liberados la organizacion ha sufrido cambios, los cuales no siempre se consideran. La mayoría del software corporativo es obsoleto por los cambios a los que están expuestas las organizaciones que los requieren.

De la necesidad de buscar y aplicar nuevos enfoques para la construcción de software para los sistemas de información, la Ingeniería de Sistemas pretende mediante metodologías innovadoras, como lo es la Orientada a Objetos, crear sistemas lo mayormente confiables, flexibles, mantenibles y capaces de evolucionar para satisfacer los requerimientos del cambio de cualquier organización, además de minimizar el tiempo de desarrollo invertido en su construcción, creación de librerías y código reutilizable, entre otros.

En esta tesis se conocerán, comprenderán y analizarán las metodologías orientadas a objetos de la ingeniería de software, además se efectuará una evaluación comparativa de ellas para concluir si los objetivos pretendidos por estas metodologías son funcionales, los cuales son:

- Permitir el desarrollo de Sistemas de software de una forma más dinámica, que respondan al ambiente en constante evolución de las organizaciones.
- Cambiar la forma en que se realiza el modelado de algún problema, mediante nuevos enfoques de abstracción en la identificación de los elementos que integran un sistema y su interacción.
- Proveer una rica notación para la documentación del análisis y diseño de los sistemas de información.
- Promover el uso de técnicas para la obtención de componentes reutilizables.

Este estudio se desarrollará de la siguiente forma:

La primera parte de la tesis está enfocada a dar los conceptos básicos necesarios para entender la Ingeniería de Software orientada a objetos. El primer capítulo está dedicado al estudio de los sistemas

procedimentales y sus metodologías, y en contra parte el segundo capítulo se dedica al estudio de los sistemas orientados a objetos (surgimiento, conceptos y características). De esta forma en el tercer capítulo se tienen los elementos necesarios para realizar un análisis comparativo de los Sistemas Procedimentales y los Orientados a Objetos y se establecen ventajas y desventajas de los mismos.

La segunda parte se enfoca al estudio, análisis detallado y a la evaluación de las metodologías de análisis y diseño orientadas a objetos.

En los capítulos cuarto, quinto y sexto se estudian, comprenden y analizan tres de las más completas metodologías Orientadas a Objetos de la Ingeniería de Software:

- Metodología de Wirfs
- Metodología de Yourdon
- Metodología de Booch

En el capítulo séptimo se aplican cada una de las metodologías a un caso real, será el mismo para las tres, la aplicación es hasta el diseño.

Finalmente en las conclusiones se realiza una evaluación comparativa de las metodologías estudiadas.

CONTENIDO

Página

Página

Página

Página

Página

Página

PARTE I

1. SISTEMAS PROCEDIMENTALES

1.1 Reseña Histórica

En la Historia de las computadoras de electrónica digital los años 50's y los 60's fueron décadas de hardware. Los años 70's fueron un periodo de transición y un tiempo de reconocimiento del software. Desde la década de los 80's el software ha sido reconocido como el elemento que determina la capacidad de los nuevos procesadores de nuestra era, es decir, las imponentes capacidades de procesamiento y almacenamiento del hardware moderno representa un gran potencial de cálculo, el software es el mecanismo que nos facilita utilizar y explotar ese potencial.

Desarrollar software que iguale el potencial de las computadoras resulta ser un reto mucho mayor que el construir máquinas más rápidas

La Programación Modular forma el principio en que están basados la mayoría de los avances en la construcción del software en los últimos 40 años.

El soporte más elemental para la programación modular se dió con la invención de la "subrutina" a principios de los 50's. Una subrutina se crea al sacar una secuencia de instrucciones del programa principal y darle un nombre separado. Mientras que las subrutinas proporcionan el mecanismo básico para la programación modular, se requiere mucha disciplina para crear software bien estructurado.

A finales de los 60's, el estado del software disparó un esfuerzo concertado entre los científicos de la computación para desarrollar un estilo de programación consistente y disciplinado. El resultado de ese esfuerzo fue el refinamiento de la programación modular, es el enfoque conocido como "Programación Estructurada", que se basa en la descomposición funcional; al descomponer sistemáticamente un programa en componentes, cada uno se descompone en subcomponentes y así consecutivamente hasta el nivel de subrutina individuales. De esta manera, grupos de programadores separados escriben diferentes componentes, los que se ensamblan posteriormente para formar el sistema completo.

Lourdes Montiel García
Isabel Perez Arredondo

La última innovación en la programación estructurada es la "Ingeniería de Software Asistida por Computadora" (CASE). Con CASE las computadoras administran el proceso de la descomposición funcional, verificando que todas las interacciones entre subrutinas sigan una forma correcta y específica. De hecho, los sistemas avanzados de CASE pueden construir programas completos a partir de diagramas en los que se expresa toda la información del diseño. El CASE suministra un soporte automático o semiautomático en la aplicación de los métodos en el desarrollo de sistemas, las metodologías orientadas a objetos también son contempladas en CASE.

Otro enfoque de programación automática lo representan los "Lenguajes de Cuarta Generación" (4GL's). Los 4GL's incluyen una amplia variedad de herramientas que ayudan a automatizar la generación de aplicaciones típicas de negocios, incluyendo la creación de formas, reportes y menúes.

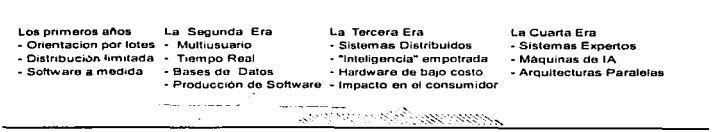
1.2 Evolución de los Sistemas de Software

Es común, al hablar de la comunicación, señalar el vertiginoso e impresionante progreso logrado en las pocas décadas de su existencia. El contexto en el que se ha desarrollado el software está fuertemente ligado a las cuatro décadas de evolución de los sistemas informáticos. Un mejor rendimiento del hardware, un tamaño más pequeño y un costo más bajo, han dado lugar a sistemas informáticos más sofisticados.

La evolución del software dentro del contexto de las áreas de aplicación de los sistemas basados en computadora se describe en la siguiente figura:

Lourdes Montiel García
Isabel Pérez Arredondo

Evolución del Software



Durante los primeros años el hardware era de propósito general, por otra parte el software se diseñaba a la medida para cada aplicación y tenía una distribución relativamente pequeña.

La mayoría del software se desarrollaba y utilizaba por la misma persona u organización; debido a este entorno personalizado del software, el diseño era un proceso implícito, ejecutado en la cabeza de alguien y la documentación no existía normalmente.

La segunda era de la evolución de los sistemas de computadoras se extiende desde la mitad de la década de los 60's hasta finales de los 70's. La multiprogramación, los sistemas multiusuarios, introdujeron nuevos conceptos de interacción hombre-máquina. Las técnicas interactivas abrieron un nuevo mundo de aplicaciones, los sistemas podían reconocer, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en vez de minutos.

La tercera era de la evolución de los sistemas de computadoras comenzó a mediados de los 70's. El sistema distribuido (en este caso computadoras múltiples, cada una ejecutando funciones concurrentes y comunicándose con alguna otra) incrementó notablemente la complejidad de los sistemas informáticos.

Redes de área local y global, comunicaciones digitales de alto ancho de banda, la llegada y amplio uso de los microprocesadores y computadoras personales, supusieron una fuerte presión sobre los desarrolladores de software.

Lourdes Montiel García
Isabel Perez Arredondo

La cuarta era en software de computadoras ha comenzado. Las técnicas de la cuarta generación para el desarrollo de software están cambiando la forma en que algunos segmentos de la comunidad informática construye los programas de computadora. Los sistemas expertos y el software de inteligencia artificial se han transladado finalmente del laboratorio a aplicaciones prácticas, en un amplio rango de problemas del mundo real.

Es así como el software se ha convertido en el elemento clave de la evolución de los sistemas y productos informáticos.

1.3 Desarrollo de Sistemas Orientados a Procedimientos. Metodologías

El proceso de desarrollo de software contiene tres fases genéricas: Definición, Desarrollo, Mantenimiento; estas fases no difieren de un sistema a otro, son independientes al área de aplicación, del tamaño del proyecto o de la complejidad que éste implique.

La fase de ***Definición*** se enfoca al qué. Esto es, durante la definición se identifica qué información ha de ser procesada, qué función, qué interfaces han de establecerse, qué ligaduras de diseño existen y qué criterios de validación se necesitan para definir un sistema. Por lo cual se debe identificar los requerimientos del sistema y del software. Al realizarse la definición se llevarán a cabo las siguientes tareas:

Análisis del Sistema. Se define el papel de cada elemento que interviene en el sistema que se desee realizar por un software.

Planificación del Proyecto Software. Se definen los limites del sistema; se realizan estimaciones de los recursos humanos y materiales (entre ellos el hardware y software) con sus respectivos costos; se definen tareas a realizar y se elabora el plan de trabajo.

Análisis de requerimientos. Se realiza un estudio más detallado del dominio de la información que se maneja en el sistema y se define cuál va a ser manipulada por el software.

Lourdes Montiel García
Isabel Pérez Arredondo

La fase de ***Desarrollo*** se enfoca al cómo. En esta fase se diseñan las estructuras de datos, la estructura del software, cómo han de implementarse los detalles procedimentales, la construcción del software en el lenguaje de programación y cómo han de realizarse las pruebas del software terminado.

Durante el desarrollo se llevarán acabo las siguientes actividades:

Diseño del Software. El diseño translada los requerimentos del software a un conjunto de representaciones (algunas gráficas, otras tabulares o basadas en lenguajes) que describen la estructura de datos, arquitectura y procedimiento algorítmico.

Codificación. Las representaciones del diseño deben transladarse a un lenguaje de programación, que dá como resultado unas instrucciones ejecutables por computadora (software).

Prueba de Software. Una vez que el software se ha implementado en una forma ejecutable por la máquina, deben ser probados para verificar su buen funcionamiento o en su caso detectar errores.

La fase de ***Mantenimiento*** se enfoca al cambio y/o correción del software, estos cambios pueden ser ya sea por adaptaciones requeridas por la evolución del entorno del software o por cambios de los requerimientos de los usuarios del sistema, ya sea para reforzar o aumentar los alcances del mismo. La fase de mantenimiento reaplica los pasos de las fases de definición y desarrollo, pero basándose en el software ya existente. Durante la fase de mantenimiento se encuentran tres tipos:

Corrección. Modificación del software para corregir errores.

Adaptación. Con el paso del tiempo es posible que cambie el entorno original (CPU, periféricos, sistema operativo, entre otros) para el cual se desarrolló el software, lo que provoca realizar una adaptación al medio a nivel de programación.

Aumento. Debido al entorno de las organizaciones, los procesos y sistemas de información cambian para daptarse al medio, lo que provoca que el software crezca o se modifique para ir al paso con el cambio y siga satisfaciendo las necesidades de información que lo crearon.

En la ingeniería de software un método es un procedimiento o una técnica que realiza una porción significante del ciclo de vida de un sistema. Una metodología es una colección de métodos basados en una filosofía común que juntos cubren parte o todo el ciclo de vida de un sistema.

Los métodos aplicados durante las fases de definición, desarrollo y mantenimiento variarán dependiendo de la(s) metodología(s) de Ingeniería de Software (o en su caso la combinación de ellas) que se aplique(n).

Los métodos abarcan la planificación y estimación de proyectos, análisis de los requerimientos del sistema y del software, diseño de las estructuras de datos, arquitecturas de programas y procedimientos algorítmicos, prueba y mantenimiento.

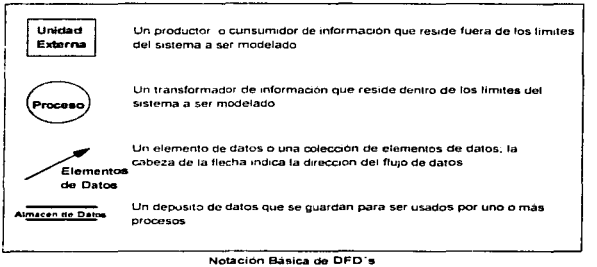
Algunas de las metodologías más conocidas para el desarrollo de sistemas en la Ingeniería de Software son:

- Método Warnier-Orr (Desarrollo Estructurado de Sistemas de Datos)
- Método Gane/Sarson
- Método Entidad-Relación
- Método Yourdon
- Método de Ingeniería de Información (IEM)
- Método DeMarco
- Método Jackson
- CASE Method

1.3.1 Diagramas de Flujo de Datos (DFD)

Debido a que la mayoría de los métodos de Ingeniería de Software utilizan los Diagramas de Flujo de Datos (DFD) se definirá lo qué son y su notación .

El Diagrama de Flujo de Datos es una técnica gráfica que representa el flujo de la información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida. Se puede usar un Diagrama de Flujo de Datos para representar un sistema o un software o cualquier nivel de abstracción. De hecho los DFD's pueden ser refinados en niveles que representen un mayor flujo de información y un mayor detalle funcional. Un diagrama de nivel 0 (cero) también es denominado modelo fundamental del sistema o modelo de contexto, representa al elemento de software completo como una sola burbuja con datos de entrada y salida representados por flechas de entrada y salida, respectivamente. Partiendo del DFD nivel 0 para mostrar más detalles, aparecen DFD's adicionales. La notación básica que se usa para crear un DFD es la siguiente:

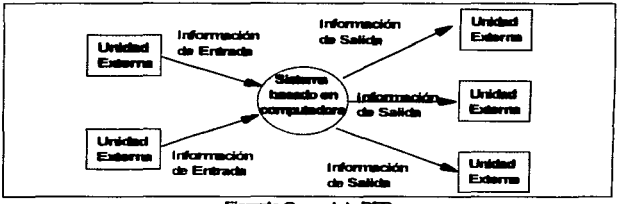


Notación Básica de DFD's

Es importante señalar que el diagrama no proporciona ninguna indicación explícita de la secuencia de procesamiento. El procedimiento o la secuencia puede estar

Lourdes Montiel García
Isabel Perez Arredondo

implícitamente en el diagrama, pero la representación procedimental explícita, queda hasta el diseño del software.



Ejemplo General de DFD

1.3.2 Método Warnier- Orr

El nombre correcto de lo que mucha gente llama metodología Warnier/Orr es: Desarrollo Estructurado de Sistemas de Datos (DSSD Data-Structured Systems Development) y es el resultado del trabajo conjunto de mucha gente.

Warnier desarrolló una notación para representar la jerarquía de la información para las tres construcciones de secuencia, selección y repetición, y demostró que la estructura del software puede derivarse directamente de la estructura de los datos.

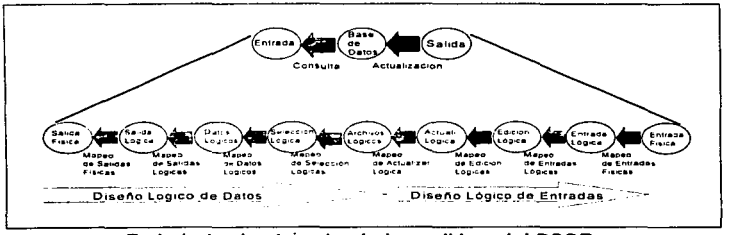
Ken Orr amplió el trabajo de Warnier para abarcar una visión algo más amplia del campo de información, evolucionando al desarrollo de sistemas estructurados en datos. El DSSD considera el flujo de la información y las características funcionales, así como la jerarquía de datos.

Poco a poco DSSD de metodología de diseño de programas pasó a metodología de diseño de sistemas. Después la metodología fue expandida para abarcar el diseño de

Lourdes Montiel García
Isabel Pérez Arredondo

bases de datos, definición de requerimientos y finalmente la planeación y arquitectura de los sistemas.

A nivel conceptual DSSD aún contiene rasgos característicos a nivel de programación. Por ejemplo, aún se enfoca (en su fase de diseño) a trabajos hacia atrás desde las salidas. DSSD trabaja hacia atrás hasta la base de datos lógica y luego hacia las entradas. La base de datos lógica se transforma en una base de datos relacional y normalizada.



Trabajo hacia atrás desde las salidas del DSSD

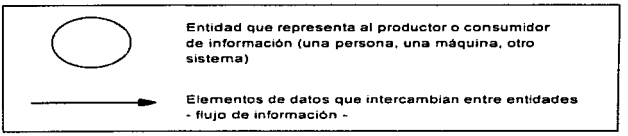
Mientras que una definición completa de los requerimientos (salidas más algoritmos) es un excelente punto de partida en el diseño, no es propiamente el punto desde el cual empezar la definición de los requerimientos. Así a través de los años, DSSD se ha extendido hasta cubrir primero el contexto, luego las funciones y finalmente los resultados del sistema.

Se hicieron necesarios un número de herramientas para facilitar este proceso. Los diagramas de Entidades ayudan a definir el contexto del sistema, y los diagramas de línea de montaje (una forma modificada de los diagramas Warnier/Orr) ayudan a definir el flujo funcional del sistema.

Lourdes Montiel García
Isabel Perez Arredondo

1.3.2.1 Diagramas de Entidades.

El diagrama de entidades utiliza una notación muy parecida al diagrama de flujo de datos. Los símbolos son similares pero tienen diferentes significados:



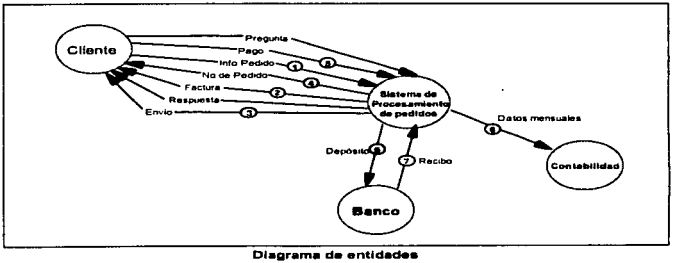


Diagrama de entidades

1.3.2.2 Diagrama de Línea de Montaje.

Es un mecanismo para acoplar la información y los procesos (transformaciones o funciones) que se aplican. Conceptualmente el Diagrama de línea de montaje desempeña el mismo papel que el diagrama de flujo de datos.

Un diagrama de línea de montaje se desarrolla partiendo del flujo de información. Se parte del último flujo que se tiene definido y se continúa hasta llegar con el primer dato identificado (obsérvese el diagrama de entidades).

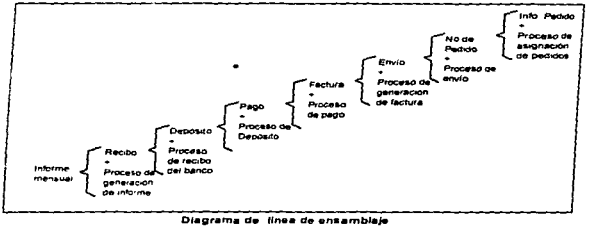


Diagrama de línea de ensamblaje

1.3.3 Método Gane / Sarson

Este método está basado en el modelado lógico el cual consiste en tomar las ideas vagas necesarias acerca de los requerimientos y convertirlas en definiciones precisas tan pronto como sea posible, este proceso puede realizarse rápidamente si se cuenta con una técnica gráfica que nos permita tener la esencia del sistema sin ir al problema de la implementación física, con esto pudiera hacerse por ejemplo un prototipo.

El modelado lógico puede verse como un proceso de 7 pasos:

1. Desarrollo de un Diagrama de Flujo de Datos (DFD) general, que describa lo que ocurre en cada área de la empresa. Para simplificar los diagramas se explican solamente cuatro símbolos que producen un diagrama mostrando la naturaleza lógica de detalle. Los DFD nos muestran los límites de un área dentro del sistema y de la empresa misma, los diagramas no son técnicos, es decir, son fáciles de entender por cualquier persona

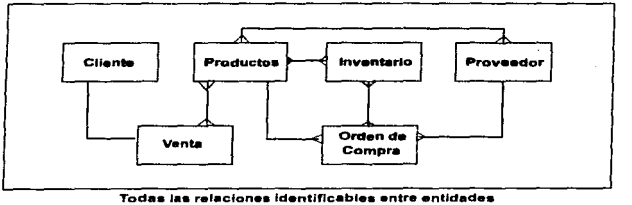
Lourdes Montiel García
Isabel Perez Arredondo

familiarizada con el área descrita, nos muestran datos almacenados y los procesos que transforman esos datos.

2. Derivar una lista de los datos que están en cada almacén de datos, la lista puede afinarse revisando los datos de entrada/salida al sistema y determinando qué representa cada dato.

3. Revisar lo que el análisis entidad-relación puede decir acerca de la estructura de los datos. Obtener las entidades y crear un diagrama con las entidades que se han definido, después buscar las relaciones existentes entre las entidades y dibujar líneas que muestren estas relaciones.

Por cada par de entidades se tiene una relación entre sus elementos, la relación puede ser uno a uno, muchos a uno o muchos a muchos. Estos diagramas nos dan mucha información acerca del sistema, mostrándonos todas las relaciones que existen entre las entidades involucradas.



Todas las relaciones identificables entre entidades

4. Emplear la información de los datos para describir un modelo que haga una liga entre dos tablas bidimensionales. Las tablas deberán estar normalizadas, es decir, la información deberá integrarse en tablas bidimensionales, las cuales se estructurarán como se requieran, quedando en:

a)1a. Forma Normal, una tabla debe contener una llave primaria, la cual debe identificar a el (los) atributo(s) del mismo renglón, esto implica que el o los atributo(s) dependan de la llave primaria.

b) 2a. Forma Normal, es semejante a una tabla en la primera forma normal, pero si la llave primaria es compuesta por dos campos de la misma tabla, esta llave compuesta identifica a los atributos del mismo renglón, lo cual implica que el o los atributos dependan de la llave primaria compuesta.

c) 3er. Forma Normal, la llave primaria está integrada de dos campos de la misma tabla, igual que la 2a forma normal, pero se está en tercera forma normal si el valor del primer atributo no depende del valor del segundo atributo y viceversa, a menos que se indique lo contrario.

5. Redibujar el DFD para reflejar una vista más precisa del sistema de datos como resultado del análisis del modelo entidad-relación y de la normalización.

6. Particionar este modelo lógico de procesos y datos dentro de unidades de procedimientos, separar los procedimientos que puedan ser ejecutados manual o automáticamente.

7. Especificar el detalle de cada unidad de procedimiento que se requiere para implementar el sistema: Extraer la unidad del DFD y mostrar dónde queda dentro del sistema, detallar las tablas accedidas por esa unidad, mostrar un esquema de pantallas y reportes involucrados en la unidad y detallar la lógica y los procedimientos que serán implantados.

Una vez definidos los procedimientos se puede decidir si será un prototipo o se debe implantar directamente en algún lenguaje.

Los pasos 6 y 7 no son estrictamente hablando un modelado lógico dado que tratan de convertir el modelo logico en modelo físico, sin embargo son parte del flujo natural a

través del proceso que comienza en la defincion de sistemas y termina con el diseño fisico.

1.3.4 Método Entidad-Relación

Es una metodología estructurada que puede sistemáticamente convertir los requerimientos del usuario en una base de datos bien diseñada. No es propiamente una metodología sino una aproximación.

Este método se basa en:

- Desarrollar un diagrama entidad-relación (ERD Entity Relational Diagram). Este paso identifica Entidades, Relaciones y atributos asociados además de la llave primaria de cada tipo de entidad.

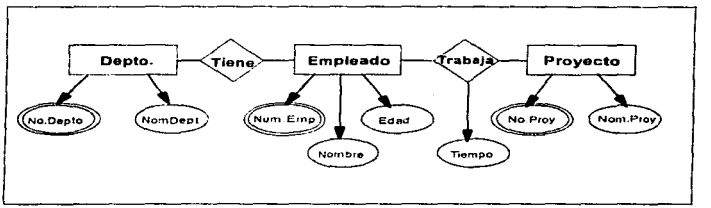


Diagrama Entidad Relación

Una entidad es una cosa, un concepto, una organización o un evento de interés para la organización que se está modelando. Una Entidad Tipo es una clasificación de entidades que satisfacen ciertos criterios. Una relación es una interacción entre entidades. Una Relación Tipo es una clasificación de relaciones basada en cierto criterio. Usualmente los sustantivos corresponden a las entidades mientras que los verbos corresponden a las relaciones.

El siguiente paso es identificar la cardinalidad de los tipos de relaciones, es decir con cuántos elementos de una entidad B se puede relacionar un elemento de una entidad A y viceversa. Se identifican después las propiedades (atributos) de cada Entidad - Relación y se expresarán gráficamente con círculos (o elipses). Las llaves primarias son identificadas con un doble círculo.

- Convertir los diagramas entidad-relación en archivos convencionales y estructuras de bases de datos.

- Desarrollar los programas de aplicación basados en los archivos y estructuras de la base de datos. Si se está usando un Sistema Manejador de Bases de Datos (DBMS) es posible escribir en este momento un programa en SQL (System Query Lenguage) para expresar la consulta a los datos.

El modelado Entidad-Relación puede ser usado no sólo como una herramienta de diseño sino como la base para el modelado en sistemas manejadores de bases de datos.

1.3.5 Método Yourdon (Orientado a Sistemas Estructurados)

El método Yourdon es una colección genérica de ideas de ingeniería de software desarrolladas por muchas personas quienes han trabajado para la compañía Yourdon. A las ideas de todas estas personas que se reunieron se les dió el nombre de técnicas estructuradas: análisis, diseño y programación estructurados, con esto Ed Yourdon intenta enlazar su metodología a las demás etapas del proceso de desarrollo de un sistema, para crear una metodología que compita con los métodos que sí cubren completamente el ciclo de vida de un sistema, es decir, los métodos que inician con la etapa de estrategia y análisis, diseño lógico y físico hasta la implementación, transición y producción.

Debido a la continua influencia de nuevas ideas de gente nueva, el método Yourdon está en continuo desarrollo, de tal forma que actualmente el método Yourdon es

muy diferente del de hace 10 años, ahora se han incorporado las mejores ideas de análisis y diseño orientado a objetos.

El método que se describe en esta parte es el que está orientado a los sistemas estructurados.

Este método consta de dos partes: herramientas y técnicas. Las herramientas son una variedad de diagramas gráficos usados para modelar los requerimientos y la arquitectura de un sistema de información. El más familiar de estos diagramas es el diagrama de flujo de datos (DFD), este diagrama ha sido extendido para soportar sistemas de tiempo-real.

"Un sistema computacional de tiempo real puede definirse como aquel que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rápidez como para influir en dicho ambiente en ese momento", (Martin, 1967).

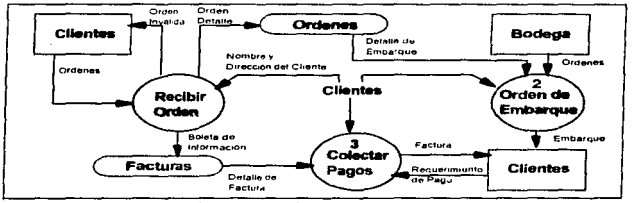


Diagrama de Flujo de Datos

Este método hace uso de diagramas entidad-relación (ERD's) y diagramas de transición de estado (STD's State Transition Diagrams) para facilitar la construcción de la base de datos de un sistema. Para terminar de definir los requerimientos de un sistema pueden emplearse las cartas de estructura para mostrar la organización de los módulos que serán implementados.

Lourdes Montiel García
Isabel Pérez Arredondo

Para complementar la descripción del sistema es necesario un soporte textual adicional, un diccionario de datos que nos describe cómo está compuesto cada elemento y un conjunto de especificaciones de procesos que nos describa el comportamiento de los procesos en cada nivel.

La técnica del método de Yourdon consta de guías que nos llevan de la nada a un modelo del sistema bien organizado. Está técnica consiste en particionar (top-down) el sistema en funciones. Actualmente el método usa una técnica conocida como participación de eventos, inicia dibujando un diagrama de contexto el cual permite definir los alcances del sistema y las interfaces con fuentes externas. Después de las entrevistas al usuario se escriben una serie de eventos que ocurren en el medio ambiente externo y a las cuales el sistema responde, esto nos permite dibujar un primer diagrama de flujo de datos. Por ejemplo, si un sistema consta de 100 eventos tendrá 100 burbujas en el DFD, dado lo anterior es posible particionar los eventos en varios y unirlos en un solo proceso, esta técnica es parecida al diseño orientado a objetos.

El Diseño Estructurado Yourdon consta de 4 pasos:

1. Diagrama de Flujo de Datos. Muestra los sistemas como un conjunto de procesos que transforman datos. Los diagramas de flujo de datos nos muestran la relación entre los procesos y el flujo de información entre ellos. Los requerimientos en tiempo real son mostrados en los Diagramas de Flujo de Datos vía el control de flujo y el control de procesos.

2. Evaluacion de Estructura. La carta de estructura son una jerarquía de unidades procedurales que documentan el control del programa, el flujo de información entre programas y la lógica del programa. Se emplean las técnicas de Análisis de Transformación y de Análisis de Transacción.

3. Evaluación del Diseño. En este se mide la calidad del diseño, para ello son empleados los criterios de acoplamiento y cohesión.

Lourdes Montiel García
Isabel Pérez Arredondo

4. Preparación para la Implantación. La lógica diseñada del programa es empaquetada en unidades de implantación física, llamadas unidades de carga.

1.3.6 Método de Ingeniería de Información (IEM)

El método de Ingeniería de Información sigue un enfoque top-down, desarrollando modelos en cada una de las etapas del ciclo de vida que cubre (estrategia, análisis y diseño), forzando el desarrollo de sistemas al manejo de metas y planes de estrategias.

IEM ofrece modelos de alto nivel para la representación total de una empresa durante la etapa de estrategia y diagramas de estructuras de datos detallados (entidades, relaciones, atributos), descomposición funcional, flujo de datos y diagramas de dependencia, y definición de reportes y pantallas durante el "análisis de las necesidades de la empresa". Estos diagramas muestran detalladamente la lógica del programa, incluyendo el manejo procedural y criterios de selección durante subsecuentes diseños y construcciones de sistemas. Matrices para el mapeo de las especificaciones del análisis y del diseño, para el manejo de metas, conjuntando los requerimientos de datos y procesos, y asegurando una terminación e integración de estos a nivel de aplicación.

1.3.7 Método DeMarco

DeMarco es un método de análisis estructurado orientado a procesos. Es utilizado para definir los requerimientos de un sistema en las etapas de análisis y diseño. Durante la etapa de análisis, DeMarco obtiene todo el conjunto de información manejada por los procesos actuales y los nuevos requerimientos de ésta y los utiliza para obtener la descripción de la funcionalidad del sistema a construir. Utiliza diagramas de flujo de datos como técnica de representación.

Pasos que sigue:

1. Construir el modelo físico actual (DFD's)
2. A partir del modelo físico, construir el modelo lógico actual (DFD's)

3. Construir el modelo lógico del nuevo sistema a desarrollar incluyendo DFD's, definiciones del diccionario de datos y especificaciones de los nuevos procesos.
4. Crear una familia de nuevos modelos físicos (DFD's)
5. Obtener el costo y tiempos estimados para cada modelo
6. Seleccionar el modelo más adecuado

1.3.8 Método Jackson

Jackson es una técnica de diseño orientada a datos, la cual se basa en las estructuras de datos requeridas para obtener la estructura de los procesos. Empieza asumiendo que la etapa de análisis ha sido terminada.

En este método el sistema es visto como un conjunto de procesos, los cuales se adecuan al manejo de las estructuras de datos, las cuales se consideran estáticas. Utiliza básicamente dos técnicas de diagramación: diagramas de redes, en los cuales se muestra el flujo de información a través de los procesos, y diagramas arborescentes, en los cuales se muestran las relaciones jerárquicas entre procesos y datos.

Pasos que sigue:

1. Dibujar los diagramas de estructura de árbol mostrando el flujo de datos entre cada proceso
2. Tomar todas las estructuras anteriores y formar un solo programa
3. Mostrar las operaciones necesarias para obtener la información de salida de la información de entrada y asignar cada una de éstas a los componentes del programa (procesos)
4. Transcribir el programa a un texto estructurado, agregando tanto lógica de selección como ciclos.

1.3.9 CASE*Method

CASE es el acrónimo de Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadora).

CASE se enfoca a la parte de las herramientas de la Ingeniería de Software, es decir, el CASE sirve para facilitar el uso de los métodos de la ingeniería de software y que los procedimentos que se definan no sean tan complejos.

El CASE surge en los años 80's a raíz de la amplia necesidad de hacer sistemas más eficientes, es decir, que además de satisfacer todas las necesidades que los crearon , permita su estructura el crecimiento del mismo sin necesidad de replantear todo el sistema para lograrlo. Frecuentemente al querer realizar un cambio al sistema se debía contar con el personal que los había creado, debido a que regularmente no se documentaba debidamente al sistema en su creación, y si el personal que los había hecho dejaba la compañía, prácticamente resultaba imposible modificar el sistema. Basándose en esto surge la herramienta CASE como tal y surge el concepto de metodología CASE.

La metodología CASE reconsidera lo que se denomina ciclo de vida de los sistemas y a diferencia de los que se hacía antes dá un gran peso a las fases de planeación y análisis considerando que si esas fases se realizan correctamente las demás etapas no tendrán problemas, a continuación se plantea lo que se denomina CICLO DE VIDA CASE:

70% Planeación
Análisis — UPPER CASE

30% Diseño — LOWER CASE
L4G
Implementación y Mantenimiento

Como se observa le da un peso muy importante a las primeras fases, este peso se refleja en el tiempo que se dedica a esta parte en la realización de un sistema, un sistema bien planeado y correctamente analizado no tendrá problemas o por lo menos no tanto como los que se presentaban cuando esto no se realizaba, otra de las ventajas de CASE es que es capaz de generarnos código y con la ayuda de los L4G (Lenguajes de Cuarta Generación) el trabajo de codificación en la fase de desarrollo se hace realmente sencillo y para realizarlo ya no requerimos de tantos programadores que a su vez tengan que

Lourdes Montiel García
Isabel Pérez Arredondo

diseñar, esto también era un problema, con la ayuda de estos lenguajes de cuarta generación se hace más económica esta etapa.

Divide además estas etapas en lo que se denomina UPPER CASE y LOWER CASE , es decir para CASE la parte fuerte es la planeación y el análisis, mientras que los sencillo es el diseño y la implementación.

Con la metodologia CASE surgen dos conceptos muy importantes en el diseño de sistemas:

REINGENIERIA, se refiere al hecho de que cuando se requiere hacer una ampliación del sistema, se parte desde la fase de análisis pero considerando todo lo que ya se ha hecho, es decir, hacer análisis del análisis anterior.

INGENIERIA DE REVERSA, plantea que si se requiere modificar el sistema e implantar algo nuevo, no es necesario empezar con el análisis ya que el primer análisis debió contemplar este crecimiento, por lo cual sólo se regresa hasta la etapa en la cual puedo implantar el cambio, siendo esta fase casi siempre el diseño, en caso de que tuviere que regresarse hasta la fase de análisis diremos que el primer análisis fue incorrecto.

2.SISTEMAS ORIENTADOS A OBJETOS

2.1 Surgimiento de los Sistemas Orientados a Objetos

De la necesidad de buscar y aplicar nuevos enfoques para la construcción de software de sistemas de información, surgen las metodologías orientadas a objetos. Entendiéndose por Orientado a Objetos como la técnica de construcción de sistemas basada en la combinación de componentes con límites bien establecidos, que enfatiza la creación de bibliotecas de componentes reutilizables.

El desarrollo de sistemas orientados a objetos tienen un doble propósito: por un lado la elaboración de herramientas que solucionan la problemática planteada, y por otro, la generación de código que puede ser integrado a una biblioteca de componentes, aumentando así el capital de software.

Los conceptos de los sistemas orientados a objetos surgen en la década de los 60's con los lenguajes de programación orientados a objetos. El precursor de estos lenguajes fue SIMULA67, nació en Noruega, donde un grupo de investigadores encabezados por Dahl y Nygaard, buscaba modificar a los lenguajes ya existentes, diseñados para aplicaciones numéricas, con el objetivo de hacerlos aptos para programar las simulaciones discretas de los problemas del mundo real. Los problemas a simular se componían de objetos que cooperaban o interactuaban de forma variada entre sí. Los noruegos extendieron a ALGOL-67 con los conceptos de objetos, clases de objetos y jerarquías de herencias entre clases, que caracterizan hoy en día a la Programación Orientada a Objetos.

Varios años después, las ideas nacidas en SIMULA se adoptaron en el diseño e implementación del lenguaje SMALLTALK a finales de los años 70's. SMALLTALK fue desarrollado por un grupo de investigadores de los Laboratorios Xerox, de Palo Alto, CA, en los Estados Unidos, bajo la dirección de Andrew Kay, en el proyecto conocido como Dynabook, cuyo objetivo fue crear una interfaz muy amigable para una computadora personal. El lenguaje SMALLTALK resultado de este esfuerzo, retomó los conceptos de

objeto, clase y herencia de SIMULA y los conjugó con la flexibilidad de la programación interactiva. Al popularizarse SMALLTALK se impuso una nueva terminología, apareció el término Orientado a Objetos.

Pero no fue sino hasta después de una década del nacimiento de SIMULA que los conceptos presentes en el lenguaje tuvieron un amplio impacto. Esto se dió por las siguientes razones: Durante la década en el que la Programación Orientada a Objetos estuvo en estado latente, la cultura computacional tuvo avances importantes, en particular en las áreas de software de base (compiladores, sistemas operativos, etc.) y en la teoría de la computación (semántica de lenguajes de programación, teoría de tipos, etc.). Estos avances en la computación, aunados a las fuertes demandas impuestas por las aplicaciones de finales de los años 70's, ya que no eran satisfechas por los lenguajes contemporáneos y, en especial, a la fuerte publicidad que se le hizo al lenguaje SMALLTALK, hicieron que éste corriera una suerte totalmente distinta a la del lenguaje SIMULA, nacido en una época antes de su tiempo, en un país pequeño, alejado de grandes centros de investigación y de los apoyos de la industria de las computadoras.

La segunda mitad de los 80's y 90's se han distinguido por una gran cantidad de propuestas de nuevos lenguajes orientados a objetos. Entre los más importantes se encuentra la extensión del lenguaje C, conocida como C++ de Stroustrup: el más apegado a la herencia de SIMULA, EIFFEL de Meyer, y una extensión orientada a objetos de LISP, conocida como CLOS. También, cabe hacer resaltar las distintas modificaciones de Pascal, como OBJECT Pascal o recientes versiones de TURBO Pascal.

2.1.1 SMALLTALK

Smalltalk fue creado por los miembros del Xerox Palo Alto Research Center Learning Research Group como un elemento de El Dynabook, siendo un proyecto de Alan Kay. Smalltalk tiene una gran influencia del lenguaje SIMULA, sin embargo también toma ideas del lenguaje FLEXY del trabajo de Seymore Papert and Wallace Feurzeig. Smalltalk representa un lenguaje y un ambiente de desarrollo de aplicaciones. Es una lenguaje "puro" orientado al objeto, ya que visualiza todo como un objeto. Smalltalk es quizás el

Lourdes Martínez García
Isabel Pérez Arredondo

más importante lenguaje orientado a objetos porque su conceptualización influye a casi todos los lenguajes orientados a objetos posteriores a él, además de contar con una interface gráfica con el usuario, semejante a la de Motif o la interface de Macintosh.

Smalltalk implica más de una década de trabajo. Dan Ingalls fue el líder durante el desarrollo de Smalltalk, también contribuyeron para su desarrollo Peter Deutsch, Glenn Krasner y -kim McCall. En paralelo, los elementos de ambiente de Smalltalk fué desarrollado por James Althoff, Robert Flegal, Ted Kachler, Diana Merry y Steve Putz.

El desafío de Smalltalk fue identificar y crear una forma lo suficientemente simple y poderosa para permitir a una persona tener acceso y crear el control adecuado y versátil sobre la información, sin importar que sea texto o números, ésto mediante imágenes y sonidos. Smalltalk está basado en dos conceptos: Todo es un objeto, y los objetos se comunican mediante el envío de mensajes.

2.1.2 C++ de Stroustrup

C++ fue diseñado e implementado por Bjarne Stroustrup de los Laboratarios Bell AT&T. El antecesor inmediato de C++ es un lenguaje llamado C con Clases, también desarrollado en 1980. En su momento C++ fue fuertemente influenciado por lenguajes como C y SIMULA. La diferencia de C++ y C es que además de contar con más funciones C++ es orientado a objetos.

En un inicio C++ fue hecho para correr en arquitectura del Sistema Operativo Unix, pero a la fecha C++ y sus compiladores nativos del mismo son compatibles en casi todas las arquitecturas que existen.

C++ no solamente corrige la mayoría de las deficiencias de C, sino que introduce muchas características completamente nuevas que fueron diseñadas para dar al lenguaje el soporte para la abstracción de datos y la programación orientada a objetos. A continuación se enumeran las más importantes características que se han introducido:

- Clases, las construcciones básicas del lenguaje para la creación de tipos de datos definidos por el programador.
- Variables miembro, las cuales describen los datos en Tipos de Datos Abstractos.
- Funciones miembro, las cuales definen las operaciones permitidas en los tipos de datos.
- Conversión automática de tipos controlada por el programador, lo cual permite combinar los tipos propios con otros y los tipos de datos básicos ofrecidos por el lenguaje C++.
- Clases derivadas, las cuales heredan las variables miembro y funciones miembro de la clase base y pueden ser diferenciadas de ella añadiendo otras variables y funciones miembro.
- Funciones virtuales, las cuales permiten a una clase derivada redefinir funciones miembros de la clase base. Así podemos escribir programas muy generales olvidándonos de las clases específicas de los objetos que manipulan; por medio del ligado automático el sistema elegirá la función apropiada para una clase particular.

2.1.3 EIFFEL

Eiffel fue diseñado por Bertrand Meyer, quien fue una vez Presidente de la Asociación de Usuarios de SIMULA.

En 1985 llegó la ocasión de buscar un ambiente de desarrollo para Interactive Software Engineering y habiéndose dado cuenta que SIMULA no era una solución viable y estando escéptico en cuanto a la posibilidad de utilizar C, decidió diseñar una nueva versión de SIMULA, la cual finalmente se llamó Eiffel. Esta versión es la más moderna de SIMULA, liberada de algunos rasgos innecesarios y mejorada por la experiencia acumulada durante 10 años aproximadamente.

En Eiffel donde la mayoría de la atención fue dedicada a conservar el lenguaje pequeño y tratar de hacerlo elegante, no se encuentran conceptos generalmente

aceptados como variables globales, tipos enumerados, subrangos de tipos, instrucciones goto (así como la mayoría de las cláusulas que han sido inventadas para ellos a través de los años, como "exit", "break", "continue" y otras), conversiones forzadas de tipos "casts", aritmética de punteros, entrada y salida definida por el lenguaje y algunas otras.

La exclusión de estos conceptos no significa que todos ellos sean intrínsecamente malos, algunos son simplemente redundantes o incompatibles con los mecanismos que fueron más importantes para el desarrollo de Eiffel.

La definición del lenguaje incluye solamente un pequeño número de construcciones que están implicadas, las ideas básicas son claras y los más difíciles aspectos simplemente resultan de llevar las ideas básicas a sus últimas consecuencias.

Hasta hace poco tiempo, una sola compañía (Interactive Software Engineering) fue la proveedora de compiladores de Eiffel, y una sola persona (Bertrand Meyer) estuvo a cargo de la definición del lenguaje. Esta situación ha cambiado radicalmente: nuevas implementaciones están ahora disponibles, las cuales han sido desarrolladas por empresas de la competencia; y el control sobre el futuro del lenguaje está siendo manejado por una organización que representa a los usuarios de Eiffel en todo el mundo, la Nonprofit International Consortium for Eiffel (NICE).

2.2 Conceptos Básicos

2.2.1 Objeto

"Es una abstracción de un conjunto de cosas del mundo de forma que: todas las cosas del conjunto del mundo real - las instancias -tengan las mismas características; y que éstas a su vez estén sujetas a las mismas reglas"[1]

Un objeto representa una entidad o cosa individual identificable, ya sea real o abstracta, la cual tiene una actividad o un rol bien definido en un problema[2]

"Objeto es una entidad que agrupa a estructuras de datos con el conjunto de operaciones que las manipulan, el identificador de un objeto es único y se distingue de todos los demás"[3]

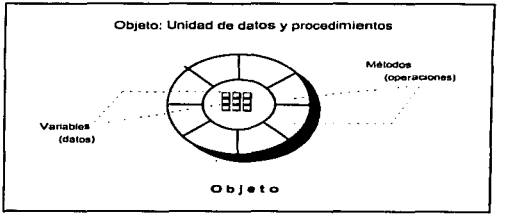
De acuerdo a las definiciones anteriores podemos concluir que un objeto es una representación de una cosa ya sea física o abstracta, y que forma módulos de software ideales debido a que pueden definirse y mantenerse independientemente, formando cada uno un universo autocontenido. Todo lo que un objeto conoce está expresado en sus variables. Todo lo que puede hacer está expresado en sus métodos.

Un objeto por tanto, tiene:

- Identidad
- Atributos
- Relaciones
- Comportamiento (conjunto de métodos)

[1] Seminario Metodología Orientada a Objetos. Diciembre 1994. P. 9
[2] Object Oriented Desing Grady Booch P. 76
[3] Programación Orientada a Objetos ¿Moda o Realidad? Soluciones Avanzadas. Abril Mayo 1993. P. 41

Lourdes Montiel García
Isabel Pérez Arredondo



2.2.2 Identidad

Es un identificador único que distingue a un objeto de todos los demás.

2.2.3 Atributo

Es la abstracción de las características de un objeto, estos atributos deberán hacer al objeto: más completo, exclusivo y mutuamente independiente, es decir, que tomen su valor independientemente uno de otro.

El conjunto de atributos se conoce también como variables de instancia porque definen el estado del objeto en cualquier momento. El valor de las variables puede cambiarse dinámicamente durante la vida de un objeto.

2.2.4 Relación

Es un evento que relaciona a dos o más objetos por medio de atributos.

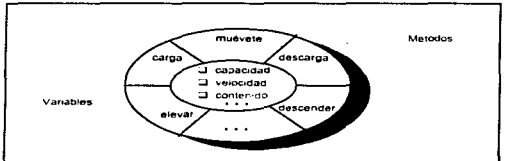
Lourdes Montiel García
Isabel Pérez Arredondo

2.2.5 Métodos

Podemos definir a un método como el conjunto de operaciones que son capaces de realizar los objetos de una clase. Es cómo un objeto actúa o reacciona a los mensajes que otros objetos le envíen.

Al representar a un objeto real tendríamos que describir sus acciones posibles como métodos y sus características como variables. Durante la simulación, el objeto ejecutaría sus diferentes métodos, cambiando sus variables conforme sea necesario para reflejar así los efectos de dichas acciones.

Por ejemplo, para representar un vehículo automatizado en la simulación de una fábrica debemos considerar que el vehículo puede realizar una variedad de acciones, tales como moverse de una posición a otra, elevar su carga y descargar su contenido. También tiene que mantener información sobre sus características inherentes: capacidad de carga, velocidad máxima, etc., así como de su estado actual: contenido, posición, orientación y velocidad.



Un Vehículo Automatizado como un Objeto

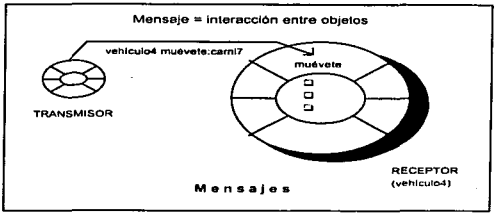
Para representar el vehículo como un objeto tendríamos que describir sus acciones posibles como métodos y sus características como variables.

Lourdes Montiel García
Isabel Pérez Arredondo

2.2.6 Mensaje

La forma en que los objetos interactúan entre sí es enviándose mensajes pidiendo que se ejecute un método específico. Un mensaje consiste simplemente del nombre del objeto a quien va dirigido seguido del nombre del método que el objeto receptor sabe cómo ejecutar. Si el método requiere información adicional, el mensaje incluye esta información como parámetros.

Los objetos del mundo real se pueden afectar en infinita variedad entre sí: crear, agregar, mover, enviar, remover, etc.



El objeto que inicia el mensaje se conoce como transmisor y el que lo recibe como receptor. Un sistema Orientado a Objetos consiste de varios objetos interactuando unos con otros enviándose mensajes entre sí. El objetivo de un mensaje consiste en la activación de alguno de los métodos que tiene el objeto receptor.

Siguiendo con nuestro ejemplo, un objeto X que funje como transmisor, envía al objeto vehiculo4, quien es el receptor, un mensaje en el que le solicita que active el método muévete, como información adicional le envía el parámetro carril7.

Lourdes Montiel García
Isabel Pérez Arredondo

2.2.7 Encapsulación

Es el mecanismo que permite ocultar los detalles de la representación interna de un componente, presentando la interfaz disponible para el usuario.

2.2.8 Clase

"Una clase es una familia de objetos de un tipo en particular. Los objetos que pertenecen a una clase son llamados **instancias**, por ello una clase puede tener una o varias instancias, pero una instancia tiene sólo una clase, ésto se conoce como herencia simple."[4]

Mientras un objeto es algo concreto que existe en tiempo y espacio, una clase representa solamente una abstracción, la esencia de un objeto. Podemos definir una clase como un grupo o un tipo marcado con un comun atributo:

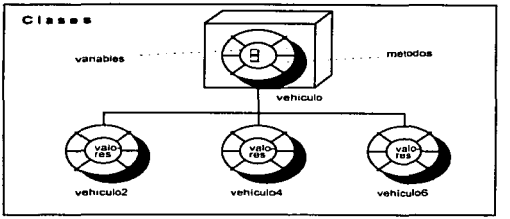
"Una clase es una definición de objetos que comparten una estructura y un comportamiento comunes."[5]

"Una clase es un prototipo que define los métodos y variables que serán incluidas en un tipo de objeto particular, las descripciones de los métodos y variables que los soportan se describen sólo una vez, en la definición de la clase."[6]

De acuerdo con las definiciones anteriores podemos concluir que una clase es aquel conjunto de objetos que tienen caracteristicas y comportamiento semejantes, por lo tanto los objetos que no tengan atributos semejantes no pueden ser agrupados en una misma clase. La agrupación de objetos en clases es una forma de ordenar y organizar el dominio de objetos que conforman un sistema.

[4] Item 1. P. 17
[5] Item 2. P.93
[6] "Nuevas Tecnologías para el Desarrollo de Aplicaciones. Octavio Orozco y Orozco. Soluciones Avanzadas. Noviembre - Diciembre 1993. Pp. 4-9

Las clases surgieron porque en raras ocasiones un sistema involucra un solo objeto de cada tipo. Es mucho más común requerir más de uno.



Los objetos que pertenecen a una clase se denominan instancias de la clase, y contienen tan solo los valores particulares para las variables, compartiendo el código de los métodos. Para continuar con el ejemplo del vehículo automatizado, la colección de vehículos podría representarse por la clase llamada vehículo, y esa clase contendría la definición de sus métodos y variables. Los vehículos en sí estarían representados por instancias de esta clase, cada uno con su nombre único: vehículo2, vehículo4, etc. Cuando uno de estos vehículos reciba un mensaje para ejecutar, iría a la clase por la definición del método y después aplicaría el método en sus propios valores locales.

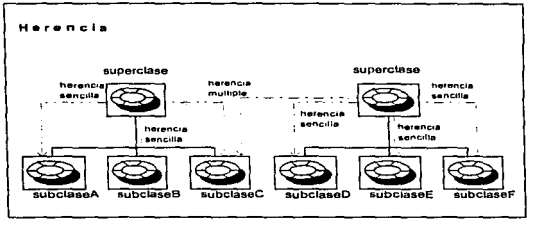
Lourdes Montiel García
Isabel Pérez Arredondo

2.2.9 Herencia

"La herencia es un mecanismo por el cual una clase de objetos puede definirse como un caso especial de una clase más general, con lo cual automáticamente incluye toda la definición de métodos y variables de la clase general."[7]

"Herencia, es una relación entre clases, en donde una clase comparte la estructura o comportamiento definidas en una de ellas o más con otras clases."[8]

A la clase que hereda se le conoce como superclase y a la clase heredera se le denomina subclase. Una subclase puede añadir y/o modificar operaciones a su comportamiento, formando así una especialización de la superclase. Una clase puede ser subclase de una o más superclases. En el primer caso se habla de **herencia sencilla** y en el segundo de **herencia múltiple**.



La herencia entre clases puede extenderse a cualquier grado. El resultado es una estructura arborescente conocida como **Jerarquía de Clases.**

[7] Item 6, P. 7
[8] Item 2, P.514

Lourdes Montiel García
Isabel Pérez Arredondo

Cuando hay herencia de clases, los métodos definidos en una superclase se heredan a los objetos de las subclases.

2.2.10 Polimorfismo

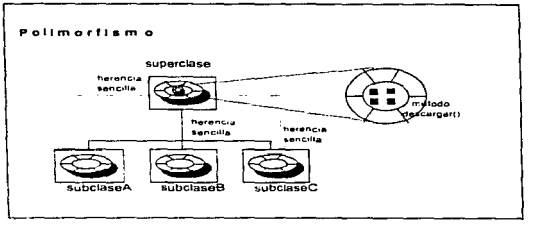
Es la definición de un objeto el cual hereda las características de varias clases, las cuales están relacionadas con una superclase, esto permite al objeto responder a algunos mensajes comunes establecidos que definen un proceso de diferentes formas.[9]

Polimorfismo, es la capacidad de un objeto para responder al mismo mensaje de una o más clases, y cada uno de diferente forma. Esto significa que un objeto no necesita conocer quién envía el mensaje, él sólo necesita conocer cuántos tipos de objetos han sido definidos para responder a un mensaje en particular.[10]

El polimorfismo nos permite reconocer y explotar semejanzas entre diferentes clases de objetos.

Supongamos que en nuestro ejemplo se requiere de un método que descargue el contenido de un vehiculo. El problema de descargar el contenido de los vehiculos 2 y 4 no consiste en lo mismo. Se puede definir un método descargar, para cada subclase de la clase vehículo. La codificación para el manejo de vehículos se simplifica considerablemente. Este comportamiento es una especie de polimorfismo. El procedimiento que está manipulando a los vehículos es de múltiples formas. La misma rutina adopta formas distintas en el sentido de que puede llevar a cabo diferentes actividades dependiendo del subtipo del vehículo involucrado.

[9] Item 2, P 517
[10] Designing Object - Oriented Software. Rebecca Wirfs-Brock, Pag 23



2.3 Caracteristicas de los Sistemas Orientados a Objetos

El desarrollo de sistemas con la metodologia orientada a objetos tiene un doble propósito, por un lado la elaboración de herramientas que solucionen la problemática planteada y por otro, la generación de componentes que puedan ser integrados a la biblioteca de componentes, aumentando asi el capital de software.

El impacto económico de la adopción de la metodología orientada a objetos no puede medirse con base a la construcción de un solo sistema, la ganancia del uso de esta metodologia se muestra a largo plazo, donde la reutilización de los componentes capitalizados se vuelve importante.

La esencia de las metodologias orientadas a objetos son la abstracción de datos, tipos abstractos, reutilización de código, polimorfismo, ocultamiento o encapsulación de información. Entendiendo por :

- Abstracción de datos. La abstracción consiste en capturar una idea general, es decir, la esencia de una entidad más compleja, para posteriormente representarla con algún

elemento interno de un programa. El representar sólo los atributos generales del objeto sin considerar características específicas ni el funcionamiento interno del objeto, permite concentrarse en la manipulación y operaciones a realizar sobre él.

- Tipos Abstractos. La creación de tipos abtractos permiten al programador adaptar el lenguaje de programación a sus necesidades específicas. Un tipo de dato abstracto, es un tipo de dato definido por el usuario y representa la abstracción de un objeto; por tanto se ajusta al problema que se está resolviendo y puede ser manipulado con los tipos internos del lenguaje.
- Reutilización de Código. Consiste en emplear los recursos utilizados en proyectos anteriores. Las metodologías orientadas a objetos proponen dos estrategias para la reutilización de código: la composición y la herencia.

  1) Composición, permite definir una nueva clase de objetos mediante la unión de un conjunto de clases ya existentes.

  2) Herencia, ésta puede ser Simple: crear una nueva clase basada en otra clase más general; o Múltiple: crear una clase de un conjunto de clases, de tal forma que si utiliza algún método y no lo tiene, lo solicita a las clases de las cuales se derivó.
- Polimorfismo. Permite la manipulación de objetos de clases distintas como si fueran de la misma clase, con lo cual es posible definir interfaces uniformes para diferentes tipos de objetos.
- Ocultamiento o Encapsulamiento de Información. Se da a raíz de la abstracción, ya que al extraer la esencia de una entidad, las características internas quedan ocultas. Un objeto debe ser transparente en cuanto a su uso, es decir únicamente debe mostrar sus atributos externos para su manipulación, ocultando su implementación, algoritmos para sus operaciones, estructuras de datos que manejan, etc.

Otra característica distintiva de los sistemas orientados a objetos es que los objetos "saben" qué operaciones se pueden aplicar sobre ellos. Este conocimiento se consigue combinando abstracciones de datos y de procedimientos en un solo componente de programa (denominado objeto).

Los objetos encapsulan datos y procesos; las operaciones de procesamiento son parte del objeto y son iniciados pasando un mensaje al objeto.

Existe una representación especial para los sistemas orientados a objetos, y éstos dependen de la metodología que se haya utilizado para desarrollarlo.

En términos de mantenimiento de Software el manejo de objetos facilita la actualización.

Los tiempos promedios de corrección de fallas de un sistema, al implantarlo, se reduce notablemente, lo que da una flexibilidad para la oportuna corrección del sistema.

El uso de la tecnología de objetos ofrece la facilidad de crear prototipos, los cuales, ayudan de manera importante en el desarrollo de sistemas de información.

El manejo de esta tecnología soporta la adecuación de los sistemas de acuerdo a las necesidades cambiantes del medio organizacional.

" El desarrollo del software mediante estas técnicas, además de ofrecer rapidez, ofrece una mayor confiabilidad"[1]

[1] Una Panorámica de la Ingeniería de Software" Soluciones Avanzadas , No. 8, marzo - abril, 1994, pp 4-6

3. SISTEMAS ESTRUCTURADOS vs SISTEMAS ORIENTADOS A OBJETOS

3.1 Ventajas de los Sistemas Estructurados

- Ofrece a los diseñadores pasos sencillos a seguir.
- Maneja ténicas de diagramación sencillas.
- Las metodologías para sistemas estructurados son aplicables a sistemas pequeños.
- Manejo de diccionario de datos, que contienen detalles tales como atributos e identificadores únicos.
- Existen varias herramientas CASE que manejan metodologías de sistemas estructurados.

3.2 Desventajas de los Sistemas Estructurados

- Algunas metodologías ya han quedado obsoletas, pues se enfocan al desarrollo de lenguajes de tercera generación o bien son para sistemas de base de datos jerárquicas y no relacionales.
- Las metodologías para sistemas estructurados sólo cubren una parte del ciclo de vida de un sistema.
- Algunos métodos emplean para sus modelos diagramas que son muy difíciles de entender, por lo cual se requiere de un entrenamiento especial para aplicarlos.
- En contraparte al punto anterior, algunos métodos emplean diagramas tan simples que por lo mismo resultan incompletos.
- Las actividades de mantenimietno tienden a degradar la estructura de un sistema de software, haciendo que el mantenimiento sea cada vez más costoso.
- La calidad del software resultante de un mantenimiento tiende a degradarse.
- El análisis estructurado se utiliza en aplicaciones de negocios y administrativas, pero no puede utilizarse para multimedia, en un ambiente gráfico o en un sistema distribuido. En los dos últimos casos puede emplearse, pero a costa de grandes esfuerzos, sobre todo en programación.
- El desarrollo de un sistema mediante metodologías estructuradas tiende a crear un conjunto personalizado de subrutinas

Lourdes Montiel García
Isabel Pérez Arredondo

- Los métodos estructurados proponen la reutilización de código, mediante la creación de subrutinas, sin embargo no han tenido éxito debido a que son personalizadas y específicas de un solo sistema.
- El diseño estructurado está basado en la descomposición jerárquica del sistema, alrededor de las operaciones, mientras que los datos se encuentran en un espacio común compartido por las operaciones. Este tipo de organización funciona bien mientras los sistemas tengan un tamaño moderado y sean desarrollados individualmente. Lo cual no es aplicable a los sistemas actuales.
- Rara vez es posible anticipar el diseño completo de un sistema antes de que sea implantado. Entre mayor sea el sistema, es más frecuente que se requiera reestructurarlo.

3.3 Ventajas de los Sistemas Orientados a Objetos

"Para cimentar los resultados tangibles y continuar esta tendencia, tanto los ejecutivos de la empresa, como los administradores de proyectos "deben entender la naturaleza del desarrollo orientado a objetos". Sin este entendimiento, las empresas probablemente adoptarán la tecnología sin realizar la inversión requerida, concentrada notablemente al inicio para desarrollar clases de objetos y modelos reutilizables, en aprender y utilizar los conceptos de modelado, la programación exploratoria y el desarrollo de sistemas de manera evolutiva. Si lo anterior ocurre los decepcionará sin importar que tan buen equipo de desarrollo posean, y pasará más tiempo antes de que se implementen los nuevos sistemas de información que modelan y soportan estrategias de negocio en un mercado global y dinámico en forma competitiva."[1]

- Los métodos Orientados a Objetos (OO) proporcionan una mejor abstracción de los datos y mejor "ocultamiento de la información", permiten la concurrencia más directamente y responden mejor a los cambios del mundo real modelado por el software.
- La metodología OO maneja el concepto de "Sistema en Evolución", que significa que un sistema de información requiere de adecuaciones constantes.

[1] Nuevas Tecnologías para el Desarrollo de Aplicaciones. Octavio Orozco y Orozco
Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios. Noviembre-Diciembre 93 Pag. 9

- Para sistemas que requieran un desarrrollo y evolución acelerados es ideal la tecnología de objetos.
- El desarrollo de sistemas con las metodologías OO tienen un doble propósito:
  La elaboración de herramientas que solucionen la problemática planteada.
  La generación de componentes que puedan ser integrados a la biblioteca de componentes reutilizables.
- La tecnología OO ofrece mecanismos que apoyan la reutilización. En particular, tiene mecanismos para la definición (clases y objetos), especialización (herencia) y construcción de componentes (clases parametrizadas).
- Promueve el cambio de subrutinas personalizadas a una industria de componentes reutilizables.
- La reutilización de componentes puede hacer que el software sea desarrollado más rápidamente, más económicamente y con mayor confiabilidad.
- El código es reducido, porque se utilizan bibliotecas de clases creadas con anterioridad.
- El tiempo promedio de corrección de fallas de sistemas se reduce notablemente, dando una flexibilidad para la oportuna corrección de sistemas.
- La tecnología de objetos ofrece la facilidad de crear prototipos.
- La tecnología OO maneja tipos de datos complejos.
- Las metodologías OO permiten la asociación de una abstracción del mundo real con un componente computacional que refleja directamente las características de la abstracción.
  1. Las metodologías OO generan módulos con un alto grado de cohesión y ligeramente acopladas entre sí.
  2. Aumenta la calidad de software al usar objetos ya probados.
- El mantenimiento es cada vez más específico y enfocado a los objetos de nueva creación y a las relaciones que se establecen con objetos reutilizados.
- Los métodos OO son apropiados para desarrollar en forma escalable sistemas complejos y de gran tamaño.
- La complejidad de un sistema se vuelve más manejable gracias a la posibilidad de jerarquizar las abstracciones de un sistema mediante las herencias entre clases.
- El trabajo de implementación se puede repartir fácilmente entre grupos de programadores, gracias a la poca y bien definida dependencia entre las clases.

Lourdes Montiel García
Isabel Pérez Arredondo

- El proceso de crear clases reutilizables y modelos requiere un examen cuidadoso y detallado de los componentes y operaciones de una empresa, este análisis producirá nuevas perspectivas en el negocio que pueden resultar en nuevas fuentes de ingreso y mejoras significativas en la forma de operar.
- Las metodologías OO facilitan la documentación de un sistema, en términos de la aplicación.

3.4 Desventajas de los Sistemas Orientados a Objetos

- La curva de aprendizaje es mayor a la de las metodologías estructuradas.
- Requiere de un mayor conocimiento de las metodologías OO.
- Requiere de mayor capacitación.
- Requiere que todos los miembros del equipo de trabajo estén compenetrados con la metodología OO seleccionada.
- No existe mucha experiencia en el manejo de las tecnologías OO, lo que provoca una falta de aprovechamiento de las mismas.
- Requiere de un esfuerzo notable de actualización y capacitación por parte del personal de sistemas.
- Se corre el riesgo con el desconocimiento de las metodologías OO de no lograr los objetivos deseados.
- La falta de disciplina puede originar la creación de muchos objetos pequeños, bien estructurados, entendibles aisladamente, pero que sus interacciones son prácticamente imposibles de descifrar.
- Los lenguajes de programación OO están en proceso de estandarizarse, notablemente C++ y Smalltalk. Sin embargo, persisten problemas de portabilidad en la mayoría de los lenguajes e incompatibilidades entre diferentes proveedores.
- Las herramientas han mejorado notablemente, sin embargo, requieren ser perfeccionadas.
- El costo de la conversión asociada con la transicion a esa tecnología es significativo (nuevo software: lenguajes, bases de datos y herramientas que permitan aplicar la tecnología).
- Debido a que la mayoria de los sistemas OO utilizan gráficas de manera intensiva y las empresas que utilizan principalmente terminales de caracteres para acceder

Lourdes Montiel Garcia
Isabel Perez Arredondo

minicomputadoras o mainframes requieren invertir en máquinas tipos estación de trabajo o computadoras personales, creando prácticamente arquitecturas cliente/servidor.

- La educación y capacitación de administradores y técnicos es también costosa, y los programadores seguramente bajarán su productividad durante el aprendizaje de las nuevas técnicas.

3.5 Cuadro Comparativo

Consideración	Sistemas Estructurados	Sistemas Orientadosa Objetos
Metodologías idóneas para sistemas	Pequeños	Muy grandes
Etapa del Ciclo de Vida de un Sistema que demanda mayor cantidad de tiempo	Etapa de Implementación	Etapa de Diseño
Costo de Mantenimiento del sistema	Cada vez mayor	Cada vez menor
Calidad del software resultante de un mantenimiento constante	Tiende a degradarse	Tiende a ser más confiable
Cantidad de esfuerzo en programación en aplicaciones para un ambiente gráfico o un sistema distribuido	Considerable	Moderado
Propuesta de reutilización de código mediante	Conjunto de subrutinas personalizadas	Biblioteca de componentes reutilizables
Manejo del concepto de Sistema en Evolución	No	SI
Adecuación a sistemas que requieran de un desarrollo y evolución acelerados	Pequeña	Grande (idoneo)

Lourdes Montiel García
Isabel Pérez Arredondo

Este diseño es una técnica que basa la descomposición modular de un sistema computacional en	Funciones que el sistema realiza	Clases de los objetos que el sistema manipula
Nivel de abstracción máximo que alcanzan los sistemas de este tipo	Módulos	Jerarquía de Herencia
Cuenta con mecanismos para Definición	Sí	Sí
Cuenta con mecanismos de Herencia	No	Sí
Busca la solución de un problema a través de	Procedimientos Jerárquicos y datos manipulados a través de ellos	Objetos que encapsulan datos y métodos
Estructura de un programa diseñado en este tipo de sistemas	Jerárquica	Plana
Similitud con las estructuras de pensamiento humanas en la forma de atacar los problemas	Mínima	Mucha
Tiempo empleado en el Diseño	Menor	Mayor
Tiempo empleado en desarrollo inicialmente	Regular	Bastante Más
Tiempo empleado en desarrollo subsecuente	Regular	Cada vez menos
Forma en que se activa a un conjunto de operaciones	Llamadas a funciones	Envío de mensajes
Tiempo promedio de corrección de fallas en los sistemas	Mayor	Menor
Cantidad de código que debe reescribirse en cada nuevo desarrollo	Casi todo	Conforme se tengan más componentes reutilizables será menor
Curva de Aprendizaje	Menor	Mayor

Lourdes Montiel García
Isabel Perez Arredondo

Costo de conversión asociado con la transición a esta tecnología	Moderado	Elevado (muy cargado al inicio)
Necesidad de inversión inicial en hardware	No requiere hardware muy especializado	Requiere de hardware más especializado
Costo de la educación y la capacitación de los administradores y técnicos	Moderado	Elevado (implica una nueva forma de modelar que afecta las prácticas de programación)
Productividad de los programadores durante el periodo de aprendizaje de las nuevas técnicas y las nuevas herramientas	Baja pero prontamente aumentará	Baja y tardará un poco más en aumentar
Aprovechamiento de las tecnologías	Mucho, porque hay una gran experiencia en su manejo	Poco, porque hay gran desconocimiento y falta de experiencia en su manejo
Riesgo de no alcanzar los objetivos deseados al iniciar con esta tecnología	Baja	Mucho mayor
Grado de portabilidad en la mayoría de los lenguajes e incompatibilidad entre los diferentes proveedores	Mínimo	Mayor
Manejo de diccionarios de datos para la documentación de los sistemas	Sí	Sí
Utiliza el principio conocido como Encapsulación	No	Sí

Lourdes Montiel García
Isabel Pérez Arredondo

Diseño de aplicaciones pensando en que éstas deben ser fácilmente reusadas, refinadas, probadas, mantenidas y extendidas	No	SI
Capacidad de encapsular piezas de un sistema en unidades que pueden ser implementadas sin considerar las interacciones con el resto del sistema	Dificilmente	SI
Años en que estos paradigmas han impactado la Ingeniería de Sistemas Computacionales	Los años 70's	Los años 90's

2

[2] Artículos fuente para la elaboración de las secciones de este capítulo:
"Programación Orientada a Objetos moda o realidad?", Dra. Hanna Oktaba
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios. Abril-Mayo 1993 Pag. 39 a 42
"El Impacto en la Ingeniería de Software la Programación Orientada a Objetos", M.C. Gloria Quintanilla
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios. Abril-Mayo 1993 Pag. 43 a 46
"La Migración hacia la Tecnología Orientada a Objetos", José Raúl Pérez Cázares
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios. Noviembre-Diciembre 1993
"Paradigma vs Metodología" (Parte II), M.C. Warren R. Greiff
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios. Enero-Febrero 1994
Nuevas Tecnologías para el Desarrollo de Aplicaciones, Octavio Orozco y Orozco
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios. Noviembre-Diciembre 93 Pág. 9

PARTE II

En la segunda parte de la tesis se realiza el estudio de las siguientes tres metodologías orientadas al objeto:

- Metodología Wirfs-Brock
- Metodología de Edward Yourdon
- Metodología de Grady Booch

Estas metodologías han sido seleccionadas por el prestigio que los autores tienen en el ámbito de la Ingeniería de Software.

En la búsqueda de las metodologías que serían objeto de estudio, se consultaron documentos que trataban la Ingeniería de Software, en fuentes tales como: publicaciones técnicas (revistas)[1], páginas de Wide World Web[2] y libros técnicos[3]. En las cuales los más comentados fueron el método de Yourdon, el de Booch y el de Wirfs-Brock.

---

[1] Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios, Personal Computing México, Byte
[2] http://www.toa.com/pub/html/mer.lhtml
http://dmalek.simplenet.com/odd/DOOS.htm/method.htm
http://www.yourdon.com/bio.html
http://www.prenhall.com/divisions/ptc/yourdon/yourdon.html
[3] Ingeniería del Software, Un Enfoque Práctico de Roger S. Pressman, Sistemas de Información Administrativa de Robert G. Murdick con John C. Munson

Lourdes Montiel García
Isabel Pérez Arredondo

Entre los antecedentes que encontramos de estos autores, podemos mencionar los siguientes:

Edward Yourdon, ha trabajado por más de 30 años en la industria de la computación incluyendo altos puestos en General Electric Co. y Digital Eq. Co. En 1974 fundó una empresa llamada Yourdon Inc. para desarrollo, consultoría, educación y publicaciones en la tecnología de software, llegando a publicar más de 150 libros sobre computación y un amplio rango de tópicos referentes a la Ingeniería de Software. Muchos de ellos textos "clásicos" estándar en las universidades en el mundo. También es autor de más de 200 artículos técnicos. Sus más recientes libros y artículos han estado orientados al tema "Análisis y Diseño Orientados al Objeto".

Grady Booch, su trabajo ha dado significativas contribuciones para incrementar la efectividad de los desarrolladores de software a nivel mundial. Ha trabajado en la Rational Software Co. como jefe científico desde poco despues de su fundación en 1980. Su trabajo inicial con ADA y sus facilidades para la abstracción de datos y el ocultamiento de la información lo llevaron a la aplicación de la tecnología de objetos. Desarrolló dos grandes conceptos: modelos interactivos de desarrollo de software y la importancia de la arquitectura de software.

Los métodos e ideas de Grady Booch han sido utilizados para desarrollar algunos de los más complejos y demandantes sistemas de información, que van desde un sistema de control de tráfico aéreo, sistemas financieros, sistemas de comunicaciones a un sistema de defensa.

Es autor de cinco libros inluyendo "Ingeniería de Software con ADA y componentes de software con ADA". A la par de su segundo libro, impulsó la fundación de librerías de clases escritas en ADA y C++. El

tercer libro "Análisis y Diseño Orientado a Objetos" describe la teoría, notación, procesos y práctica de la tecnología orientada a objetos. También ha publicado más de 100 artículos de la tecnología orientada a objetos a la Ingeniería de Software.

Wirfs - Brock, ha contribuido al conocimiento de la tecnología orientada a objetos, como autora y coautora de cinco libros, entre ellos "Object Oriented Design". Actualmente colabora en un panel de discusiones de tecnología titulado "ParcPlace - Digitalk, Inc."[4] . Entre las aportaciones a la orientación a objetos se encuentran su conceptualización de responsabilidades, colaboraciones, protocolos y su respectiva notación.

[4] http://www.bell-labs.com/user/cope/oopsla//StandardizingOOAD

4. METODOLOGIA DE REBECA WIRFS-BROCK

4.1 INTRODUCCION

Esta metodología abarca únicamente la etapa de Diseño del Ciclo de Vida del Software y asume que los requerimientos para un programa en particular ya están dados.

A partir de esta información de entrada se producirá un diseño. La salida -el diseño final- consistirá de:

- un sistema de objetos que satisfaga los requerimientos
- una descripción del comportamiento público de esos objetos, y
- los patrones de comunicación entre los objetos

A través del estudio de la presente metodología se encuentran unos pocos conceptos que es conveniente estudiar antes de proseguir.

4.1.1 Signatura

Es un concepto relacionado con el envío de mensajes. Mientras un mensaje consiste del nombre de un método y sus argumentos, si es que son requeridos, una signatura es el nombre de un método, el tipo de sus parámetros y el tipo del objeto que el método regresa.

4.1.2 Clases Abstractas

En cierta forma, hasta el momento se ha dicho que una clase tiene la particularidad de crear instancias de ella misma, sin embargo, esto no se efectúa necesariamente. Las clases que no están proyectadas para producir instancias de sí mismas, sino exclusivamente para heredar su comportamiento a otras clases son llamadas clases abstractas.

Estas clases existen para que el comportamiento común a una variedad de clases pueda ser generado en un solo lugar, en donde puede ser definido una vez y después reusado.

Las clases abstractas pueden ser diseñadas en dos formas distintas. Pueden proveer implementaciones funcionales completas del comportamiento que ellas contienen o pueden proveer un patrón para el comportamiento que posteriormente se redefinirá por métodos específicos en las subclases.

Las subclases concretas heredan el comportamiento de sus superclases abstractas y añaden otras habilidades únicas para sus propósitos.

4.1.3 Diseño Orientado a Objetos

Es el proceso por el cual los requerimientos de software son transformados en una detallada especificación de objetos. Esta especificación incluye una completa descripción de los roles y responsabilidades de los objetos y cómo éstos se comunican con otros objetos.

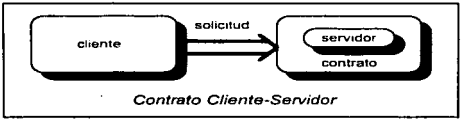
4.1.3.1 Subsistemas de Clases

Dentro de la etapa inicial del diseño orientado a objetos las clases se utilizan como una forma de particionar y estructurar la aplicación. Cuando se comienza a descomponer la aplicación, se deben identificar inmediatamente las clases. Pero también deben encontrarse otras cosas: piezas que tienen cierta integridad lógica, pero que son susceptibles de descomposición en piezas más pequeñas. Estas piezas son referidas como subsistemas. Un subsistema es un conjunto de clases (y posiblemente otros subsistemas) colaborando para satisfacer un conjunto de responsabilidades. Aunque los subsistemas no existen al momento de ejecución, son entidades conceptuales muy útiles.

4.1.3.2 Clientes y Servidores

Las colaboraciones entre los objetos que componen al sistema han sido vistas hasta el momento como interacciones en un solo sentido: un objeto solicita un servicio de otro objeto. El objeto que efectúa la solicitud es el cliente, y el objeto que recibe la solicitud y luego provee el servicio es el servidor.

La forma en que un cliente puede interactuar con un servidor es descrita por un contrato (lista de solicitudes que un cliente puede hacer a un servidor)



Contrato Cliente-Servidor

4.2 FASE EXPLORATORIA

4.2.1 Búsqueda de Clases en el Sistema

El punto de partida para todo el diseño está constituido por la Especificación de Requerimientos, que es la única entrada que se tiene. En caso de no tenerla entonces escribir una descripción de los objetivos que el diseño debe satisfacer, en caso de ya tenerla entender ésta claramente antes de comenzar. El propósito debe ser crear clases de objetos que modelarán el dominio de la aplicación.

Leer la especificación de requerimientos cuidadosamente buscando sustantivos; cambiar todos los plurales por singulares y hacer una lista preliminar. Revisando la lista se puede hacer una primera selección dividiendo los sustantivos en tres categorías: clases

obvias, disparates y frases de las que no se está seguro. Se deben descartar los disparates. Las otras dos categorías pueden constituirse en clases candidatas, aunque no todas ellas estarán en el diseño final.

Para decidir cuáles otros sustantivos son candidatos significativos a clases se debe atender a lo siguiente:

- Modelar objetos físicos, como discos o impresoras.
- Modelar entidades conceptuales que formen una abstracción, como una ventana de desplegado o un archivo.
- Escoger una palabra por cada concepto, de modo que si más de una palabra es usada para el mismo concepto, se escoge la que es más significativa en términos de todo el sistema.
- Ser cuidadoso del uso de adjetivos. Un adjetivo puede sugerir un diferente tipo de objeto, un diferente uso del mismo objeto o puede ser totalmente irrelevante. Si el uso del adjetivo señala que el comportamiento del objeto es diferente, entonces hay que hacer una nueva clase.
- Ser cuidadoso de enunciados en voz pasiva o aquellos cuyos sujetos no son parte del sistema. Algunos enunciados escritos en voz pasiva pueden implicar sujetos, aunque no aparezca ningún sustantivo en la descripción. Similarmente, muchos enunciados pueden estar escritos en voz activa, pero sus sujetos son cosas que están fuera del sistema.
- Modelar categorías de clases como clases específicas e individuales.
- Modelar las interfaces del sistema, tales como la interface del usuario, interfaces con otros programas o interfaces con el sistema operativo, hacer esto tan completamente como el entendimiento inicial del sistema lo permita.

- Modelar los valores de los atributos de los objetos, más no los atributos en sí mismos.

El resultado de este procedimiento es la primera lista tentativa de las clases en el programa.

Otro punto a considerar es que probablemente no todas las partes de la aplicación han sido igualmente detalladas. Se diseñarán en forma más rica y compleja aquella partes del sistema que se entiendan mejor. Si hay alguna parte o subsistema que no se haya entendido completamente, podrá tomarse como una caja negra que posteriormente será descubierta.

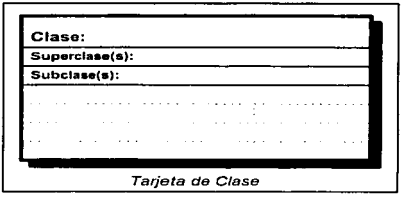
4.2.1.1 Encontrando Clases Abstractas

Reexaminando la lista de clases candidatas se deben identificar tantas clases abstractas como sea posible. Una clase abstracta surge de un conjunto de clases que comparten un atributo. Si el comportamiento es compartido por varias clases, se deberá diseñar una superclase abstracta para capturarlo en un solo lugar.

4.2.1.2 Registro de las clases

Escribir los nombres de las clases en tarjetas, cada clase en una tarjeta y en el reverso escribir una pequeña descripción del propósito general de la clase.

También deben registrarse en una tarjeta, cada una de las superclases que se hayan identificado y escribir sus subclases en las líneas inferiores al nombre. Regresar a todas las tarjetas de clases y registrar sus superclases y subclases, si es que alguna de ellas se conoce, en las líneas inferiores al nombre.



Tarjeta de Clase

4.2.2 Búsqueda de relaciones

Las responsabilidades de un objeto son todos los servicios que provee para todos los contratos que soporta. Por lo tanto las responsabilidades incluyen dos temas clave:

- el conocimiento que un objeto mantiene
- las acciones que un objeto puede realizar

Un contrato entre dos clases representa una lista de servicios que una instancia de una clase puede solicitar a una instancia de otra clase. Un servicio puede ser la realización de alguna acción o el retorno de información. Todos los servicios listados en un contrato particular son las responsabilidades de el servidor para ese contrato.

Las responsabilidades intentan representar solamente los servicios públicos disponibles. Un objeto puede necesitar conocer y hacer otras cosas para satisfacer sus responsabilidades públicas, pero dichas cosas son consideradas privadas para el objeto.

4.2.2.1 Identificación de Responsabilidades

Ellas pueden ser identificadas de varias maneras. En un principio dos recursos son la especificación de requerimientos y las clases que ya han sido identificadas.

En cuanto a la especificación de requerimientos, ésta debe leerse cuidadosamente, buscando verbos y determinar cuáles de ellos representan acciones que algún objeto en el sistema debe realizar. También se debe buscar información en cualquier lugar donde ésta sea mencionada, ya que la información que algún objeto pueda mantener y manipular también representa responsabilidades.

Con respecto a las clases, el hecho de que ya se haya identificado una clase indica que se ha encontrado una necesidad que se debe satisfacer por al menos una responsabilidad.

4.2.2.2 Asignación de responsabilidades

Se debe asignar cada responsabilidad encontrada a la clase o clases con la(s) que lógicamente deben estar. Guías para la asignación de responsabilidades:

- Distribuir la inteligencia del sistema
- Declarar las responsabilidades tan generalmente como sea posible
- Mantener el comportamiento con la información relacionada
- Mantener la información acerca de una cosa en un solo lugar
- Compartir responsabilidades entre objetos relacionados

Distribuir la Inteligencia del Sistema

Un sistema posee una cierta cantidad de inteligencia, la cual es lo que el sistema conoce, las acciones que puede realizar y el impacto que tiene en otros sistemas (incluyendo a los usuarios) con los que interactúa.

Cuando se diseñan las clases, debe decidirse cómo distribuir la inteligencia entre ellas, es decir, cuáles deben incorporar qué aspectos del total de inteligencia del sistema. Es recomendable distribuirla entre las clases, tanto como sea posible, de manera que las clases sean lo más parecidamente inteligentes posible, esto sin olvidar que dependiendo

de los roles que tenga cada clase puede ser que haya mayor o menor inteligencia en alguna de ellas.

Declarar las Responsabilidades tan Generalmente como sea Posible

Si se declaran las responsabilidades en términos generales, pueden encontrarse responsabilidades que se comparten entre clases más fácilmente. Esto es, si en lugar de declarar las responsabilidades de cada uno de los elementos que componen a una clase, se mencionan las de esta última, se puede encontrar que una clase descrita en esta forma puede ser más útil.

Mantener el Comportamiento con la Información Relacionada

Si un objeto es responsable por el mantenimiento de cierta información, es lógico asignarle la responsabilidad de realizar cualquier operación necesaria sobre esa información. Este punto habla sobre la esencia de la encapsulación, la cual asume que cosas parecidas deben estar juntas.

Mantener la Información acerca de una Cosa en un solo Lugar

En general, la responsabilidad del mantenimiento de información especifica, no debe ser compartida. El compartir información implica una duplicación que podria guiar a la inconsistencia. Si dos o más objetos tienen que conocer la misma información para realizar una acción, existen tres posibles soluciones:

- Un nuevo objeto, creado para ser el único depósito de la información
- Reasignación de la responsabilidad del manejo de la información a quien la tiene
- Unir los objetos que requieren la misma información en uno solo

Compartir responsabilidades entre objetos relacionados

Ocasionalmente, se descubrirá que una cierta responsabilidad pareciera ser muchas responsabilidades o una responsabilidad compuesta y que es mejor dividirla o compartirla entre dos o más objetos.

4.2.2.3 Examinando relaciones entre clases

Responsabilidades adicionales pueden ser identificadas examinando las relaciones entre clases. Tres relaciones muy útiles son:

- "Es del tipo de"
- "Es análogo a"
- "Es parte de"

Cuando una clase parece ser *del tipo de* otra, frecuentemente es un signo de una relación subclase-superclase entre ellas. Es una ventaja asignar una responsabilidad a una superclase, porque de esta manera es asignada una vez y heredada por todas las subclases.

Cuando varias clases parecen ser *análogas*, suele ser un signo que comparten una superclase en común, que probablemente aún no ha sido identificada. Se debe asignar una responsabilidad a una superclase abstracta, solamente si todas las subclases de la superclase abstracta comparten la responsabilidad.

Cuando una clase parece *ser parte de* otra clase, no se implica la herencia del comportamiento. El que un objeto esté compuesto de muchas instancias de otras clases no quiere decir que se comporte como esas partes. De hecho, suele comportarse un tanto diferente, esto debido a que deberá interpretar cómo deberá comportarse.

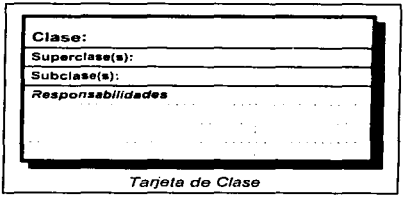
El proceso de la asignación de las responsabilidades no es siempre directo. Las dificultades en ello ocurren más frecuentemente porque:

- una clase está faltando en el diseño, o

- la responsabilidad podría ser asignada, de manera razonable, a más de una de las clases candidatas.

4.2.2.4 Registro de Responsabilidades

En cada tarjeta de clase que se haya creado registrar cada responsabilidad asignada a esa clase. La lista de responsabilidades debe ser lo más sucinta posible (solamente una frase por cada una).



Tarjeta de Clase

4.2.3 Identificación de Colaboraciones entre Clases

Las colaboraciones representan solicitudes de un cliente a un servidor para la satisfacción de una responsabilidad del cliente. Un objeto puede satisfacer una responsabilidad él mismo, o puede requerir la ayuda de otros objetos (para lo cual debe enviarles mensajes). Aunque cada colaboración trabaja para satisfacer una responsabilidad, la satisfacción de una responsabilidad no requiere necesariamente de una colaboración puede requerir de varias colaboraciones o puede no requerir de ellas.

La identificación de colaboraciones entre clases forza a determinar cuáles clases jugarán el rol de clientes y cuáles de servidores para cada contrato.

4.2.3.1 Encontrando Colaboraciones

Para determinar colaboraciones entre clases, deben hacerse las siguientes preguntas para cada una de las responsabilidades de cada clase:

¿La clase es capaz de satisfacer su responsabilidad ella misma?
Si no ¿Qué necesita?
¿De qué otra clase puede obtener lo que necesita?

Cada responsabilidad que se haya decidido compartir entre clases también representa una colaboración entre esas clases. Para cada clase también se debe preguntar:

¿Qué hace o conoce?
¿Qué otras clases necesitan sus resultados o información?

Examinando las relaciones entre clases puede proveer una gran utilidad para identificar colaboraciones. Tres de ellas particularmente útiles son:

- "Es parte de "
- "Tiene conocimiento de"
- "Depende de"

Relación "Es parte de"

Es frecuentemente indicada en la Especificación de Requerimientos por enunciados tales como "X está compuesto de Y", además se encuentran de dos distintos tipos: relaciones entre clases compuestas y los objetos que las componen, y relaciones entre clases contenedoras y sus elementos.

Una clase compuesta es responsable del manejo de sus partes en alguna forma o del mantenimiento de relaciones especificas entre ellos. Estas clases regularmente satisfacen una responsabilidad delegándola a una o más de sus partes.

Una relación entre las clases contenedoras y los elementos que contienen puede o no requerir una colaboración. Un ejemplo claro es un arreglo. Los arreglos contienen sus elementos, pero no necesitan enviarles mensajes. Los elementos de los arreglos son accesados a través de un índice y es un objeto en el sistema el que necesita de dicho acceso. Por lo tanto, los arreglos no tienen la responsabilidad de mantener información acerca de sus elementos (otra que no sea saber cuáles elementos están almacenados en qué índice) y no necesitan colaborar con ellos.

Relación "Tiene conocimiento de"

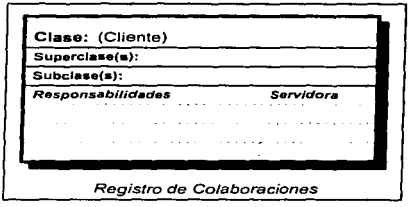
Las clases algunas veces, conocen acerca de otras clases, aunque no estén compuestas de ellas. Relaciones de este tipo pueden ser identificadas en la especificación por frases como "la cual obtiene de". Estas relaciones pueden implicar responsabilidades para conocer sobre cierta información y por lo tanto implica una colaboración entre la clase que tiene el conocimiento y la que la necesita.

Relación "Depende de "

Estas relaciones son algunas veces indicadas en la especificación por frases como "cambia con", lo que indica que una clase depende de información que tenga otra clase para efectuar correctamente su responsabilidad.

4.2.3.2 Registro de Colaboraciones

Tomar la tarjeta de la clase cliente y escribir el nombre de la clase servidora inmediatamente enfrente de la responsabilidad que la colaboración satisface. Si la responsabilidad requiere varias colaboraciones, se debe escribir el nombre de cada clase requerida. Asegurarse que exista una responsabilidad para cada colaboración que se registre.



Registro de Colaboraciones

4.3 FASE ANALITICA

4.3.1 Jerarquías

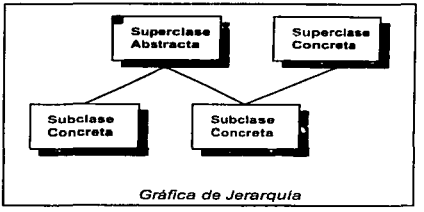
4.3.1.1 Herramientas

Tres herramientas son sumamente útiles para obtener una perspectiva global de las relaciones de herencia en el sistema:

- Gráficas de jerarquía
- Diagramas de Venn
- Contratos

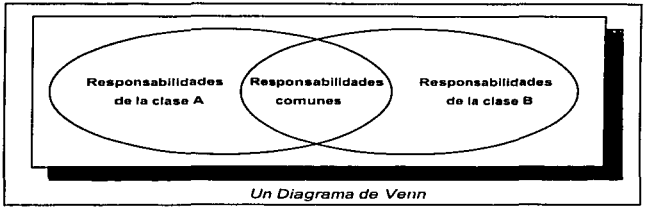
La gráfica de jerarquía da una representación gráfica de las relaciones de herencia entre clases. Las clases se representan por rectángulos, etiquetados con el nombre de la clase. La herencia es indicada por una línea que va de la superclase a la subclase y por la posición en la página, superclases están arriba de sus subclases. Las clases abstractas son indicadas con un sombreado en la esquina superior izquierda.

Lourdes Montiel García
Isabel Pérez Arredondo



Gráfica de Jerarquía

El diagrama de Venn muestra qué responsabilidades son comunes entre clases, indicando dónde superclases abstractas deberían ser creadas.



Un Diagrama de Venn

4.3.1.2 Construcción de Correctas Jerarquías de Clases

Procede analizar las relaciones de herencia entre clases para identificar y resolver problemas en el diseño. Dicho análisis se basa en:

Lourdes Montiel García
Isabel Pérez Arredondo

- Modelar la herencia de acuerdo a "tipo de"
- Agrupar responsabilidades comunes tan alto como sea posible
- Asegurar que las clases abstractas no hereden de clases concretas
- Eliminar clases que no añadan funcionalidad

Modelar la herencia de acuerdo a "tipo de"

Cada clase debe ser un tipo específico de sus superclases. Las subclases deben soportar todas las responsabilidades definidas por sus superclases y probablemente más.

Los Diagramas de Venn pueden ayudar cuando una subclase falla en soportar todas las responsabilidades de sus superclases. Cuando el comportamiento de una subclase incluye solo una parte de las responsabilidades definidas por sus superclases, se debe crear una clase abstracta con todas las responsabilidades faltantes.

Agrupar responsabilidades comunes tan alto como sea posible

Si un grupo de clases soporta una responsabilidad en común, debe heredar dicha responsabilidad de una superclase común, si ésta no existe todavía, crearla y mover la responsabilidad a ella.

Se necesita sólo una responsabilidad para definir una superclase abstracta, pero se necesitan al menos dos subclases especificas de ella antes de que se diseñe una abstracción útil.

Asegurar que las clases abstractas no hereden de clases concretas

Las clases abstractas por su naturaleza nunca deben heredar de clases concretas. Si en el diseño se da tal caso, puede resolverse el problema haciendo otra clase abstracta de la cual ambas, clases abstractas y concretas, puedan heredar el comportamiento común.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Eliminar clases que no añadan funcionalidad

Las clases que no tienen responsabilidades deben ser eliminadas, a menos que la forma en que una clase la implemente sea única, por lo que se considera que añade funcionalidad a pesar de no tener responsabilidades por sí misma.

4.3.1.3 Identificación de Contratos

Un contrato es otra herramienta de diseño, para agrupar las responsabilidades de una clase que están relacionadas de alguna manera. Sirve como un mecanismo adicional de abstracción para la creación de un nivel en el cual el diseño es menos complejo. También ayuda a asignar y reasignar responsabilidades.

La palabra contrato no es un sinónimo de responsabilidad. La diferencia radica en que ésta última es algo que un objeto puede hacer para otros objetos, ya sea realizando alguna acción o respondiendo con alguna información, mientras que un contrato define un conjunto de responsabilidades que un cliente puede esperar obtener.

Una clase puede soportar cualquier número de contratos. Cada responsabilidad será parte de a lo sumo un contrato, pero no todas serán parte de un contrato. Algunas de ellas representan comportamiento que una clase debe tener, pero que no puede ser solicitada por otros objetos y son llamadas responsabilidades privadas.

Para determinar qué responsabilidades deben estar juntas con un contrato se debe:

- Agrupar responsabilidades usadas por los mismos clientes
- Maximizar la cohesión de clases
- Minimizar el número de contratos

Agrupar responsabilidades usadas por los mismos clientes

Frecuentemente una clase con muchas responsabilidades soporta solamente un contrato (las responsabilidades representan una vista más detallada de los servicios

proveídos por la clase). Sin embargo, en otros casos cuando más de un cliente solicita las responsabilidades de una clase es correcto definir los contratos necesarios.

Maximizar la cohesión de clases

Así como un contrato debe estar compuesto por un grupo de responsabilidades, una clase debe soportar un conjunto de contratos.

La maximización de la cohesión tenderá a minimizar el número de contratos soportados por cada clase. A menor número de contratos, mayor facilidad para que nuevas clases sean construidas partiendo de la estructura ya establecida.

El principio de cohesión ayuda a hallar el balance entre unas clases pequeñas, fácilmente entendibles y reusables, con el conflicto de un pequeño número de clases cuyas relaciones con otras clases sean fácilmente captadas.

Si una clase define un contrato que tiene poco en común con el resto de los contratos definidos por ella, debe ser movido a una diferente clase (superclase, subclase u otra clase no relacionada).

Minimizar el número de contratos

Un sistema será más comprensible si hay menos detalles que comprender, por ello, se debe minimizar el número de contratos. Sin violar su cohesión, la mejor forma de reducir su número es buscar responsabilidades similares que puedan ser generalizadas y posteriormente implementadas de diversas formas dependiendo de la clase (polimorfismo), permitiendo esto mover dichos contratos más alto en la jerarquía.

Aplicación de los lineamientos

Para la definición de contratos una técnica simple es comenzar definiendo los de las clases superiores en la jerarquía. Se necesita definir nuevos contratos solamente para subclases que añadan nueva funcionalidad (para ser usada por distintos clientes).

4.3.1.4 Registro

Examinar cada tarjeta de clase. Determinar los servicios ofrecidos por las clases y asignar cada responsabilidad listada a un contrato. Escribir un enunciado que describa cada contrato y asignarle un número único. Numerar las responsabilidades acorde al contrato al que están asignadas. Ahora ya se han identificado los servidores para cada contrato listado (son las clases en cuyas tarjetas aparecen los contratos). Revisar nuevamente las tarjetas de clases y para cada colaboración, determinar qué contrato representa esa colaboración.

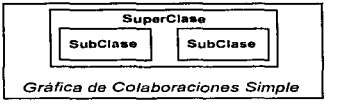
4.3.2 Subsistemas

4.3.2.1 Gráficas de Colaboraciones

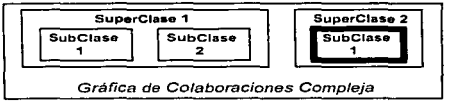
Despliegan las colaboraciones entre clases y subsistemas en forma gráfica, ayudando a identificar áreas de innecesaria complejidad, duplicidad o lugares donde es violada la encapsulación.

Representan clases, contratos y colaboraciones; además de relaciones superclase-subclase, donde una superclase representa los contratos soportados por todas sus subclases. Las clases son mostradas como rectángulos etiquetados:

Las subclases son gráficamente anidadas dentro de los límites de sus superclases:



Una subclase que tiene más de una superclase es dibujada varias veces, dentro de cada superclase. Se muestra la jerarquía en una superclase y se marca la clase anidada en todas las otras superclases para mostrar que además se encuentra en otro lugar.



Los contratos son mostrados como pequeños semicírculos dentro de los bordes de la clase con la que deben estar.

4.3.2.1.1 Subsistemas

Son grupos de clases o grupos de clases y otros subsistemas que colaboran para soportar un conjunto de contratos.

No hay que confundir subsistemas con superclases. Superclases son identificadas encontrando una categoría de clases que comparten responsabilidades idénticas. Subsistemas son identificados encontrando un grupo de clases, cada uno de las cuales satisface diferentes responsabilidades, de forma que cada una colabora estrechamente con otras clases en el grupo para satisfacer una responsabilidad mayor.

Una clase es parte de un subsistema sólo si existe exclusivamente para satisfacer los objetivos de ese subsistema.

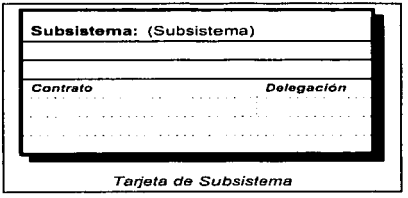
4.3.2.1.2 Contratos de Subsistemas

Los subsistemas como las clases soportan contratos. Para determinar los contratos soportados por un subsistema, encontrar todas las clases que proveen servicios a clientes fuera del subsistema. Estos (al menos inicialmente) son los contratos soportados por el subsistema.

Los subsistemas son entidades conceptuales usados para simplificar el diseño, pero no existen durante la ejecución. Por lo tanto, no pueden satisfacer directamente ninguno de sus contratos. En su lugar, delegan cada contrato a una clase dentro de ellos (la que actualmente soporta el contrato).

4.3.2.1.3 Tarjetas de Subsistemas

Escribir el nombre de un subsistema en una tarjeta. Añadir una pequeña descripción en la parte posterior. Registrar cada contrato requerido por sus clientes, al lado de ellos registrar la clase interna a la que delegan el contrato.

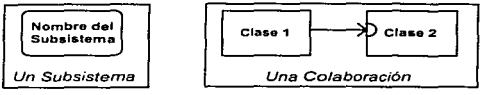


Tarjeta de Subsistema

Paralelamente modificar las colaboraciones en las tarjetas de clases. Si una clase fuera de un subsistema colabora con una clase dentro de él, cambiar la colaboración hacia el subsistema.

4.3.2.1.4 Representación de los Subsistemas en las Gráficas de Colaboraciones

Son mostrados dibujando un rectángulo con las esquinas redondeadas conteniendo las clases y subsistemas que incluyen. Una flecha debe ir desde el contrato del subsistema a la clase que actualmente soporta el contrato.



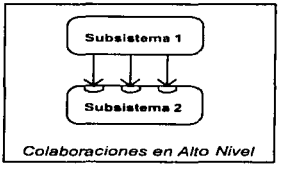
4.3.2.2 Identificación de Subsistemas

Se puede comenzar a encontrar subsistemas dibujando las gráficas de colaboraciones. Posteriormente, buscar clases fuertemente acopladas. El acoplamiento entre dos clases es una medida de cuánto depende una de la otra.

Una forma de determinar si un grupo forma un subsistema es tratar de nombrarlo. Si puede nombrarse se habrá encontrado el propósito para el cual funcionan.

Se puede aplicar el principio de ocultamiento de información a las gráficas que muestran subsistemas, omitiendo del dibujo las clases y subsistemas dentro de un subsistema, permitiendo una vista de alto nivel de las colaboraciones del diseño.

Lourdes Montiel García
Isabel Pérez Arredondo



4.3.2.3 Simplificación de Colaboraciones entre y en subsistemas

La identificación de subsistemas se da para simplificar los patrones de colaboración. Sin dicha simplificación, los caminos de comunicación podrían fluir casi de cualquier clase a cualquier otra.

Guías básicas para la Simplificación de Patrones de Colaboración:

- Minimizar el número de colaboraciones que una clase tiene con otras clases o subsistemas
- Minimizar el número de clases y subsistemas a los que les delega un subsistema
- Minimizar el número de diferentes contratos soportados por una clase o un subsistema

Minimizar el número de colaboraciones que una clase tiene con otras clases o subsistemas

Las clases deben colaborar con tan pocas otras clases como sea posible. Menos colaboraciones quieren decir que la clase es menos propensa a ser afectada por cambios a otras partes del sistema.

Una forma de lograr esto es centralizar las comunicaciones que fluyen en un subsistema. Se puede crear una nueva clase o subsistema para ser el principal intermediario de las comunicaciones o se puede usar una/uno ya existente.

Minimizar el número de clases y subsistemas a los que les delega un subsistema

Un subsistema bien diseñado tiene pocas clases o subsistemas que directamente soportan sus contratos y un mayor número de colaboraciones entre las clases y subsistemas internos.

Este principio corresponde a la guía de centralizar comunicaciones; si una clase o subsistema sirve como intermediario de comunicaciones para un subsistema, entonces los contratos son delegados principalmente o exclusivamente al intermediario.

Después de cambiar una gráfica, se deben actualizar apropiadamente las tarjetas de clases y subsistemas.

Minimizar el número de diferentes contratos soportados por una clase o un subsistema

Demasiados contratos para un subsistema puede ser signo de que demasiada inteligencia de la aplicación está concentrada en ese subsistema. Si esto sucede, quizás las clases pudieran ser divididas en varios subsistemas, cada uno soportando uno o dos contratos. Una vez que estos subsistemas han sido identificados, quizá las comunicaciones entre ellos puedan ser simplificadas.

Una vez que se han dividido los contratos de una clase compleja entre varias clases simples, se deben examinar los nuevos patrones de comunicación.

4.3.2.4 Registro

El propósito de analizar colaboraciones es simplificar un diseño y reducir el número de clases que son dependientes de cualquier otra. Si interacciones complejas requieren

ser rehechas, cambios al diseño original pueden hacerse extensivos. Hay que redibujar las gráficas de colaboraciones para reflejar el nuevo diseño y asegurarse que las tarjetas de clases y subsistemas reflejan las nuevas colaboraciones y responsabilidades.

4.3.3 Protocolos

Un protocolo es un conjunto de signaturas a las cuales una clase responderá. Deben preservar el uso general que se ha pensado para cada clase que se ha diseñado, para ello se presentan las siguientes guías:

- Construir protocolos para cada clase (las signaturas específicas para los métodos que cada clase implementará)
- Escribir la especificación de diseño para cada clase y subsistema
- Escribir la especificación de diseño para cada contrato

4.3.3.1 Refinamiento de Responsabilidades

Durante el proceso de transformar contratos en protocolos debe preservarse la encapsulación y utilidad general de las clases que se han diseñado. Debe obtener un diseño que sea fácil de entender y refinado. Una forma de reducir la complejidad del diseño es seleccionar los nombres de los métodos cuidadosamente. En un conjunto de clases bien diseñado, dos clases que pertenecen a una superclase, responderán al mismo nombre cuando los clientes envíen un mensaje por la misma razón. En otras palabras, cada nombre de método tiene un único objetivo.

- Usar un nombre para cada operación conceptual, donde sea que se encuentre en el sistema
- Asociar una operación conceptual con cada nombre de método
- Si varias clases satisfacen la misma responsabilidad específica, hacerlo explícito en la jerarquía de herencia

Todas las clases que satisfacen una responsabilidad dada deben responder al mismo mensaje (o conjunto de mensajes). Si dos clases comparten una responsabilidad común, idealmente deberían heredarla de una superclase común.

Hacer protocolos de uso tan general como sea posible. Entre más general es una responsabilidad, mayor número de mensajes se pueden definir para especificarla.

Definir valores por default razonables. Los objetos serán más reusables si sus protocolos han sido diseñados para anticipar tantos diferentes usos como sea posible.

- Primero, definir el mensaje más general, uno que permita a los clientes sustituir todos los posibles parámetros requeridos
- Después, proveer valores por default para cualquier parámetro
- Finalmente, analizar cómo cada cliente usa ese mensaje general. Definir un conjunto de mensajes que permitan a los clientes especificar sólo algunos de los parámetros

Listar las responsabilidades de cada clase o subsistema y transformar cada responsabilidad en un conjunto de signaturas. Cada responsabilidad tendrá uno o más mensajes asociados. Nombrar estos mensajes, conjuntamente especificar los tipos de todos los argumentos requeridos y el tipo de objeto retornado por el método, si lo hay.

4.3.3.2 Especificación del Diseño

Retornar a las gráficas de jerarquía, gráficas de colaboraciones y las tarjetas de clases y subsistemas y modificarlas para reflejar cualquier cambio que se haya hecho como resultado del análisis. Redibujar las gráficas, una página para cada una y numerar estas últimas para que puedan ser referenciadas. Ordenar las gráficas de colaboraciones desde la más global hasta la más específica.

4.3.3.2.1 Especificación de Clases

1. Escribir el nombre de la clase en la parte superior de la hoja y definir si es abstracta o concreta

2. Listar sus superclases y subclases inmediatas
3. Incluir referencias a la posición de la clase en la jerarquía y en las gráficas de colaboraciones
4. Describir el propósito de la clase, dicha especificación debe ser más detallada que la que se encuentre en el reverso de la tarjeta de clase
5. Listar cada contrato para el cual la clase es un servidor. Listar los contratos heredados y la clase de la cual son heredados
6. Para cada contrato, listar las responsabilidades de la clase que lo soporta. Debajo de cada responsabilidad, escribir las signaturas de los métodos que implementan la responsabilidad. Si el método requiere servicios de una clase que define varios contratos, indicar el número del contrato apropiado.
7. Listas las responsabilidades privadas que han sido definidas. Para cada una, registrar la misma información que para las responsabilidades soportadas por contratos.
8. Hacer anotaciones que puedan servir para la posterior implementación. No descuidar las condiciones de error; especificar el comportamiento del método para todas las entradas dadas, incluyendo las anormales.

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: nombre de la clase (Concreta o Abstracta)
Superclases: nombres de clases
Subclases: nombres de clases
Gráfica de Jerarquía: página #
Gráfica de Colaboraciones: página #
Descripción: descripción de la clase
Contratos
#. nombre del contrato
responsabilidad
signatura
usa lista de colaboraciones
descripción de la signatura

- página # -

Especificación de una Clase

4.3.3.2.2 Especificación de Subsistemas

1. Escribir el nombre del subsistema en la parte superior de la página. Listar todas las clases y subsistemas encapsulados.

2. Incluir referencias de la posición del subsistema en la gráfica de colaboraciones.
3. Describir el propósito del subsistema.

4. Listar los contratos para los cuales el subsistema es servidor.

5. Para cada contrato. el subsistema o la clase a la cual el contrato es delegado.

Lourdes Montiel García
Isabel Perez Arredondo

Subsistema: nombre del subsistema
Clases: lista de las clases y subsistemas
Gráfica(s) de Colaboraciones: página #
Descripción: descripción del subsistema
Contratos
#. nombre del contrato
Servidor: nombre de la clase o subsistema

- página # -

Especificación de un Subsistema

4.3.3.2.3 Formalización de Contratos

Para cada contrato, especificar el nombre y número, el servidor(servidores), el cliente y una descripción del contrato.

Contrato #: nombre del contrato
Servidor: nombre de la clase
Clientes: nombres de las clases o subsistemas
Descripción: descripción del contrato

- página # -

Especificación de un Contrato

4.3.3.3 Resultados

- Una o más gráficas de jerarquía mostrando los patrones de herencia

Lourdes Montiel García
Isabel Perez Arredondo

- Una o más gráficas de colaboración mostrando todos los caminos posibles de comunicación
- Un conjunto de especificaciones formales de contratos mostrando las clases servidor y cliente cubiertas por el contrato, y
- Una especificación de cada clase y subsistema

5. METODOLOGIA DE EDWARD YOURDON

5.1 INTRODUCCION

Esta metodología consta de las etapas de Análisis y Diseño Orientados a Objetos.

El término Objeto proviene de dos diferentes áreas de estudio:

1. Del Modelado de Información; una representación de algo del mundo real y cierto número de instancias de él.
2. De los Lenguajes de Programación Orientados a Objetos; una instancia al momento de ejecución de algún proceso y sus valores, definido por una descripción estática llamada "clase".

El Análisis Orientado a Objetos (AOO) se construye sobre los mejores conceptos del Modelado de Información y de los Lenguajes de Programación Orientados a Objetos (LPOO).

Del modelado de información provienen: Atributos, Relaciones, Estructura y la representación de un objeto como un número de instancias de algo en el dominio del problema. De los LPOO provienen la encapsulación de Atributos y Servicios exclusivos, el tratamiento de Atributos y Servicios como un todo, la Estructura de Clasificación y la expresión explícita de la semejanza vía Herencia.

El AOO identifica y define Clases y Objetos que directamente reflejan el dominio del problema y las responsabilidades del sistema dentro de él. Sus características principales son:

1. Define y comunica los requerimientos dentro del sistema de organización humana (objeto y atributos, estructura de clasificación, y estructura ensambladora).
2. Se enfoca principalmente en el entendimiento del dominio del problema.
3. Trata a los atributos y servicios exclusivos en esos atributos como un todo.
4. Analiza y especifica utilizando mínima dependencia entre un objeto y otros.

5. Obtiene mayor poder a través de la expresión explícita de las semejanzas.
6. Aplica una consistente y poderosa representación para Análisis (lo que es construido) y Diseño (cómo es construido).

El AOO consiste de cinco pasos principales:

- Identificación de Objetos
- Identificación de Estructuras
- Definición de la Materia
- Definición de Atributos (y Conexiones de Instancias)
- Definición de Servicios (y Conexiones por mensajes)

Una vez que el modelo es construido, es presentado y revisado en cinco capas mayores:

- Materia. Este es un mecanismo para controlar qué cantidad del modelo considera el lector en un mismo tiempo.
- Objeto. Un objeto es una abstracción de datos y procesos sobre esos datos, reflejando las capacidades de un sistema para mantener información acerca de o interactuar con algo en el mundo real.
- Estructura. La estructura representa complejidad en el espacio del problema. La estructura de clasificación describe una organización que refleja generalización-especialización. La estructura ensambladora muestra el todo y sus partes.
- Atributo. Un atributo es un elemento de dato usado para describir una instancia de un objeto o estructura de clasificación. Las conexiones de instancia representan la relación de una instancia de un objeto con otra(s) instancia(s).
- Servicio. Un servicio es un proceso a efectuarse tras la recepción de un mensaje. Las conexiones por mensaje representan el envío de un mensaje de un objeto a otro para obtener algún proceso en su nombre.

5.2 IDENTIFICACION DE OBJETOS

La definición de objetos como una abstracción del mundo real ayuda a obtener entendimiento sobre el dominio del problema, el cual se documenta en el Modelo del Sistema. Dicho modelo provee las bases para una expresión inicial del contexto del sistema, el cual es una indicación de cuánto del problema será abarcado por él, los datos que serán preservados a través del tiempo y qué tan sofisticado será el sistema.

Los objetos representan la expresión inicial del contexto. Los subsecuentes pasos en el AOO proveen un mayor detalle en la descripción del contexto en términos de estructuras, conexiones, atributos y servicios.

5.2.1 Lugares dónde buscar posibles objetos

En primera instancia, observar el espacio del problema, lo cual envuelve cierta investigación que puede iniciar con una revisión de la bibliografía profesionalmente escrita que ayude a aprender la terminología y los fundamentos de un tópico para obtener una perspectiva general del espacio del problema a manejar.

Es también de gran ayuda invertir un poco de tiempo escuchando a la gente para quien se desarrollará el sistema, ya que un consumidor de servicios y sus necesidades son dignos de consideración.

Con respecto a la lectura que se haga deben considerarse los sustantivos en el material escrito; estas palabras suelen dar pistas acerca de Objetos Potenciales en el sistema. Esto no implica que cada sustantivo será un objeto, en la realidad para definirlos se requieren de más guías.

Por lo que se refiere a las imágenes, colectar todas las posibles, como diversos diagramas. Adicionalmente, dibujar algunas imágenes propias que muestren cómo interactúan las piezas entre sí.

Lourdes Montiel García
Isabel Pérez Arredondo

5.2.2 Lo que se debe buscar para localizar objetos

Para encontrar objetos potenciales buscar: estructuras, otros sistemas, dispositivos, eventos recordados, roles, locaciones y unidades organizacionales.

Estructuras. La estructura es tan significativa para encontrar objetos y para la representación de la jerarquía del problema que tiene su propio paso en la metodología de AOO. Las estructuras de clasificación y de ensamblaje son las más fructíferas.

Otros Sistemas. La interacción que tendrá el sistema con otros sistemas, la cual puede ser física, de transmisión o de instancias.

Dispositivos. Algunos dispositivos pueden intercambiar datos e información de control con el sistema. No añadir en esta etapa componentes específicos de la implantación.

Eventos Recordados. Esto se refiere a si hay un evento histórico que debe ser observado y recordado por el sistema.

Roles. Preguntarse qué roles juegan los seres humanos con el sistema estudiado. Objetos que representan gente surgen de dos formas: aquéllos representando a un usuario del sistema, y aquéllos representando gente que no interactúa directamente con él, pero acerca de quienes el sistema guarda alguna información.

Locaciones. Se refiere a lugares físicos, oficinas o instalaciones sobre los que el sistema necesita tener conocimiento. Un objeto de este tipo puede mantener registro de los atributos de uno o varios lugares, dependiendo del problema.

Unidades Organizacionales. Se deben considerar las unidades organizacionales con los que los humanos deben estar juntos, ya que si el sistema requiere conservar información acerca de ellas se convierten en objetos potenciales.

5.2.3 Lo Que se Debe Considerar

Una vez que se ha encontrado un objeto candidato, hay que considerar ciertos puntos para decidir si debe ser incluido en el modelo. Considerar:

Recuerdo Necesario. Si el sistema necesita recordar algo acerca de una ocurrencia de dicho objeto y se localizan algunos atributos potenciales de él, entonces puede ser válido y relevante incluirlo en el sistema.

Servicios Necesarios. En cuanto el recuerdo de algo se aplica, los Servicios serán necesarios, como mínimo para mantener las ocurrencias del objeto, esto es que el sistema necesita proveer un proceso en nombre del objeto. Sin embargo, un objeto podría requerir servicios pero no recuerdo (atributos).

Más de un Atributo. Este criterio ayuda a filtrar objetos potenciales cuando un analista tiene un nivel demasiado bajo en su pensamiento. Si un objeto tiene un solo atributo, se debe dudar de él, puede ser mejor incluirlo como un atributo dentro de otro(s) objeto(s). La clave es el nivel de detalle: los objetos son mejor descritos por atributos, los que después se describen en un Diccionario de Atributos.

Atributos Comunes. Cada vez que surge una ocurrencia de un objeto, el sistema debe tener un valor para cada atributo. Sin embargo, si a veces sólo ciertos atributos se aplican, es una indicación de que existe una estructura de clasificación.

Servicios Comunes. Un proceso puede ser requerido para crear una ocurrencia de un objeto, establece conexiones entre objetos y provee monitoreo en marcha de las ocurrencias de un objeto. Si los servicios son los mismos para cada ocurrencia, está bien; si los servicios varían dependiendo del tipo de ocurrencia, dicha variación indica la existencia de una estructura de clasificación.

Se deben considerar los atributos y servicios comunes en un sentido general, durante la investigación del problema para un conjunto inicial de objetos.

Requisitos Esenciales. Son aquellos que el sistema debe tener, sin tomar en cuenta la tecnología específica usada para construir el sistema (diseño e implantación). Enfocarse en la determinación de la información que es requerida por el usuario del sistema. Usar prototipos y estudios de conceptos operacionales para definir y refinar la información requerida. Pero secuencias particulares de pantallas, menús, ventanas, deben estar en una carpeta de diseño y finalmente en el diseño; ellos son mecanismos para la implantación de los requerimientos, así como para lo que el sistema debe producir para sus usuarios. Mantener un archivo de notas de diseño ayuda a evitar decisiones prematuras de diseño.

También tomar en consideración que:

Si un sistema no necesita mantener información acerca de algo en el mundo real a través del tiempo o proveer servicios para él, se debe remover el objeto. Si un objeto con una sola ocurrencia realmente refleja el problema, permitir su existencia. Pero si otro objeto con los mismos atributos y servicios existe y exactamente refleja la misma realidad, considerar el fusionarlos. Si otro objeto con similares atributos y servicios existe y describe la realidad, entonces usar una estructura de clasificación. Finalmente, examinar el modelo para resultados derivados (por ejemplo, los reportes), no hacer de cada resultado derivado un objeto, en su lugar capturar los datos en objetos de los cuales cada resultado derivado pueda ser obtenido.

5.2.4 Cómo Nombrar Objetos

Usar un sustantivo singular o adjetivo+sustantivo. Un nombre de objeto debe describir una ocurrencia del objeto. Escoger los nombres usando vocabulario estándar de la materia.

Lourdes Montiel García
Isabel Pérez Arredondo

5.2.5 Notación

Representación Gráfica de un Objeto

5.3 IDENTIFICACION DE ESTRUCTURA

La estructura de clasificación captura una organización clase-miembro. La estructura ensambladora describe una organización todo-parte. Ambos tipos de estructuras son componentes importantes de la metodología de AOO.

Una Clase es una descripción de uno o más Objetos, que contienen un conjunto uniforme de características similares (Atributos y Servicios). Clase&Objeto es un término que significa "una Clase y los Objetos en esa Clase" y normalmente cuando nos referimos a un objeto estaremos retomando esta idea.

El símbolo de Objeto (Clase&Objeto) representa a ambos, a la clase (representada por el rectángulo remarcado dividido en tres secciones horizontales) y su(s) Objeto(s) (representado por un rectángulo suavemente marcado). Una variación del símbolo Clase&Objeto es el de Clase (algunas veces éstas son referidas como clases abstractas).

La estructura de clasificación ayuda a representar una jerarquía clase-miembro del problema. Muestra generalización y especialización de cosas del mundo real, con características comunes y extensión de ellas en casos especializados. Provee una importante partición del espacio del problema. Una partición consiste en la división de atributos y servicios en grupos mutuamente exclusivos. Otra partición consiste en el uso de estructuras para identificar un más alto nivel de abstracción que objetos o estructuras (capa de materia).

Lourdes Montiel García
Isabel Pérez Arredondo

También provee un nivel de información acerca del dominio del problema, colocando servicios y atributos que son comunes al más alto nivel y luego extendiendo los servicios y atributos a más bajo nivel.

El concepto de herencia es una parte integral de la estructura de clasificación. La herencia provee un método explícito para la identificación y representación de atributos y servicios comunes. En una estructura de clasificación, la herencia hace posible compartir atributos y servicios, añadir atributos y añadir o extender servicios. Un objeto comparte los atributos definidos arriba en una estructura de clasificación. De igual manera comparte los servicios.

Un objeto representado por una estructura de clasificación puede agregar a sus atributos heredados, extendiendo la generalización arriba de él en la estructura y luego agregar o extender sus servicios heredados de la misma manera.

5.3.1 Cómo Definir la Estructura

Con respecto a la estructura de clasificación, primero considerar un objeto como una generalización y ver diferentes posibilidades de especialización en el espacio del problema. Revisar si:

- Las especializaciones del objeto pueden ser descritas con diferentes atributos o servicios.
- La especialización refleja una significativa especialización del mundo real.
- La especialización está dentro del espacio del problema.

Después considerar un objeto como una especialización. Revisar si:

- Otros objetos del espacio del problema caben en una generalización, expresando atributos o servicios comunes.
- La generalización refleja una generalización del mundo real.
- La generalización por sí misma está en el espacio del problema.

Agregar la estructura de clasificación, dibujando de lo general a lo específico, de arriba a abajo.

Colocar atributos y servicios en la estructura de clasificación, usando herencia para expresar sus semejanzas; colocar los atributos y servicios comunes en el padre y poner los especializados en los hijos.

Con respecto a la estructura ensambladora primero considerar cada objeto como un ensamblaje. Revisar:

- Cuáles son sus partes o componentes.
- El sistema necesita mantener rastro de cada ocurrencia de una parte.
- Cada ocurrencia de una parte puede ser descrita con atributos.
- Las partes reflejan partes del mundo real.
- Está cada parte dentro del ámbito del sistema en estudio.

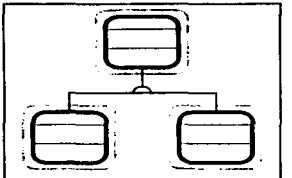
Después considerar cada objeto como una parte. Revisar:

- Dentro de qué ensamblajes cabe cada objeto.
- Qué otros objetos combinan con éste para formar un ensamblaje.
- El sistema necesita mantener rastro de cada ocurrencia del ensamblaje.
- El ensamblaje refleja uno del mundo real.
- El ensamblaje está dentro del espacio del problema.

Agregar la estructura ensambladora, dibujando del todo a las partes, de arriba hacia abajo. Usar una barra simple para mostrar una sola parte; usar patas de gallo para mostrar partes múltiples. Poner una barra simple dentro del ensamblaje cuando una parte pueda ocurrir solamente en un ensamblaje (caso normal); usar una pata de gallo en otro caso.

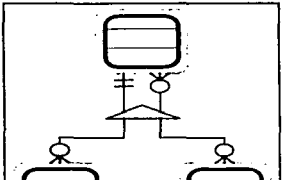
Lourdes Montiel García
Isabel Pérez Arredondo

5.3.2 Notación



Estructura de Clasificación

Las terminaciones de las líneas de una estructura Gen-Espec son colocadas para reflejar una relación entre clases más que entre objetos. Mientras que las líneas de una estructura Todo-Parte son colocadas para reflejar una relación entre objetos más que entre clases.



Estructura Ensambladora

Lourdes Montiel García
Isabel Pérez Arredondo

5.4 IDENTIFICACION DE LA MATERIA

Las materias proveen mecanismos para controlar cuánto de un modelo un lector es capaz de considerar y comprender a un tiempo. También dan una perspectiva general de los diagramas en el modelo AOO.

Los fundamentos para la identificación de la materia es la estructura del espacio del problema definida con las estructuras de clasificación y ensambladoras.

5.4.1 Cómo Definir las Materias

Para seleccionar materias:

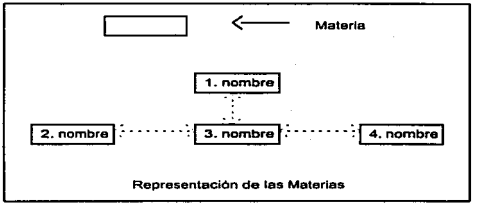
- Agregar una materia correspondiente a cada estructura
- Agregar una materia correspondiente a cada objeto

Para construir la capa de materia:

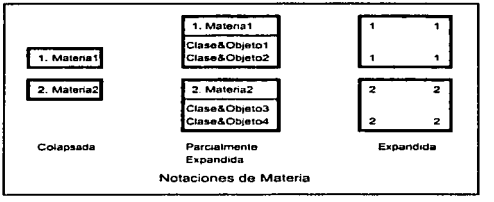
- Mostrar las materias y los mensajes de conexión entre materias en esta capa.
- Numerar las materias. Mostrar las materias en los diagramas de capa para guiar al lector de materia en materia. Para facilitar la comunicación, si se requiere, cada capa debe ser organizada en diagramas separados por materia.
- Dibujar las materias como un simple rectángulo con nombres apropiados. No mostrar los atributos (o estructuras subordinadas) en el diagrama de capa de materia.

El cuándo las materias deben ser introducidas en el modelo, depende de la complejidad del modelo en sí mismo. En proyectos muy pequeños, una capa de materia puede no ser necesaria del todo (las otras capas son suficientemente simples). Para proyectos con varias docenas de objetos, los objetos y estructuras pueden ser identificados primero y después las materias como una forma de guiar al lector a través del modelo.

5.4.2 Notación



Las materias pueden ser mostradas en forma colapsada, parcialmente expandida o totalmente expandida.



5.5 DEFINICION DE ATRIBUTOS

El modelo del sistema está obteniendo ahora mayor especificación y detalle. Los objetos son descritos por atributos, los cuales son mejor descritos en el Depósito de

Lourdes Montiel García
Isabel Pérez Arredondo

Objetos. Con los atributos, nos hacemos más específicos respecto a las abstracciones del problema (objetos y estructuras) en el modelo AOO.

Los atributos aclaran lo que es querido decir por el nombre del objeto, agregando mayor detalle. También describen información que será ocultada en un objeto, para ser manipulado exclusivamente por sus servicios. Si otra parte del sistema necesita obtener información de un objeto, lo hará especificando un mensaje correspondiente a un servicio definido en el objeto.

5.5.1 Cómo Definir Atributos

Definir atributos aplicando los siguientes pasos:

- Identificar los atributos
- Colocar los atributos
- Identificar y definir conexiones de instancia
- Especificar los atributos y restricciones de conexión de instancias

5.5.1.1. Identificar los Atributos

Para comenzar, el analista debe retornar a la materia y preguntar qué atributos se aplican para cada instancia de un objeto o estructura de clasificación. Este paso requiere retornar a las descripciones del problema e interactuar con el usuario.

Ligar cada atributo al objeto o estructura de clasificación del problema que corresponda a la cosa del mundo real que el atributo describe más cercanamente. En la mayoría de los casos esta correspondencia es directa.

Identificar cada atributo a un nivel de concepto atómico. Este concepto puede ser un elemento de dato individual, un grupo natural de elementos de datos o direcciones. La motivación para la expresión del concepto atómico es producir un más simple modelo para revisión, con menos nombres de atributos y grupos naturales de datos para una más fácil asimilación.

5.5.1.2 Colocar los Atributos

Colocar los atributos usando la herencia en las estructuras de clasificación. Si una generalización/especialización existe, colocar los atributos comunes más alto en la estructura y mostrar la especialización abajo. Si un atributo se aplica a la mayoría de las especializaciones, colocarlo en la generalización, después anularlo por las especializaciones que no lo necesiten.

La notación "x" al lado de un atributo colocado en la representación gráfica de un objeto indica que un atributo heredado no se aplica a una especialización particular o sus subordinadas. Si se encuentra la situación donde un atributo algunas veces tiene un valor significativo, pero otras no es aplicable, considerar una estructura de clasificación.

5.5.1.3 Identificar Conexiones de Instancia

Una conexión de instancia representa un mapa simple asociando una o varias instancias de un objeto o estructura con una o varias instancias de otro.

Pasos para la identificación y definición de conexiones de instancia:

1. Agregar líneas de conexión de instancia
2. Definir multiplicidad
3. Definir participación
4. Examinar casos especiales

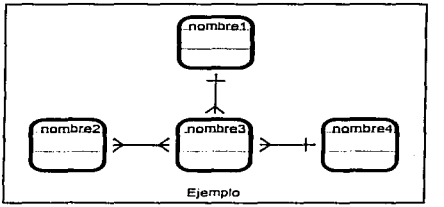
Primero, agregar las líneas de conexión de instancias. Dibujar líneas para reflejar las conexiones del problema entre instancias de objetos (es decir, entre objetos y no entre clases). Limitar las conexiones a aquéllas de interés en el dominio del problema. Cada línea de conexión de instancia implica una correspondiente línea de conexión de mensaje, porque una instancia necesita enviar un mensaje a otra instancia siempre que un identificador de conexión implícito es modificado.

Se debe conectar con una estructura de clasificación al nivel de generalización cuando la conexión se aplica a todas las instancias, en otro caso, sólo a las especializaciones específicas.

El segundo paso es definir multiplicidad. Por cada dirección de una línea de conexión de instancia, considerar la multiplicidad preguntando si para cada dirección, la conexión de instancia entre los objetos es:

una conexión simple (1:1)
una colección de conexiones (1:M)

Añadir una barra simple (vertical) para mostrar una conexión 1:1 y una pata de gallo para mostrar una conexión 1:M.



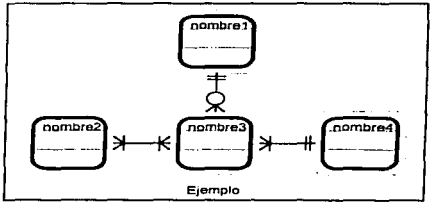
Ejemplo

Por ejemplo, una instancia de nombre2 puede interactuar con algunas instancias de nombre3 (potencialmente una colección de conexiones). Pero una instancia de nombre3 interactúa con una instancia de nombre1 y una de nombre4.

Después de dibujar las líneas de conexión y de agregar multiplicidad, el paso siguiente es definir participación. Preguntando si por cada dirección, la conexión de

Lourdes Montiel García
Isabel Pérez Arredondo

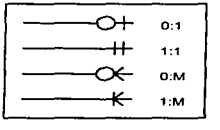
instancia entre objetos es obligatoria u opcional. Agregar una barra simple para conexiones obligatorias y la letra "o" para conexiones opcionales.



Ejemplo

Las conexiones entre nombre2, nombre3 y nombre4 son obligatorias. Estas conexiones indican que para una instancia de nombre2 las correspondientes instancias de nombre1, nombre3 y nombre4 deben estar presentes. Una instancia de nombre1 puede o no tener alguna conexión a alguna instancia de nombre3.

Se pueden combinar estos últimos dos pasos (multiplicidad y participación). La notación siguiente refleja esta idea:



aunque a veces sea más claro enfocarse primero en el número de conexiones potenciales y luego establecer si una conexión es requerida o no.

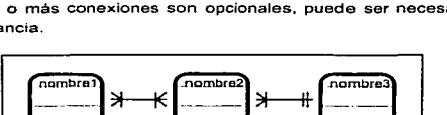
Lourdes Montiel García
Isabel Pérez Arredondo

El paso final es examinar casos especiales. Los tipos más comunes que existen son:

1. Conexiones a través de tres o más objetos o estructuras de clasificación
2. Conexiones de instancia muchos a muchos
3. Conexiones de instancia entre instancias del mismo objeto o estructura de clasificación
4. Conexiones de instancia múltiple entre dos objetos

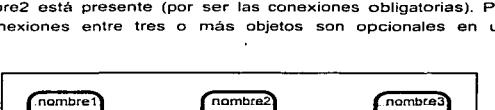
1. Conexiones a través de tres o más objetos o estructuras de clasificación

Cuando una o más conexiones son opcionales, puede ser necesario añadir otras conexiones de instancia.



Conexiones de Instancia Obligatorias

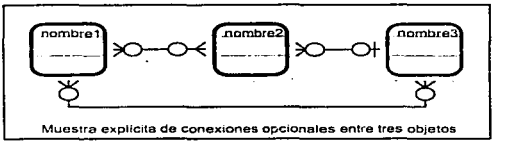
En la figura anterior, nombre1 y nombre2 tienen una conexión obligatoria, nombre2 y nombre3 también tienen una conexión obligatoria. El mapa entre nombre1 y nombre3 a través de nombre2 está presente (por ser las conexiones obligatorias). Pero qué pasa cuando las conexiones entre tres o más objetos son opcionales en una o ambas direcciones.



Conexiones de Instancia Opcionales

Lourdes Montiel García
Isabel Perez Arredondo

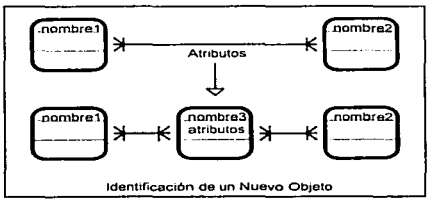
En la figura anterior las conexiones entre nombre1 y nombre2 y entre nombre2 y nombre3 son opcionales. Hay que preguntarse si una instancia de nombre1 y una de nombre3 pueden estar presentes sin que ninguna de las dos tenga una conexión con nombre2. Si es así, agregar una conexión de instancia explícita entre nombre1 y nombre3, que quedarían de la siguiente forma:



Muestra explícita de conexiones opcionales entre tres objetos

2. Conexiones de instancia muchos a muchos

Para una conexión de este tipo, revisar si algunos atributos actualmente describen la conexión entre objetos. Esta redundancia significa que los atributos no deben estar realmente con un objeto u otro, pero describen la conexión entre ellos cuando están conectados. Si es así, se ha identificado un nuevo objeto, como se muestra a continuación:



Identificación de un Nuevo Objeto

Lourdes Montiel García
Isabel Pérez Arredondo

Por lo tanto, agregar un nuevo objeto y colocar los atributos descriptivos (más un nombre) en el nuevo objeto. Agregar la multiplicidad correspondiente.

3. Conexiones de instancia entre instancias del mismo objeto o estructura de clasificación

Una instancia de un objeto puede tener conexiones de instancia potenciales con otras instancias del mismo objeto. Revisar si una conexión de instancia tiene atributos descriptivos. Si es así, introducir un nuevo objeto para capturar la conexión de instancia.

4. Conexiones de instancia múltiples entre dos objetos

Cuando se descubran más de una conexión de instancia entre dos objetos, retroceder y considerar el significado de las dos conexiones. Capturar la distinción semántica fundamental con uno o más atributos. Agregar los atributos para capturar la distinción y colocar solo una conexión de instancia entre los dos objetos en el modelo.

5.5.1.4 Referente a los Atributos de Identificación

Los atributos de identificación proveen un medio conveniente para describir una instancia de un objeto más las conexiones para otras instancias.

Cada objeto y estructura de clasificación necesita tales identificadores. Así que por convención, para mantener los diagramas lo más simples, cada objeto y estructura de clasificación tiene un identificador implícito (oid es el identificador de objeto) e identificadores de conexión (cid es el identificador de conexión).

La razón primaria para denotar estos identificadores es su conveniencia en la especificacion de los Servicios. En el modelo de AOO se quiere que cada instancia de un objeto sea informada y en este caso, conocer acerca de sus conexiones con otras instancias.

Otra razón para el uso de atributos identificadores implícitos, es evitar seleccionar identificadores "del mundo real" como identificadores unicos. Un identificador único debe

ser único y no cambiar una vez que está en el sistema. Este aislamiento significa efectuar algunas selecciones de diseño porque los identificadores del mundo real no pueden garantizarse como únicos

Nombrar a cada atributo con un sustantivo singular o un adjetivo+sustantivo. Escribir los nombres completos, evitando códigos especiales, conservar la consistencia.

5.5.1.5 Revisar Objetos

Cuando se añaden atributos, se requiere revisar algunos objetos o estructuras de clasificación. Para ello se mencionan las siguientes guías:

Atributos "No Aplicables" Si algunos atributos no aplican a todas las instancias de un objeto o a una instancia particular de una estructura de clasificación, entonces considerar la introducción de estructuras de clasificación adicionales.

Un solo Atributo. Si un objeto o una instancia de una estructura de clasificación tiene solamente un atributo, entonces el modelo puede ser revisado para reflejar un nivel más alto de abstracción. Mover el atributo directamente al objeto que describe y remover el objeto innecesario.

Valores Repetidos para uno o más Atributos. Si un atributo de una instancia de objeto tiene valores repetibles, considerar la agregación de un nuevo objeto. Esto es, probablemente se deban colocar en el nuevo objeto los atributos repetibles, de forma que resulten dos objetos, cada uno con más de un atributo.

5.5.1.6 Especificar Atributos

Especificar los atributos con nombres y descripciones, dependiendo de lo que ayude a entender la descripción o de lo que el cliente requiere, también se pueden añadir ciertas restricciones de los atributos (valores permitidos, rango, límite, unidad de medida, etc.).

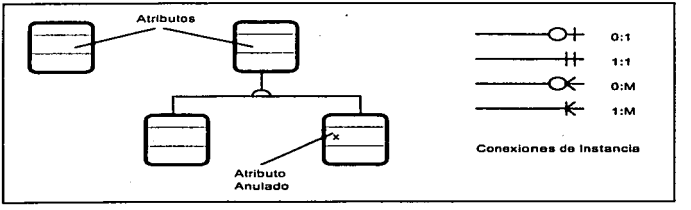
También identificar la categoría de cada atributo:

- Descriptivo - el valor es establecido y mantenido
- Definición - el valor es potencialmente aplicable a más de una instancia de un objeto o estructura de clasificación
- Siempre derivable - el valor es derivable a partir de otros datos en cualquier momento (por lo tanto, el valor no requiere ser conservado)
- Ocasionalmente derivable - el valor es derivable sólo ocasionalmente (el valor requiere ser conservado)

5.5.1.7 Especificar las Restricciones de Conexión de Instancia

Especificarlas desde la perspectiva de una instancia y sus restricciones de conexión con otras. También incluir las restricciones para cualesquiera conexiones ensambladoras en las que un objeto participa como un todo o una parte.

5.5.2 Notación



Lourdes Montiel García
Isabel Pérez Arredondo

5.6 DEFINICION DE SERVICIOS

La cuestión principal en la definición de servicios es definir el comportamiento requerido para cada objeto y estructura de clasificación.

Las estrategias para la identificación de servicios son tres:

- Causalidad Inmediata - Estado-Evento-Respuesta
- Historia - Historia del Objeto
- Función - Servicios Fundamentales

Una segunda cuestión en la definición de servicios es definir la comunicación necesaria entre instancias de objetos (interacción de mensajes). Servicios y conexiones de mensaje son especificados en el Depósito de AOO, con una especificación textual de los requerimientos del proceso observables y cuantificables.

5.6.1 Cómo Definir Servicios

La estrategia consiste de cuatro pasos:

- Identificar los Servicios (Estrategia Primaria). Este paso obtiene los nombres de servicios sobre el diagrama.
- Identificar los Servicios (Estrategia Secundaria). Este paso obtiene nombres de servicios adicionales sobre el diagrama.
- Identificar Conexiones de Mensajes. Este paso establece las necesidades de proceso inter-instancia.
- Especificar los Servicios. Este paso desarrolla los requerimientos de proceso.

5.6.1.1 Identificar los Servicios (Estrategia Primaria)

Para identificar los servicios en el modelo, considerar tres servicios fundamentales para cada objeto o estructura de clasificación:

1. Ocurrir (instancia para añadir, cambiar, borrar y seleccionar)
2. Calcular
3. Vigilar

1. Ocurrir

Este servicio establece y mantiene una instancia de un objeto o estructura de clasificación, con opciones añadir, cambiar, borrar y seleccionar. Cada objeto y estructura de clasificación requiere este servicio. En vez de poner la palabra ocurrir en cada objeto o estructura de clasificación en un diagrama, este servicio es tratado como implícito:

- No aparece en el diagrama de AOO
- Es especificado una vez, como un servicio implícito para todos
- Puede ser anulado por un objeto o una estructura de clasificación, cuando sea necesario

2. Calcular

Este servicio calcula resultados para una instancia o en nombre de otra instancia. Cuando este servicio varía, añadir la apropiada estructura de clasificación al diagrama. Cuando se añada este servicio a la estructura de clasificación, mostrar el servicio general más alto y las especializaciones más bajo.

3. Vigilar

Este servicio efectúa una vigilancia de un sistema externo, dispositivo o usuario. Añadir este servicio para instancias que deben vigilar otro sistema o un dispositivo. Cuando este servicio varía, añadir la apropiada estructura de clasificación. Cuando se añada este servicio a la estructura de clasificación, mostrar el servicio general más alto y las especializaciones más bajo.

5.6.1.2 Identificar los Servicios (Estrategia Secundaria)

1. *Historia de la Vida del Objeto.* El método consiste de:
- Definir la Secuencia Histórica Básica. Dibujar la secuencia básica para un objeto o estructura de clasificación.

Lourdes Montrel García
Isabel Perez Arredondo

- Revisar Variaciones en Cada Paso. Expandir cada paso, mostrando variaciones a la secuencia básica.
- Agregar a la Secuencia Básica. Esto es, a qué otros eventos el objeto o estructura de clasificación responde.
- Agregar Servicios. Agregar a los servicios fundamentales cuando sea necesario, de manera que cada proceso sea descrito correctamente a través de la historia del objeto.

Documentar la vida histórica del objeto en la sección "VidaHistóricaObjeto" del patrón de especificación.

2. *Estado-Evento-Respuesta* Esta estrategia tiene los siguientes pasos:

- Definir los principales estados del sistema. Esto es, los estados de comportamiento del mismo.
- Para cada estado, listar los eventos externos y las respuestas requeridas y colocar esta información en una tabla Estado-Evento-Respuesta.
- Expandir los Servicios (y Conexiones de Mensaje). Para proveer los procesos requeridos en respuesta a cada evento.

Documentar la tabla mencionada en la sección "EstadoEventoRespuesta" del patrón de especificación.

5.6.1.3 Identificar Conexiones de Mensajes

Una conexión de mensaje combina las perspectivas Evento-Respuesta y Flujo de Datos; esto es, cada una de ellas representa un mensaje que es enviado y también una respuesta que es recibida. Son usadas para acomodar las necesidades de Servicios.

Una conexión de mensaje relaciona una instancia con otra, en la cual un "remitente" envía un mensaje a un "receptor", para obtener algún proceso. El proceso requerido es nombrado en la especificación de servicios del remitente y es definido en la especificación de servicios del receptor.

Agregar conexiones de mensaje entre objetos y estructuras de clasificación conectadas con conexiones de instancia; también, examinar los objetos y estructuras de clasificación junto con los atributos encapsulados en ellos, buscando los servicios requeridos por una instancia u otra.

La representación gráfica de una Conexión de Mensaje es mostrada a continuación:

Una flecha junto con una cabeza de flecha indica que una conexión de mensaje existe (el remitente envía un mensaje; el receptor recibe el mensaje; el receptor regresa una respuesta al remitente). Una doble cabeza en una flecha muestra mensajes que son enviados desde cada uno de los objetos o estructuras de clasificación participantes a los otros.

Documentar las conexiones de mensaje en la especificación de servicios del remitente; documentar el servicio ejecutado correspondiente en la especificación de servicios del receptor. Para encontrar conexiones de mensaje adicionales se deben documentar los detalles de todos los servicios.

5.6.1.4 Especificar ios Servicios

La especificación de servicios es presentada en las siguientes secciones:

- Enfocarse en el Comportamiento Externamente Observable
- Usar un Patrón
- Agregar Diagramas para Simplificar las Especificaciones de Servicios

- Agregar Tablas de Soporte
- Desarrollar la Redacción de los Servicios
- Colocar Toda la Documentación Junta

Usar un Patrón. La estructura general de dicho patrón es la siguiente:

especificación <Nombre del Objeto>

atributosDescriptivos <...>
definiciónDatos <...>
atributosSiempreDerivables <...>
atributosOcasionalmenteDerivables <...>

entradaSistema <...>
salidaSistema <...>

restriccionesConexionesInstancia <...>

estadoEventoRespuesta <...>
historiaVidaObjeto <...>

notas <...>

propósito <...>

servicio <Nombre>
servicio <Nombre>
servicio <Nombre>

fin especificación

Lourdes Montiel García
Isabel Pérez Arredondo

En donde:

atributosDescriptivos Atributo(s) cuyos valores son manipulados por servicios "Ocurrir" (agregar, cambiar, borrar y seleccionar). Listar restricciones de instancia si se requieren.

definiciónDatos Atributo(s) cuyos valores son potencialmente aplicables a más de una instancia de un objeto o clasificación.

atributosSiempreDerivables Atributos derivables en cualquier momento (por lo tanto, no se requiere mantener el dato a través del tiempo). Listar restricciones de instancia si se requieren.

atributosOcasionalmenteDerivables Atributos derivables sólo ocasionalmente (se requiere mantener registro de ellos). Listar restricciones de instancia si se requieren.

entradaSistema Entrada desde un dispositivo o sistema externo. Listar restricciones de datos si se requieren.

salidaSistema Salida a un dispositivo o sistema externo. Listar restricciones de datos si se requieren.

restriccionesConexionesInstancia Las restricciones de conexión de instancia para una instancia.

estadoEventoRespuesta Las mismas usadas en la identificación de servicios.

historiaVidaObjeto El patrón de vida histórica de un objeto usada en la identificación de servicios.

notas Consideraciones del Analista.

propósito El propósito del requerimiento.

servicio La especificación de un servicio, una lista de requerimientos.

El patrón puede ser acortado para colocar las necesidades específicas de un proyecto y también puede ser extendida por si se requiere información adicional en algún contexto.

Agregar Diagramas para Simplificar las Especificaciones de Servicios. La mayoría de las especificaciones de servicios estarán bien estructuradas y razonablemente cortas, pero se deben usar diagramas para guiar al lector a través de los pasos del proceso en un servicio particular.

Agregar Tablas de Soporte. Las dependencias inter-objetos son capturadas en un modelo con conexiones de mensajes. Para resumir las interacciones utilizar: (1) un resumen de servicios y estados aplicables, (2) análisis de secuencias de ejecución y (3) análisis de presupuestos.

Colocar Toda la Documentación Junta. El paquete completo contiene:

- Diagramas del AOO (Capas de Materia, Objeto, Estructura, Atributo y Servicio)
- Depósito del AOO (Una entrada por servicio o estructura de clasificación)
- Tablas de Soporte, si las hay.

5.7 DISEÑO ORIENTADO A OBJETOS

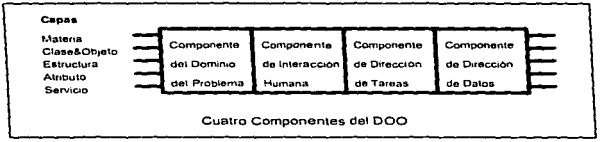
Las capas del AOO modelan el espacio del problema. El Diseño Orientado a Objetos (DOO) modela un espacio de implantación particular de requerimientos. La expansión ocurre principalmente con Clases y Objetos añadidos.

El modelo de DOO, al igual que el modelo de AOO, consta de cinco capas: Materia, Clase&Objeto, Estructura, Atributo y Servicio. Estas cinco capas gradualmente presentan más y más detalle del modelo del problema y corresponden, en un enfoque general, a las cinco actividades principales introducidas en el AOO: Búsqueda de Clases y Objetos, Identificación de Estructuras, Identificación de Materia, Definición de Atributos y Definición de Servicios. Estas son, en efecto, actividades y no necesariamente pasos secuenciales.

En adición a lo anterior, el modelo del DOO consta de cuatro componentes:

- Componente del Dominio del Problema
- Componente de Interacción Humana
- Componente de Dirección de Tareas
- Componente de Dirección de Datos

Los componentes son rebanadas verticales del modelo general. Desde un enfoque general, los cuatro componentes corresponden a las cuatro principales actividades del DOO: Diseño del componente del dominio del problema, Diseño del componente de interacción humana, Diseño del componente de dirección de tareas y Diseño del componente de dirección de datos.

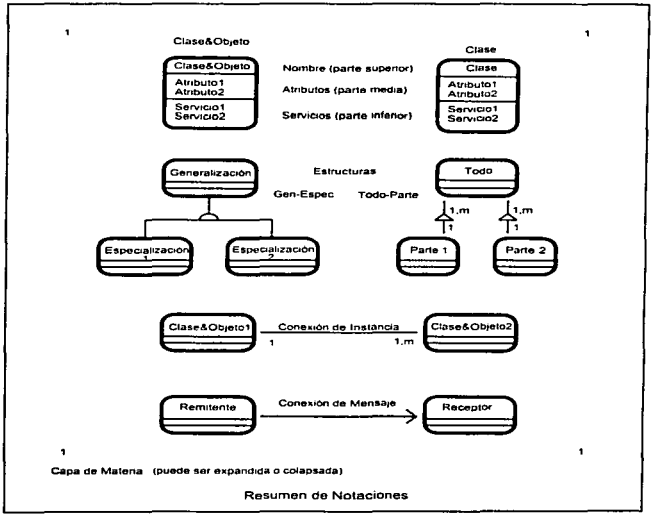


Cuatro Componentes del DOO

Los resultados del AOO son colocados directamente en el Componente del Dominio del Problema. Dentro de este componente existe la necesidad potencial de realizar combinaciones y separaciones de ciertas clases, estructuras, atributos y servicios. El componente de interacción humana incluye los despliegados y entradas reales requeridas para una efectiva interacción humano-computadora. El componente de dirección de tareas incluye definición de tareas, comunicación y coordinación. El componente de dirección de datos incluye el acceso y manejo de datos persistentes.

Lourdes Montiel García
Isabel Pérez Arredondo

5.7.1 Resumen de Notaciones



Resumen de Notaciones

Lourdes Montiel García
Isabel Pérez Arredondo

5.8 DISEÑO DEL COMPONENTE DEL DOMINIO DEL PROBLEMA

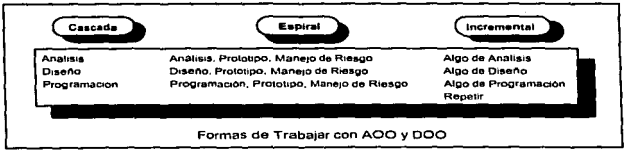
En el DOO los resultados del AOO se colocan en el Componente del Dominio del Problema (CDP). Los resultados del análisis son una parte integral del modelo multicomponente del DOO. El diseño CDP sigue la siguiente estrategia:

- Aplicar el AOO
- Usar los resultados del AOO -mejorarlos durante el DOO
- Usar los resultados del AOO -aumentarlos durante el DOO

5.8.1 Aplicar el AOO

Un método requiere notación (una representación) y estrategias (cómo realizar el trabajo). El AOO y el DOO usan la misma notación. El AOO es organizado alrededor de cinco capas, el DOO es organizado alrededor de cuatro componentes.

La forma de proceder en la aplicación de AOO y DOO puede ser en cascada, en espiral o en desarrollo incremental. Tanto el AOO como el DOO trabajará con cualquiera de estos procesos.



Formas de Trabajar con AOO y DOO

Lourdes Montiel García
Isabel Perez Arredondo

5.8.2 Usar los resultados del AOO -mejorarlos durante el DOO

Poner los resultados del AOO directamente en el CDP. Algunas modificaciones probablemente ocurrirán, algunas de ellas son debidas a cambios en los requerimientos y otros se deben a la falta de entendimiento del dominio del problema por parte del analista. En cualquier caso, simplemente cambiar los resultados del AOO y después reflejarlos en el CDP.

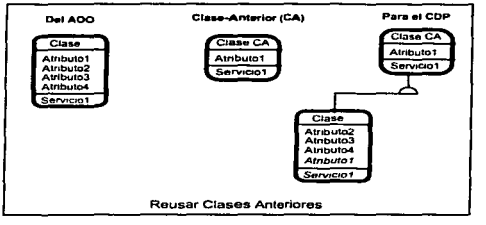
5.8.3 Usar los resultados del AOO -añadirles durante el DOO

El criterio de adición del CDP incluye:

- Reusar clases de diseño y programación
- Agrupar clases de un problema específico
- Establecer un protocolo añadiendo una clase de generalización
- Acomodar el nivel de herencia soportado
- Soportar el componente de dirección de datos
- Agregar componentes de bajo nivel
- No modificar solo para reflejar asignaciones entre equipos
- Revisar las adiciones a los resultados del AOO

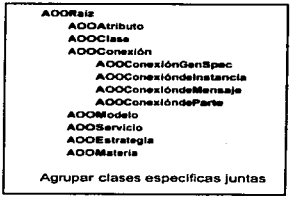
Reusar clases de diseño y programación. Añadir una Clase-Anterior que ya haya sido elaborada al CDP. Identificar los atributos o servicios dentro de esta clase que no vayan a ser usados más. Agregar una especialización Gen-Espec, desde la Clase-Anterior hacia la Clase del problema. Después, identificar la porción de la Clase del problema que no va a ser más requerida en esa clase, aquellos atributos y servicios que ahora se heredarán de la Clase-Anterior.

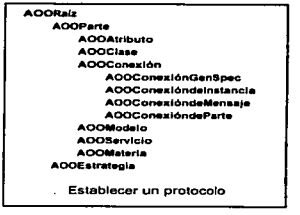
Lourdes Montiel García
Isabel Pérez Arradondo



Agrupar clases de un problema específico. En el AOO, una clase no es introducida hasta arriba o en el inicio de todas las clases, sólo para actuar como una Clase Raíz y agrupar a todas las subordinadas. Sin embargo, en el DOO una Clase Raíz puede ser introducida principalmente para mantener juntas las clases de un problema específico dentro de una Librería de Clases, tal como la Clase "ClaseRaíz" que se muestra en el ejemplo:

Lourdes Montiel García
Isabel Pérez Arredondo





Establecer un protocolo añadiendo una clase de generalización. A veces un número de clases de especialización necesitan definir un conjunto similar de servicios (y probablemente, los correspondientes atributos). Cuando este es el caso, una clase adicional puede ser introducida para establecer ese protocolo -el nombre de un conjunto común de servicios- a ser definido en detalle dentro de las clases de especialización.

Acomodar el nivel de herencia soportado. Si las estructuras Gen-Espec del AOO incluyen herencia múltiple, algunas modificaciones serán necesarias cuando se use un lenguaje de programación con herencia simple o cero.

Con un lenguaje de herencia simple, dos procesos pueden ser aplicados para pasar las estructuras de herencia múltiple a unas de herencia simple: (1) *Descomponer en jerarquías múltiples, con relaciones entre ellas.* En este método, descomponer el patrón de herencia múltiple en dos o más jerarquías con las relaciones entre ellas, esto es, una estructura todo-parte o una conexión de instancia. (2) *Achatar a una jerarquía simple.* En este método, una o más generalizaciones-especializaciones ya no serán explícitas en el diseño, lo que quiere decir que algunos atributos y servicios serán repetidos en las clases de especialización. Con un lenguaje de herencia cero, cada estructura Gen-Espec necesitará achatar la jerarquía en un grupo de clases&objetos.

Soportar el componente de dirección de datos. Para hacerlo, cada objeto a ser almacenado necesita conocer cómo almacenarse él mismo. Hay dos métodos posibles: cada objeto se salva solo o cada objeto se envía al componente de dirección de datos, el cual lo salva.

Para el primer método: decirle a un objeto que se salve (cada objeto sabe cómo salvarse), para ello añadir un atributo y un servicio para definir esto. Para el segundo método, decirle a un objeto que se salve (cada objeto sabe qué mensajes enviar al Componente de Dirección de Datos para que su estado sea salvado), para ello añadir atributos y servicios para definir esto.

Para ambos métodos, los atributos y servicios requeridos son los siguientes: (1) Con herencia simple, modificar el CDP directamente: añadir un atributo para identificar a un objeto que debe estar junto a una clase particular, añadir un servicio para definir cómo un objeto va a almacenar sus valores, mantenerlos a ambos implícitos (no incluidos en los diagramas pero definidos en los textos de cada símbolo Clase&Objeto). (2) Con herencia múltiple: poner estas adiciones en una nueva clase, después modificar cada clase del CDP con objetos almacenables para incluir una clase de generalización adicional, colocar un atributo para identificar que un objeto debe estar junto con una clase particular, añadir un servicio para definir cómo un objeto va a almacenar sus valores y mantener la herencia implícita: no en diagramas pero definido en el texto de cada símbolo de Clase&Objeto.

5.9 DISEÑO DEL COMPONENTE DE INTERACCION HUMANA

En el DOO, el Componente de Interacción Humana (CIH) añade a los resultados el diseño y los detalles de la interacción, este componente captura cómo el humano manejará el sistema y cómo el sistema presentará la información al usuario. La estrategia para diseñar este componente consiste de lo siguiente:

- Clasificar los humanos
- Describir a los humanos y sus tareas
- Diseñar la jerarquía de comandos
- Diseñar la interacción detallada
- Continuar el prototipo
- Diseñar las clases de CIH
- Diseñar la Interface Gráfica del Usuario (cuando sea necesario)

5.9.1 Clasificar los humanos

Pensar acerca de lo que la gente quiere lograr, de las tareas que necesita efectuar, las herramientas que pueden proveérseles para realizar esa tarea y la forma de hacer esas herramientas más discretas.

Empezar clasificando a los humanos en diferentes categorías (si una estructura Gen-Espec está presente en el CDP, usarlo como un punto de inicio). Después considerar subconjuntos adicionales. Se puede considerar la clasificación por: experiencia, nivel en la organización o membresía en diferentes grupos.

5.9.2 Describir a los humanos y sus tareas

Para cada categoría de humanos definida considerar y tabular lo siguiente (de preferencia en primera persona):

Quién
Propósito
Características (edad, educación, limitaciones, etc.)

Lourdes Montiel García
Isabel Pérez Arredondo

Nombre:	DLGCNES
Documentación:	*Catálogo de delegaciones y municipios*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición no 1
Espacio de complejidad:	Longitud por registro 42 bytes

Nombre:	PSES
Documentación:	*Catálogo de nacionalidades*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No 1
Espacio de complejidad:	Longitud por registro 42 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

- El usuario no debe ser abandonado sin ninguna retroalimentación en: (1) qué progreso está siendo hecho y (2) la cantidad de progreso que se ha hecho, siempre que el usuario deba esperar a que el sistema realice alguna acción
- Deshacer. Los humanos suelen cometer errores y siempre es muy bueno encontrar la forma de deshacer algunos de ellos
- Programar la computadora, no la persona. La gente no debe recordar o escribir información de una pantalla a otra, su memoria debe ser utilizada lo menormente posible
- Tiempo y esfuerzo para aprender. Proveer una referencia en línea para características más avanzadas

5.9.5 Continuar el prototipo

El realizar el prototipo de la interacción hombre-máquina es esencial para el diseño de la CIH. La gente necesita experimentar, jugar con y refinar la interacción propuesta. La interface debe ser discreta (la gente invierte el menor tiempo en entender y manejar la interface e invierte más tiempo realizando el trabajo que debe realizar).

5.9.6 Diseñar las clases de CIH

Este tipo de clases variará dependiendo de la interface gráfica usada. Comenzar organizando el diseño de interacción humana a través de ventanas y componentes.

5.10 DISEÑO DEL COMPONENTE DE DIRECCION DE TAREAS

El Componente del Dirección de Tareas (CDT) tiene como punto principal el identificar y diseñar las tareas más los servicios incluidos en cada una y se hace necesario bajo ciertas circunstancias especiales en sistemas que requieren de múltiples tareas, tales como: ciertas interfaces humanas (en las que múltiples ventanas pueden ser seleccionadas simultáneamente para entradas), para tareas múltiples en un procesador, una tarea adicional puede requerirse para coordinar y comunicar tareas durante su ejecución. El definir tareas simplifica el diseño y la codificación cuando es necesario comportamiento concurrente, sin embargo, las tareas añaden complejidad durante el

diseño, codificación, prueba y mantenimiento, así que cada una debe ser cuidadosamente seleccionada e incluso justificada.

La selección y justificación de tareas, sigue la siguiente estrategia:

- Identificar tareas manejadas por eventos
- Identificar tareas manejadas por reloj
- Identificar tareas prioritarias y críticas
- Identificar un coordinador
- Evaluar cada tarea
- Definir cada tarea

5.10.1 Identificar tareas manejadas por eventos

Estas tareas pueden ser responsables de la comunicación con un dispositivo, una o más ventanas, otra tarea, un subsistema u otro sistema.

Una tarea puede ser diseñada para entrar en funcionamiento con base en un evento, normalmente señalando el arribo de algún dato. Esa entrada puede provenir de una línea de entrada o de un buffer.

La forma de trabajo de la tarea cuando el sistema está corriendo es: duerme (no consume tiempo de procesador) esperando la interrupción de alguna fuente; tras la recepción de la interrupción despierta, procesa el dato, lo coloca en algún destino, notifica a quien tenga que saber acerca de él y vuelve a dormir.

5.10.2 Identificar tareas manejadas por reloj

Estas tareas son puestas en funcionamiento para realizar algún proceso en un intervalo específico de tiempo. Ciertos dispositivos pueden requerir adquisición y control periódico de datos. Ciertas interfaces humanas pueden requerir comunicación periódica.

Lourdes Montiel García
Isabel Pérez Arredondo

La forma de trabajo de estas tareas cuando el sistema está corriendo es: establece un tiempo para despertar y va a dormir, lo que hace esperando una interrupción del sistema. Tras la recepción de esa interrupción despierta, hace su trabajo, notifica a quien necesite saber acerca de ello y después regresa a dormir.

5.10.3 Identificar tareas prioritarias y críticas

Una tarea prioritaria acomoda de acuerdo a las necesidades del proceso alta o baja prioridad. Una tarea crítica es usada para aislar cierto proceso que es especialmente crítico para el éxito o falla del sistema, proceso que suele tener estrictas restricciones de fiabilidad.

Algunos servicios pueden ser de muy alta prioridad y pueden requerir ser aislados en una tarea separada para realizarse en un tiempo urgente. Al otro extremo, algunos servicios pueden ser de baja prioridad y relegados (background). Tareas adicionales pueden ser usadas para aislar tales procesos. Algunos servicios pueden ser altamente críticos para la continua operación del sistema y pueden usarse tareas adicionales para aislar ese proceso crítico, esto particiona el diseño, programación y pruebas requeridas para un proceso altamente confiable.

5.10.4 Identificar un coordinador

Con tres o más tareas es conveniente el considerar añadir una más, la cual puede actuar como un coordinador. Esta tarea añade más trabajo; el tiempo para cambiar de una tarea a otra puede desalentar este diseño. Sin embargo, trae el beneficio de encapsular la coordinación intertareas. Usar este tipo de tareas para coordinar las tareas y nada más, no se use para implementar servicios que estarían mejor en una Clase&Objeto colocado en alguna tarea.

5.10.5 Evaluar cada tarea

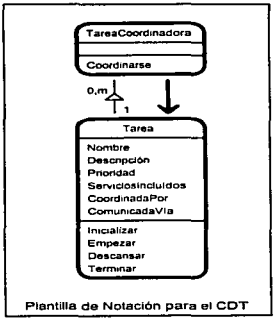
Mantener el número de tareas al mínimo. El solo hecho de tener una o varias cosas ocurriendo a un mismo tiempo puede ser evaluación suficiente en términos del

entendimiento durante el desarrollo y el mantenimiento. Así que, asegurarse que cada tarea satisface uno o más de los criterios de ingeniería para selección de tareas: manejadas por eventos, por reloj, prioritaria/crítica o de coordinación.

5.10.6 Definir cada tarea

Definir cada tarea de acuerdo a tres claves:

¿Qué es? Comenzar especificando el nombre de la tarea y describiéndola brevemente. Añadir una nueva restricción -el nombre de la tarea- a cada servicio en los componentes del DOO. Si un servicio es dividido a través de más de una tarea, entonces modificar los nombres y descripciones de los servicios, de forma que cada servicio pueda ser relacionado con una tarea.



Plantilla de Notación para el CDT

¿Cómo se coordina? Definir cómo se coordina cada tarea. Indicar si es manejada por eventos o por reloj. Por cada una que se maneje por eventos describir dichos eventos, para las que son manejadas por reloj describir los intervalos que deben transcurrir antes de que sea puesta en funcionamiento, además indicar si es repetitivo o en una sola ocasión.

¿Cómo se comunica? Definir cómo se comunica cada tarea. Definir de dónde obtiene sus valores y a dónde los envía después.

5.11 DISEÑO DEL COMPONENTE DE DIRECCION DE DATOS

El Componente de Dirección de Datos (CDD) provee la infraestructura para el almacenamiento y recuperación de objetos en un sistema manejador de datos. Los tres principales métodos de manejo de datos son archivos planos, relacionales y orientados a objetos.

Manejo de Archivos Planos. Ofrece un manejo rudimentario de archivos y ciertas capacidades de ordenación.

Sistema Manejador de Bases de Datos Relacionales. Este tipo de sistemas están construidos sobre la teoría relacional y manejan datos en cierto número de tablas. Cada tabla tiene un nombre al igual que sus columnas. Cada línea representa un conjunto de valores dentro de la tabla y cada columna tiene un solo valor en ella (atómica). Las tablas, el nombre, los nombres de columnas y las restricciones de columnas son definidas en un esquema de la base de datos.

Operaciones especiales hacen posible cortar y pegar tablas basadas en criterios específicos para hacer tablas adicionales. Entre dichas operaciones están: select (para extraer ciertas líneas), project (para extraer ciertas columnas) y join (emparejar líneas entre tablas y después extraer ciertas líneas). Usando estas operaciones, las relaciones interlíneas no tienen que ser predefinidas (como en el manejo de archivos planos).

Cada línea debe ser identificable de manera única. Una o más columnas pueden ser definidas como la llave primaria -el único identificador para cada línea en la tabla. Una

o más columnas pueden ser definidas como una llave exterior, para facilitar el acceso a líneas en otra tabla.

Las tablas y sus columnas pueden ser organizadas para reducir la redundancia de datos y consecuentemente reducir el número de pasos necesarios para modificar los datos consistentemente. Esta disciplina es llamada Normalización y el grado de eliminación de redundancia de datos es definido en Formas Normales.

- Primera forma normal. El valor del atributo debe ser atómico (esto es que por cada atributo, no hay repetición de valores ni estructura interna de datos).
- Segunda forma normal. La primera forma normal más, cada atributo que no es llave describe algo identificable solamente por la llave entera (y no sólo por parte de la llave).
- Tercera forma normal. La segunda forma normal más, cada atributo que no es llave depende solamente de la llave y no es solo una descripción más detallada de otro atributo no llave.
- Cuarta forma normal. La tercera forma normal más, los valores de dos o más atributos que no son llave no siempre se relacionan con otro atributo que no es llave.
- Quinta forma normal. La cuarta forma normal más, los valores de dos o más atributos que no son llave no siempre se relacionan con otro atributo que no es llave, siguiendo una operación join.

Sistema manejador de bases de datos orientado a objetos. Las dos principales propuestas dentro de ellos son: relacional extendido y LPOO extendido.

Los productos relacionales extendidos aumentan un sistema de manejo de bases de datos relacional, añadiendo tipos de datos abstractos y herencia, algunos servicios de propósito general para crear y manipular clases y objetos.

Los productos de LPOO extendidos aumentan un lenguaje de programación orientado a objetos con sintaxis y capacidad para manejar el recuerdo de un objeto a través del tiempo dentro de una base de datos.

5.11.1 Diseñando el CDD

Este diseño requiere incluir tanto el diseño de la distribución de los datos como el de los correspondientes servicios.

Diseño de la Distribución de los Datos. Esta distribución puede hacerse tanto con archivos planos, como relacionales como orientados a objetos.

Con manejo de datos en archivos planos:

Definir la primera forma normal de las tablas
- Listar cada clase y sus atributos
- Pasar esa lista a la primera forma normal para que resulte la definición de las tablas en la primera forma normal

Definir un archivo para cada tabla en la primera forma normal
Medir las necesidades de rendimiento y almacenaje
Después retroceder de la primera forma normal para encontrar los requerimientos de rendimiento y almacenaje
- Si es necesario, colapsar los atributos para una estructura Gen-Espec en un archivo simple. Esto reducirá el número de archivos (a cambio de un poco más de espacio de almacenamiento por los campos vacíos)
- Si es necesario, combinar algunos atributos en alguna forma de valor codificado, mejor que usar campos separados. Esto reducirá la cantidad de almacenamiento necesario (a cambio de añadir proceso necesario para codificar y decodificar dichos campos)

Con manejo de bases de datos relacionales:

Definir las tablas en la tercera forma normal
- Listar cada clase y sus atributos
- Pasar la lista a la tercera forma normal, resultando las definiciones de tablas en la tercera forma normal

Definir una tabla de base de datos para cada tabla en la tercera forma normal

Medir las necesidades de rendimiento y almacenaje. Entonces retornar de la tercera forma normal para encontrar los requerimientos de rendimiento y almacenaje

Con manejo de bases de datos orientadas a objetos:

Propuesta relacional extendida

Aplicar el mismo método descrito para el manejo de bases de datos relacionales

Propuesta de LPOO extendidos

Puede no haber pasos necesarios para normalizar los atributos. El mismo sistema manejador de bases de datos relaciona los valores de los objetos con sus valores almacenados

Diseño de los Servicios Correspondientes. Agregar un atributo y servicio a cada Clase&Objeto con objetos que serán almacenados. Ya que dicho atributo y servicio se aplica a cada uno de estos objetos, tratar a cada uno como implícito.

Con este diseño un objeto sabrá cómo almacenarse solo. El atributo y servicio "me-almaceno-solo" forma un puente necesario entre el CDP y el CDD. Con herencia múltiple, dicho atributo y su correspondiente servicio podría ser definido y luego heredado por cada clase que tiene objetos que requieren ser almacenados.

Con manejo de datos en archivos planos:

Un objeto necesita saber qué archivos abrir, cómo situarse en el registro correcto, cómo recuperar viejos valores y cómo actualizar con nuevos valores

Definir una Clase&Objeto llamado ObjetoServidor con servicios para (1) decirle a un objeto cómo salvarse solo y (2) recuperar objetos almacenados (buscar, obtener valores, crear e inicializar objetos)

Con manejo de bases de datos relacional:

Un objeto necesita saber qué tablas revisar, cómo revisar las líneas requeridas, cómo recuperar los viejos valores y cómo actualizar con nuevos valores.

Además, definir una Clase&Objeto llamada ObjetoServidor con servicios para (1) decirle a un objeto cómo salvarse solo y (2) recuperar objetos almacenados (buscar, obtener valores, crear e inicializar objetos)
Con manejo de bases de datos orientadas a objetos:

Propuesta relacional extendida
El mismo método descrito para el manejo de bases de datos relacionales
Propuesta de LPOO extendidos
No se requieren servicios agregados. El sistema manejador de la base de datos provee el comportamiento "me-almaceno-solo" para cada objeto a ser conservado sobre el tiempo. Solo marcar un objeto cuando sea necesario que sea conservado y el sistema manejador de la base de datos orientado a objetos se preocupará por salvarlo y recuperarlo.

5.12 APLICACION DEL DOO CON LPOOs O MENOS QUE UN LPOO

Para resolver la cuestión de cómo aplicar el DOO con lenguajes de programación orientados a objetos o aquéllos que son menos que un LPOO, se deben tener en cuenta las siguientes cuestiones:

- Ser totalmente pragmático acerca de los lenguajes
- El impacto del lenguaje en el desarrollo OO
- Evaluación de la sintaxis y de las características de cada lenguaje
- Selección de un LPOO

5.12.1 El impacto del lenguaje en el desarrollo OO

Cada lenguaje de programación provee ciertas capacidades para capturar explícitamente la semántica del dominio del problema. Desde una perspectiva orientada a objetos, la sintaxis del lenguaje que captura más y más de dicha semántica es significativo, por varias razones:

- Representación uniforme desde AOO a DOO a POO

Lourdes Montiel García
Isabel Pérez Arredondo

- Reutilización
- Mantenimiento

En la selección y uso de un lenguaje no es tan importante cuál lenguaje es más puro o cuál lenguaje es realmente orientado a objetos, sino mejor cuál lenguaje que esté a nuestra disposición hará el mejor trabajo en la captura de la semántica del dominio del problema.

5.12.2 Evaluación de la sintaxis y de las características de cada lenguaje

Esta evaluación puede realizarse considerando el soporte que se encuentre en cada lenguaje bajo consideración para lo siguiente:

- Clase, Objeto
- Generalización-Especialización
- Todo-Parte
- Atributo, Servicio

Para la Gen-Espec, las cuestiones incluyen herencia y resolución de conflictos de nombres, esta última es requerida para manejar el potencial para nombres duplicados en más de una clase de generalización y sólo se considera con lenguajes de herencia múltiple.

Para atributos, los puntos incluyen el soporte para conexiones de instancia (parte del estado de un objeto), visibilidad y restricciones.

Para servicios, los puntos incluyen el soporte para conexiones de mensaje (describiendo la interacción de los objetos), visibilidad y vinculación dinámica. La vinculación dinámica describe la habilidad de una aplicación para elegir o vincular un servicio particular al tiempo de corrida, exactamente cuando la aplicación requiere ejecutar tal servicio en particular.

- C++ es un lenguaje de programación con extensión que añade características orientadas a objetos al lenguaje de programación estructurada C (por lo que se le considera híbrido).
- Object Pascal es un lenguaje de programación con extensión que añade sintaxis a Pascal (se considera un híbrido).
- Smalltalk fue uno de los primeros LPOOs. Fue originalmente desarrollado por miembros de Software Concepts Group en Xerox PARC y fue liberado a principios de los años 70s.
- Objective-C es un lenguaje de programación derivado de Smalltalk. Fue escrito con el propósito de proveer a los desarrolladores herramientas para la construcción de componentes de software.
- Eiffel es el lenguaje de programación orientado a objetos más cuidadosamente construido. Añade capacidades más allá de otros lenguajes y usa una sintaxis familiar a los lenguajes de programación convencionales.
- Ada es un lenguaje orientado a paquetes. Añade una construcción para empaquetar juntos cierto número de variables y procedimientos, lo que ya es una ventaja sobre lenguajes estructurados.

5.13 CRITERIOS DE APLICACION DEL DOO

Al través del proceso de diseño de los sistemas, el diseñador se debe enfrentar con la toma de decisiones entre alternativas, para ayudarle en dicha elección debe estar armado con una serie de criterios, tales como:

- Acoplamiento
- Cohesión
- Reutilización
- Criterios Adicionales

5.13.1 Acoplamiento

El acoplamiento es la interconexión de piezas en un DOO y es importante durante la evaluación de un diseño porque ayuda a centrarse en una característica muy deseable:

un cambio a una parte de un sistema debe tener un mínimo impacto en otras partes. Buscar conexiones entre Objetos y entre Clases; en el caso ideal, la examinación y entendimiento de un componente debe tener una mínima probabilidad de necesitar el entendimiento de otros componentes.

El grado de acoplamiento entre dos componentes por la cantidad y la complejidad de la información transmitida entre los componentes. En un DOO, hay dos situaciones análogas: el acoplamiento de dos objetos expresado por una conexión de mensaje y el acoplamiento entre clases de generalización y de especialización.

Acoplamiento de Interacción Bajo. En el acoplamiento de interacción un nivel bajo es deseable. La guía fundamental es: mantener la complejidad de una conexión de mensaje tan baja como sea posible, en general, si una conexión de mensaje abarca más de tres parámetros, examinarlos para ver si pueden ser simplificados.

Acoplamiento de Interface Alto. En el acoplamiento de interface un nivel alto es deseable. La herencia es una forma de acoplamiento entre una clase de generalización y una de especialización: se lucha porque una clase esté acoplada con su clase de generalización en términos de los atributos y servicios que hereda. La clave de este asunto es nombrar cuidadosamente las clases y organizarlas para reflejar cuál es más general y cuál es más específica.

5.13.2 Cohesión

Un servicio puede realizar una y sólo una función. Un servicio que realiza muchas funciones o que solamente satisface parte de una función es indeseable. Una forma para medir la cohesión es nombrar al servicio con una frase, generalmente es posible describir las responsabilidades de un servicio que tiene cohesión exactamente con una sentencia conteniendo un verbo y un objeto directo.

El segundo tipo de cohesión es la de clases. Los atributos y servicios deben tener cohesión, sin atributos ni servicios extras y que describan las responsabilidades de un objeto en la clase.

El tercer tipo de cohesión es la de la generalización-especialización. Para cada clase de especialización, se debe preguntar: qué tan interrelacionadas están las clases, si la especialización realmente describe la especialización deseada o si es algo arbitrario y fuera de la estructura.

5.13.3 Reutilización

El beneficio principal de la reutilización del software es una mayor productividad. Los beneficios de la reutilización requieren:

- En primer lugar una inversión para crear los componentes reutilizables. Esta se amortiza a través del número de nuevos sistemas que sean construidos utilizando dichos componentes. Obviamente, entre más a menudo puedan ser reutilizados los componentes, menos onerosa será la inversión inicial.

- Una inversión para lograr más elevados niveles de garantía de calidad que los normalmente esperados para los componentes utilizados una sola vez.

- Una inversión para el mantenimiento de librerías y otras facilidades para que los ingenieros de software puedan encontrar más fácilmente los componentes reutilizables cuando ellos los requieran.

Existen varios niveles de reutilización entre los que se encuentran los siguientes:

Código Reutilizable. Este concepto es usualmente interpretado como convertir una subrutina o un módulo en una librería, pero puede tomar muchas formas:

Copia y pegado de Código Fuente. Esta es la más primitiva forma de reutilización, pero es mejor que ninguna.

Código Incluido en los Lenguajes. Muchos lenguajes de programación tienen facilidades para incorporar código fuente a un programa proveniente de una librería. Tales facilidades son conocidas como "include" o "copy".

Herencia. Los lenguajes de programación orientados a objetos incluyen herencia simple o múltiple. Esto añade sintaxis para capturar la semántica del dominio del problema y de la implantación; también provee un mecanismo de reutilización vía la extensión.

Ligado. Con la mayoría de los lenguajes de programación, la actividad de compilación es seguida por una de ligado (link) con la cual todos los módulos requeridos son colocados juntos. Esto puede incluir módulos de librerías previamente compiladas que el programador invoca en el programa. Si el código reutilizable es modificado deberá ser recompilado y reinsertado en la librería, pero para los otros programas lo único que será necesario es una nueva orden de ligado para incorporar las modificaciones.

Resultados del Diseño Reutilizables. Mejor que sólo reutilizar el código es el reutilizar el modelo del diseño de un programa (por ejemplo, un modelo de DOO). La justificación más evidente para este nivel de reutilización es facilitar el paso de una aplicación a una plataforma totalmente diferente.

Resultados del Análisis Reutilizables. Un más elevado nivel de reutilización es el de las especificaciones contenidas en un modelo AOO. Este nivel sería apropiado si los desarrolladores quisieran convertir un sistema de una tecnología vieja a una más reciente y poderosa. Los requerimientos de los usuarios podrían no tener cambios, pero la arquitectura interna del sistema podría ser totalmente nueva.

El componente más importante para lograr altos niveles de reutilización del software en las organizaciones es la Dirección y hay tres cosas que debe hacer: (1) Proveer mecanismos para incentivar una mayor conciencia de los deseable que es la reutilización, (2) Proveer un mando que aliente dicha reutilización y (3) Crear un grupo cuyo único trabajo sea la creación de componentes reutilizables.

5.13.4 Criterios Adicionales

Claridad de Diseño. Esto es especialmente importante por el énfasis en la reutilización en el DOO: si otro diseñador no puede entender un diseño ajeno,

seguramente no lo reutilizará. Algunos de los factores importantes que contribuyen para que un diseño sea lo suficientemente claro para que sea reutilizado son: usar un vocabulario consistente, adherirse al comportamiento o protocolo existente, evitar demasiados patrones de mensajes (deben tener un formato consistente) y atender a las responsabilidades al momento de definir las clases.

Profundidad de la Generalización-Especialización. Esta profundidad se verá afectada con el lenguaje de programación que se utilice, sin embargo, una jerarquía excesivamente profunda puede ser difícil de modificar posteriormente. Otro error es tratar de especializar demasiado en una clase existente, una buena forma de probar es decir "X (la clase especializada) es una Y (la clase general)" y ver si tiene sentido.

Mantener Objetos y Clases simples. Para ello atender los puntos siguientes:

- Evitar atributos excesivos. Se pueden requerir en promedio uno o dos atributos por servicio en un objeto. Si hay muchos atributos, la clase es demasiado compleja y está tratando de hacer demasiado.

- Minimizar las colaboraciones de los objetos. En ocasiones un objeto realiza relativamente poco proceso por sí mismo, pero colabora con otros objetos para lograr algo. En la mayoría de los casos un objeto sólo debe colaborar con 3 ó 5 objetos.

- Evitar el tener demasiados servicios por clase. Normalmente una clase no tiene más de seis o siete servicios públicos, pero puede tener además algunos servicios privados.

Mantener Protocolos simples. El vocabulario en el protocolo de mensajes debe conservarse tan simple como sea posible. Si un mensaje requiere más de tres parámetros, algo anda mal.

Mantener Servicios simples. Los servicios del DOO típicamente son bastante pequeños. Si el servicio es demasiado largo, puede indicar que el servicio se ha

organizado con banderas o con instrucciones "case". En general, si el servicio parece un programa estructurado, las clases han sido incorrectamente elegidas (en lugar de usar "case" en varios servicios adecuar una estructura Gen-Espec).

Minimizar la Volatilidad del Diseño. Una forma de juzgar la calidad del diseño es observar su volatilidad a través del tiempo. Si un cambio debe ser hecho -por un cambio en los requerimientos o por un algún error o debilidad en el diseño existente- qué tan extenso es el cambio. La volatilidad del diseño puede ser bastante alta en sus primeras etapas hasta que gradualmente se va refinando.

Minimizar el Tamaño General del Sistema. Pequeños grupos pueden desarrollar sus diseños aún sin las técnicas formales porque el tamaño del sistema es mantenido dentro de ciertos límites. Cuando el proyecto es más grande, los equipos de trabajo también lo son pero siempre está el peligro de que el tamaño del problema exceda la capacidad del equipo. Así que mantenerlo tan pequeño como se pueda y aplicar un desarrollo más disciplinado.

Habilidad para Evaluar por Escenario. Otra forma de juzgar un diseño es realizar recorridos a través de él, esto provee una oportunidad para probar su corrección (que tan bien trabaja) y más importante aún, debe proveer una oportunidad para evaluar su calidad (si el equipo pudiera encontrar una forma de mejorarlo).

Evaluación por Factores Críticos de Exito. Hay ciertos factores que determinarán si el sistema es juzgado exitoso y por lo tanto aceptable: reutilización, claridad de lectura, desempeño, necesidad de tener cierta porción del diseño implantado y disponible al usuario pronto.

6. METODOLOGIA O.O. DE GRADY BOOCH

6.1 Definición de la Metodología Orientada al Objeto

Es un proceso creciente e iterativo, en el cual se identifican clases y objetos de acuerdo a las propiedades relevantes de un dominio en particular, con la finalidad de implantar un mecanismo que prevea el comportamiento que un modelo requiere.

El utilizar metodologías orientadas al objeto, es cambiar la forma tradicional de la visión de los elementos que componen un sistema o un modelo, ya que su abstracción y su construcción hará que el mecanismo empleado para su implantación haga de él un sistema iterativo y abierto a evolucionar.

La metodología O.O. propone que durante el proceso de diseño se retome mediante el análisis de sí mismo, haciendo una mejora basado en una nueva apreciación, este proceso debe repetirse hasta estar seguro que el diseño es correcto y completo, a este proceso se le conoce como Round-trip Gestalt Design.

6.2 Proceso de la Metodología O.O.

Round-trip Gestalt Design representa el proceso del diseño O.O. y requiere de dos elementos:

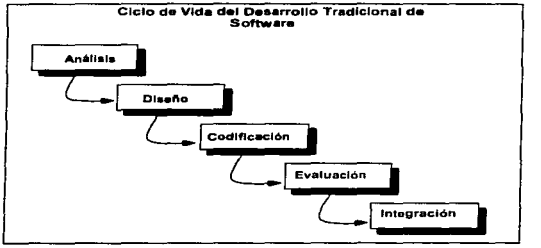
1. De una rica notación , que nos deje un modelo del problema en diferentes formas y que además permita enfocarse en las partes relativamente independientes del problema, en diferentes tiempos. Ya que el hablar de un proceso interactivo significa poder trabajar con cualquier tipo de modelo.

2. La invensión de algunos mecanismos que usen estas abstracciones, las cuales deberán ser documentadas en una serie de diagramas de objetos.

El diseño de los detalles de estos mecanismos nos lleva a la estructuración de clases, a través de las cuales se establece un protocolo, para cada una de ellas.

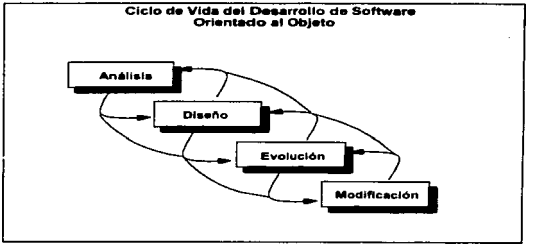
Eventualmente, al final del proceso del diseño, se termina con la definición de diagramas que proveen diferentes vistas del sistema, todos ellos deben ser coherentes y consistentes, teniendo una evolución creciente de uno a otro.

Este desarrollo creciente e interactivo hace del tradicional ciclo de vida en cascada del desarrollo de software un proceso de retroalimentación de una etapa a otra. El ciclo de vida tradicional es fundamentalmente un proceso pobre, y generalmente viola muchos de los principios de la práctica de la ingeniería, lo cual no quiere decir que el diseño debe ser realizado sin una estructura.



En su lugar, se sugiere que se permita un desarrollo evaluacionado, es decir, un proceso espiral.

Lourdes Montiel Garcia
Isabel Pérez Arredondo



El Diseño O.O. es un proceso que se obtiene mediante el seguimiento ordenado de los siguientes eventos:

1. Identificar las clases y objetos en un nivel de abstracción
2. Identificar la semántica de estas clases y objetos
3. Identificar las relaciones entre estas clases y objetos
4. Implantar estas clases y objetos

Este es un proceso creciente: la identificación de nuevas clases y objetos usualmente nos causa redefinir y mejorar la semántica de las relaciones entre clases y objetos. Es un proceso interactivo: la implantación de clases y objetos, lo cual regularmente nos lleva al descubrimiento o invensión de nuevas clases y objetos, que son una presencia de simplificación y generalización de nuestro diseño.

El inicio de nuestro proceso es descubriendo las clases y objetos que forman parte del vocabulario de nuestro dominio del problema, y nuestro objetivo final es identificar todas las entidades del dominio del problema, que tienen un papel en la aplicación, especificar la interacción entre entidades, la asociación de cada una de ellas, la asociación de cada entidad a la función que las realice, y el análisis de estas funciones a un nivel en

Lourdes Montiel García
Isabel Pérez Arredondo

que cada uno sea totalmente comprendido. El proceso termina cuando no encontramos más indicadores o mecanismos del sistema, o cuando las clases y objetos ya han sido descubiertas y que son suceptibles de implantarse por componentes de ellas mismas de software ya existente.

6.2.1 Identificación de Clases y Objetos

Se consideran dos pasos para realizar esta tarea:

1. El descubrimiento de las abstracciones primarias en un problema definido, es decir, los principales elementos que intervienen en el proceso, son cuales son candidatos a ser las clases y objetos.

2. La invención de los más importantes mecanismos que proveen el comportamiento requerido de los objetos que trabajan juntos para realizar una función. Para encontrar estos elementos y procesos es a través del análisis, usando técnicas para el modelado de objetos, durante este paso el desarrollador debe aprender el vocabulario del dominio del problema, las reglas del juego, y los eventos que deben ocurrir para las clases y objetos de nuestro diseño, en el más alto nivel de abstracción.

Aquí se enfatiza la definición de los principales elementos que intervienen en el sistema, ya que son éstos los candidatos a clases y objetos, y que además son la base del diseño.

Se inicia con la definición de lo más sobresaliente de las abstracciones más importantes para nosotros, esperando que estos sean los mecanismos que se implanten en el diseño.

Al finalizar esta etapa se debe obtener una lista de nombres que puedan ser clases y objetos (usando nombres completos para mantener su significado), por el momento sólo le llamaremos lista de cosas, ya que posteriormente éstas se convertirán en clases y

objetos y otros en simples atributos de objetos. Además de esta lista, se debe especificar el significado de estas abstracciones y mecanismos, para el llenado apropiado de las plantillas de las clases y objetos.

En la mayoria de los casos, este paso toma un corto tiempo, en considerando el tiempo que pueden llevarse los otros tres pasos.

Es recomendable se realice a esta lista una revisión, para definir un vocabulario común, para manejar entre los desarrolladores. Es también apropiado para un diseñador dibujar algunos diagramas de objetos que muestren cómo varias clases y objetos trabajan juntos. Si la descomposición física del sistema es importante, los dibujos de diagramas de módulos también deben ser creados en ese momento. Finalmente, si hay elementos de hardware para el sistema, también es de utilidad dibujar los diagramas de procesos necesarios.

6.2.1.1 Principios del Modelado de Objetos

En la identificación de objetos, propone una serie de principios para facilitar su detección y modelado, los cuales son:

1. Abstracción
2. Encapsulación
3. Modularidad
4. Jerarquía
5. Tipificación
6. Concurrencia
7. Persistencia

De los cuales identifica a los primeros cuatro como fundamentales, y los últimos tres como no esenciales pero sí muy útiles.

1. Abstracción

Una abstracción denota las características esenciales de un objeto que lo distinguen de los otros tipos de objetos. Se pueden identificar cuatro tipos de abstracción:

- Abstracción de entidades, es un objeto que representa un modelo usado por un problema en específico.
- Abstracción de acciones, es la identificación de un objeto que provee una función generalizada, es decir, todas al funciones que pueda realizar este objeto serán del mismo tipo.
- Abstracción virtual de máquina, es aquel objeto que lleva a cabo un conjunto de operaciones, las cuales son realizadas mediante un control de alto nivel, o que realiza operaciones que usan algunos controles de bajo nivel a éste.
- Abstracción de coincidencia, es aquel objeto que empaqueta un conjunto de operaciones definidos que no tienen relación entre ellas.

2. Encapsulación

Es el proceso que esconde todos los detalles de un objeto que no contribuye con sus características esenciales, es decir, es la información escondida en un objeto.

3. Modularidad

Es el acto de particionar un programa en componentes individuales, que pueden reducir su complejidad. La modularidad es una propiedad de un sistema que descompone en un conjunto de módulos definidos con cohesión y libertad entre ellos. La identificación de la modularidad de objetos, clases o comportamiento de los mismos, nos facilitarán el establecimiento del mecanismo de implantación del sistema.

4. Jerarquía

Es una hilera u ordenación de abstracciones. La mayoría de las abstracciones son identificadas en un orden jerárquico, en el diseño de un sistema la jerarquización de los objetos simplifica el entendimiento de un objeto. Se pueden identificar dos tipos de jerarquización: la sencilla y la múltiple; básicamente, la jerarquía define la relación entre clases, cuando entre varias clases comparten la misma estructura y comportamiento por una clase es sencilla, en caso de ser su comportamiento definido por varias es múltiple.

5. Tipificación

Un tipo es una característica precisa de estructura o comportamiento, y la cual un conjunto de entidades comparten. La tipificación es el coacción o ejecución de la clase de un objeto, lo cual hace que el objeto de diferentes tipos no cambie, es decir, sólo podrá hacerlo en casos especiales o restringidos.

6. Concurrencia

Es la propiedad que distingue un objeto activo de otro que no lo está, es decir, nos indica el estado de los objetos en su ciclo de vida.

7. Persistencia

Es la propiedad de un objeto de que a través de diferentes momentos y actividades existe o trasciende en el tiempo, es decir, un objeto puede continuar existiendo de un proceso a otro, ya sea por su propia existencia o a través de su cambio de dirección en el cual fué creado.

6.2.1.2 Propuestas para la identificación de Clases y Objetos

Históricamente se tienen tres propuestas para poder realizar la clasificación:

1. Categorización Clásica
2. Agrupación conceptual
3. Teoría del prototipo

1. Categorización Clásica

En la categorización clásica, todos las entidades que tienen una propiedad o colección de propiedades comunes forman una categoría. Dichas propiedades son necesarias y suficientes para definir la categoria.

Se pueden nombrar a las cosas, en este caso clases y objetos, de acuerdo al conocimiento de su naturaleza o propiedades y efectos.

2. Agrupación conceptual

En la agrupación conceptual es la más moderna forma de clasificación, e indica que primero se debe formular descripciones conceptuales de estas clases y objetos, y clasificarlas acorde a sus descripciones. Este proceso de clasificación representa más una forma probabilistica de agrupación conceptual.

3. Teoría del prototipo

Cuando se requiere identificar clases y objetos de un sistema complejo los criterios antes mencionados no son los adecuados a utilizarse, esto nos lleva a la Teoria del prototipo. Esta indica que hay algunas abstracciones que no son claramente definidas e identificables, por sus propiedades, ni por sus conceptos, por lo que hay que clasificarlos o separarlos por las propiedades que no son comunes entre estos objetos, a través de esta categorización se extienden los tipos de objetos que buscamos.

Una clase es representada por un objeto prototipo, y un objeto es considerado un miembro de esta clase si se asemeja a su prototipo es de manera significativa.

Conclusión: Se identifican clases y objetos, primero acorde a las propiedades relevantes de su dominio particular, en segundo término se considera el significado conceptual, y por último si ninguno de los dos anteriores captura el concepto requerido, se considera la clasificación por asociación.

También se pueden identificar clases y objetos siguiendo la siguiente clasificación:

1. Cosas tángibles: autos, sensores de presión, etc.
2. Roles: madre, maestro, policia, etc.
3. Eventos: buscar, interrumpir, preguntar, etc.
4. Interacciones: préstamos, juntas, intersecciones, etc.

6.2.2 Identificación de la semántica de Clases y Objetos

El segundo paso involucra al primero, ya que la estabilidad de éste depende del establecimiento de las clases y objetos identificados del paso anterior. Aquí el desarrollador es como un observador externo, es decir, su visión de cada clase debe ser desde la perspectiva de su interface, así como la identificación de las cosas que pueden ser una instancia de una clase y las cosas que cada objeto puede hacer con otro.

Este paso es mucho más difícil y requiere de más tiempo que el anterior. En esta fase se tienen fuertes debates durante las revisiones que se lleven por los diseñadores. Es en esta fase que se definen las clases y objetos, lo cual no representa una tarea fácil; decidir el protocolo para cada objeto resulta ser una tarea muy ardua. Por esta razón, el proceso del Diseño O.O. se convierte en interactivo desde este punto. La preparación del protocolo para un objeto, requerirá decisiones que cambien el significado de otro objeto. En general, la existencia de las llaves primarias o principales elementos no cambian, sino solamente se transfiere su alcance.

Una técnica útil para guiarnos en esta actividad, es la escritura de cada objeto, donde se defina su ciclo de vida, desde su creación hasta su destrucción, incluyendo las características de su comportamiento.

Se obtiene un diseño orientado al objeto, de crecimiento natural. Documentar nuestras decisiones de diseño respaldan el significado de cada clase y objeto, nosotros generalmente refinamos la plantilla que dibujamos anteriormente. Esto significa que debemos documentar toda la semántica estática y dinámica de cada llave primaria y mecanismo, de la mejor forma posible. También se debe dibujar nuevos diagramas que documenten el mecanismo que hemos inventado en este paso. Por último, se debe decidir cómo se integrarán las partes del diseño, es decir se debe crear el prototipo del sistema; para hacerlo se debe considerar nuestro actual diseño y evaluar las alternativas de desarrollo y los problemas que puedan representar un riesgo.

6.2.2.1 Identificando cuáles son clases y cuáles son objetos

Ubicar clases y objetos en el nivel adecuado de abstracción es difícil, algunas ocasiones encontramos en la dificultad de definir cuál es una clase y cuál no, lo que debemos hacer es identificar la cohesión y libertad entre sus abstracciones, y con ello colocarla por la jerarquía que le corresponda por los objetos que puedan crearse a partir de ella.

La identificación de las principales abstracciones involucra dos procesos: el descubrimiento y la invensión. A través del descubrimiento, nosotros reconocemos las abstracciones usadas en el dominio del problema (regularmente el experto del problema lo menciona), y por medio de la invensión se crean nuevas clases y objetos que son necesariamente parte del problema.

6.2.2.2 Identificando mecanismos

Usamos el término de mecanismo, para describir cualquier estructura de donde los objetos trabajan juntos para proveer algún comportamiento que satisfaga los requerimientos del problema.

Un mecanismo representa el diseño de decisión acerca de cómo una colección de objetos que deben cooperar.

El mecanismo es como el alma del diseño, y representa la estrategia de decisiones que deben los diferentes objetos realizar cooperativamente para cumplir su función.

6.2.3 Identificación de relaciones entre Clases y Objetos

Este paso es de mayor número de actividades que el anterior. Aquí, estableceremos exactamente cómo las cosas deben interactuar en el sistema. Respecto a las llaves primarias o elementos principales, se deben afirmar el uso, herencia y otros tipos de relaciones entre clases. Para los objetos en nuestra implantación, deben establecerse la semántica dinámica y estática de cada mecanismo.

Hay dos actividades relacionadas aquí que nos causan refinar la productos de nuestro diseño. Primero, nosotros debemos descubrir patrones entre clases, los que nos provoca la reorganización y simplificación de la estructura del sistema de clases, y patrones entre la colección cooperativa de objetos, esto nos lleva a generalizar los mecanismos que comprenden nuestro diseño. Esta parte del proceso del diseño llama todas las aptitudes creativas del diseñador. En segundo lugar, debemos tomar decisiones de visión, es decir, cómo una clase ve a otra, cómo los objetos ven a otros, y de igual importancia, qué objetos deben ver o no a otros. Estas decisiones nos ayudan a tomar decisiones de empaquetamiento inteligentes en el diseño de la arquitectura de módulos de nuestro sistema.

El descubrimiento de patrones y tomar decisiones de visión deben causar la refinación de protocolos de varias clases, definidas anteriormente, lo cual nos lleva a abstracciones comunes y mecanismos definidos en un solo lugar.

Una técnica que es útil para guiarnos en estas actividades es el uso de tarjetas CRC, definidas por Beck an Cunningham. Una tarjeta CRC (para clases, reponsabilidades y colaboración) es usada para cada clase de objetos, en las cuales cada una contiene el nombre de la clase, sus responsabilidades (su comportamiento), y las clases que colaboran con ella. Un diseñador puede que organice el espacio de las tarjetas y refina su contenido, de acuerdo a la ejecución de escritos o del ciclo de vida de los objetos, a manera de introducir un paso previo.

En este punto del diseño aún no podemos tener una vista completa de las llaves primarias o elementos importantes y de los mecanismos existentes, ya que todavía es prematuro, porque hasta el momento sólo se ha considerado la representación de clases y objetos en nuestro sistema, y porque todavía no conocemos lo suficiente de cada abstracción y mecanismos que usaremos.

Este paso incluye el completar la mayoría de los modelos del diseño. Se refinan los diagramas de clases hechos anteriormente, tomando en cuenta los descubrimientos de patrones y la visibilidad que tiene un objeto del otro. También se completa los detalles de los diagramas de objetos que documentan la esencia del mecanismo de nuestra solución. En la práctica primero se dibujan los objetos que conocemos que cooperan en cierta forma, después, se seleccionan pares de objetos y se pregunta si hay relaciones entre ellos, si la respuesta es afirmativa, se realizan las siguientes dos preguntas: cómo estos objetos están relacionados y que mensajes se envían el uno al otro. Lo cual no llevará a refinar el protocolo de las correspondientes clases, y este refinamiento debe causarnos el descubrimiento de nuevos patrones. Este es otro de los motivos por el que la metodología O.O. es un proceso iteractivo.

Hasta este punto hemos creado la esencia del diagrama de módulos para nuestra solución, la cual contempla las decisiones de visibilidad tomadas. También hemos creado y refinado prototipos

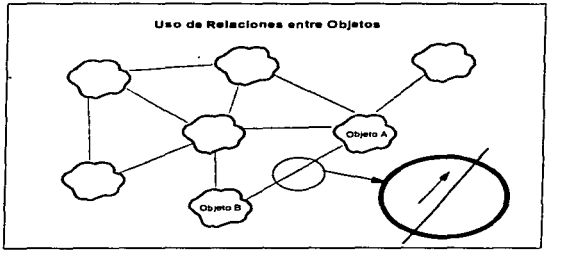
6.2.3.1 Relaciones entre Clases y Objetos

Tipos de Relaciones.

Los objetos contribuyen al comportamiento de un sistema con la colaboración de otros objetos. Las relaciones entre dos abarca la suposición de que cada uno hace para el otro, lo cual incluye las operaciones que pueden ser realizadas y el comportamiento resultante. Existen dos tipos de jerarquías de relaciones entre objetos:

1. Uso de relaciones
2. Contenido de relaciones

1. Uso de relaciones



Lourdes Montiel García
Isabel Pérez Arredondo

Esta figura ilustra cómo se usan las relaciones entre varios objetos; la línea entre dos objetos representa la existencia del uso de una relación entre dos y significa que un mensaje pasa a través de este camino (relación). El mensaje que pasa entre dos objetos es usualmente unidireccional, sin embargo es posible la comunicación entre objetos bidireccionalmente.

Dada una colección de objetos que están relacionados, cada uno puede jugar 3 diferentes roles:

- Objeto Actor, un objeto que puede operar sobre otros objetos, pero que nunca es operado por otros objetos.
- Objeto Servidor, es aquel objeto que nunca opera sobre otros objetos, solamente es operado por otros objetos.
- Objeto Agente, es aquel objeto que puede operar sobre otros objetos y que puede ser operado por otros objetos.

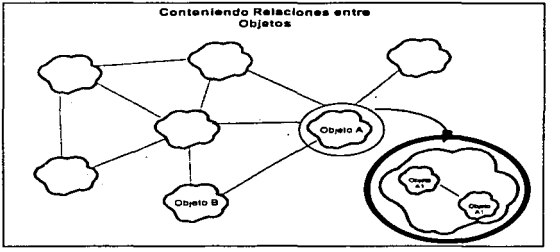
Cuando un objeto pasa un mensaje a otro deben estar sincronizados de alguna forma. Lo cual nos lleva a que un objeto puede ser utilizado como:

- Objetos secuenciales, es un objeto pasivo cuya semántica está garantizada solamente en la presencia de una secuencia de control.
- Objetos por bloques, es un objeto pasivo cuya semántica está garantizada en presencia de múltiples secuencias de control.
- Objetos concurrentes o activos, es un objeto activo cuya semántica está garantizada en la presencia de múltiples secuencias de control.

El que un objeto sea utilizado secuencialmente o por medio de una sola secuencia de control excluye las otras formas, y así con cada uno.

Entiéndase por objeto activo aquél que comprende su propia secuencia de control; y por objeto pasivo es aquél que no comprende su propia secuencia de control.

2. Contenido de relaciones



Es la encapsulación de una relación en un objeto. El contener o encapsular una relación reduce el número de objetos que deben ser visibles a un nivel de objetos.

Relaciones entre Clases

Un objeto es una entidad concreta que existe en un tiempo y espacio, una clase representa solamente una abstracción, la esencia de un objeto.

Una clase es como un grupo definido por atributos comunes, es una definición de objetos que comparten una estructura común y un común comportamiento.

La interface de una clase proporciona la vista exterior de sí misma y por tanto enfatiza la abstracción mientras esconde en su estructura los secretos de su comportamiento. Esta interface primeramente consiste de las declaraciones de todas las

Lourdes Montuel García
Isabel Pérez Arredondo

operaciones aplicables a instancias de esta clase, esto podría incluir la declaración de otras clases, constantes y variables.

Una relación entre clases indican un tipo de compartición y un tipo conexión semántica. Hay tres tipos de relaciones entre clases:

1. La generalización, denotando "un tipo" de relación.
2. La agregación, la cual denota "una parte" de relación.
3. La asociación, la cual denota alguna conexión semántica entre otro tipo de clase no relacionada.

Estas relaciones son soportadas por los lenguajes O.O. o los basados en objetos; en la combinación de las relaciones entre las clases denominadas:

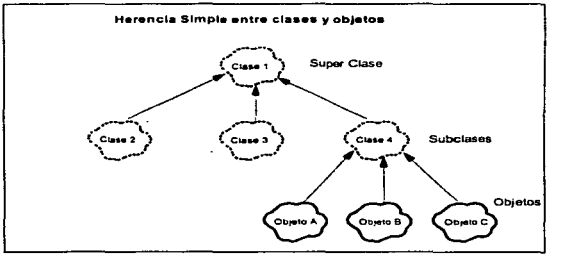
1. Herencia de relaciones, es cuando una clase comparte la estructura o comportamiento definidos en una herencia simple o múltiple (uno o varios objetos respectivamente).

   La herencia define un tipo de jerarquía, donde cada clase hereda de otra u otras clases (la de nivel más alto se le denomina superclase y las clases de nivel inferior son llamadas subclases). Una subclase tipicamente redefine la estructura definida y el comportamiento de su superclase.

   La herencia de relaciones es la base del polimorfismo, ya que por las diferentes relaciones con varias clases un objeto tiene la posibilidad de responder a una operación en diferentes formas.

   La herencia múltiple se usa cuando la relación simple no satisface las suficientes relaciones entre clases. Su uso es parte de la existencia de relaciones de un objeto con más de una clase, debido a su naturaleza y las relaciones requeridas. Cuando una clase es construida primeramente por

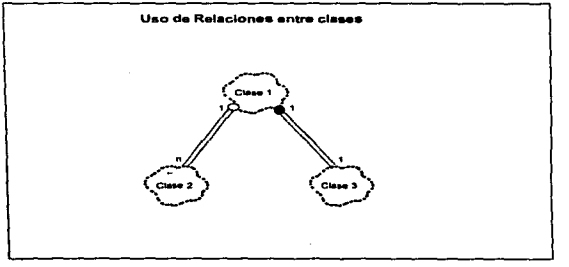
herencia múltiple y no se le aumenta su propia estructura o comportamiento se está hablando de una clase agregada.



2. Uso de relaciones, hay dos tipos de uso de relaciones entre clases:
   A. Interface de una clase, (uso de otra clase) en donde la clase usada debe ser visible a cualquiera de sus clientes.
   B. Implantación de una clase, (uso de otra clase) en donde la clase usada es escondida como parte del secreto de usar clases.

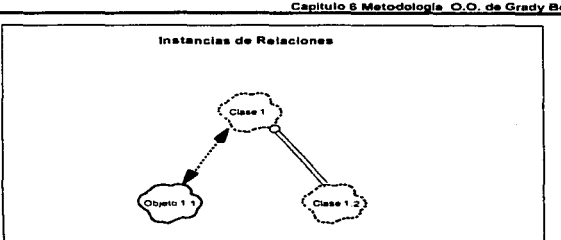
   A través de las lineas que representa la relación se puede expresar tambien la cardinalidad de la relación.

Lourdes Montiel Garcia
Isabel Pérez Arredondo



3. Instancias de relaciones, en la definición de una clase debe tener un estado preciso de los objetos que pueden pertenecer o ser definidos como instancias de una clase. Una definición es un ejemplo de un depósito de una clase cuyas instancias son colecciones de otros objetos. Un depósito de una clase denotan una colección homogénea, es decir, todos los objetos son de la misma clase, o que representan un conjunto heterogéneo, en donde cada objeto debe ser de diferentes clases, pero que comparten algunas superclases comunes.

Lourdes Montiel Garcia
Isabel Pérez Arredondo



4. Relaciones de metaclases, una metaclase es una clase, la cual sus instancias son la misma clase, es decir la clase de una clase es la metaclase.

6.2.4 Implantación de Clases y Objetos

Este paso no es necesariamente el último, Se incluyen dos principales actividades: tomar decisiones de diseño concernientes a la representación de las clases y que hemos inventado y ubicar las clases y los objetos en los módulos, y programas en procesos. Es en este punto cuando a partir de la vista de adentro de cada clase y módulo, se decide como su comportamiento se debe implantar.

Al menos las principales abstracciones y mecanismos son triviales, éste es el punto donde regularmente se regresa al primer punto para aplicar nuevamente este proceso, pero ahora con las clases y módulos existentes. En esta forma, repetimos el proceso de diseño, sin embargo en esta ocasión nos enfocamos en un nivel de abstracción menor. Lo cual no quiere decir que la metodología O.O. es un procesos TOP-DOWN exactamente, porque diseñamos primero el nivel más alto de las principales abstraciones y

mecanismos, lo cual involucra estas clases y objetos que son directamente parte del vocabulario de dominio del problema, sin embargo, en cualquier nivel del diseño, nosotros nos saltamos el nivel más bajo de abstracción de las principales abstracciones y de los mecanismos, porque ellos nos sirven más como clases y objetos primarios sobre los cuales se componen todos los altos niveles de clases y objetos. Esto nuevamente hace de la metodología O.O un proceso Round-trip gestalt.

En este paso se establece la interface concreta para cada clase y objeto importantes para nosotros en un nivel de abstracción. Nuevamente se realiza una refinación de la estructura de clases del sistema, y en particular el completar la implementación para cada parte de las plantillas de clases, de los diagramas de los módulos, y si el sistema lo demanda de los diagramas de procesos.

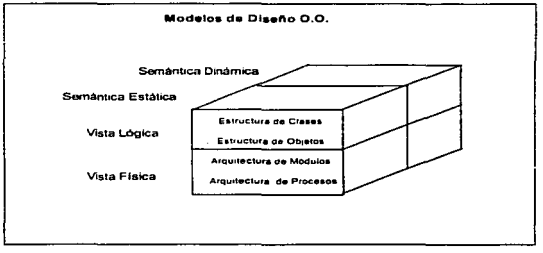
La implantación de una clase en su vista interior involucra el secreto del comportamiento de la clase. La implantación de una clase se divide en tres partes:

1. Público, la declaración que forma parte de la interface de una clase y es visible a todos los clientes que son visibles a la clase.
2. Protegido, la declaración que forma parte de la interface de una clase, pero no es visible a cualquier otro, excepto de sus subclases.
3. Privada, la declaración que forma parte de la interface de una clase, pero no es visible por cualquier otra clase.

Lourdes Montiel Garcia
Isabel Perez Arredondo

6.3 La Notación

La notación sugerida por Grady Booch, contempla la estructura de las clases, los mecanismos de herencia, el comportamiento individual de los objetos y el comportamiento dinámico de un sistema.



La figura anterior muestra los diferentes modelos que se tienen en el diseño orientado al objeto, según la metodología de Grady Booch, los cuales expresan y capturan todos los elementos del análisis y diseño orientado al objeto de un sistema.

La notación es la documentación del diseño de un sistema, esta notación debe usarse de acuerdo a las necesidades de cada proyecto. Tenemos cuatro tipos de diagramas:

1. Diagramas de clases
2. Diagramas de objetos
3. Diagramas de módulos
4. Diagramas de Procesos.

Lourdes Montiel García
Isabel Pérez Arredondo

Los primeros dos son parte de la vista lógica de un sistema, porque nos describen la existencia y el significado de las principales abstracciones y la forma del diseño. Los últimos dos diagramas son parte de la estructura física de un sistema, porque describen concretamente los componentes del software y hardware de una implantación.

La semántica Dinámica y Estática de un sistema se expresan a través de dos tipos de diagramas:

1. Diagramas de estados de transición
2. Diagramas de sincronización (Timing Diagrams)

Para cada clase deberá tener un diagrama de transición asociado a sus indicaciones de cómo el orden en qué eventos externos pueden afectar el estado de cada instancia de la clase.

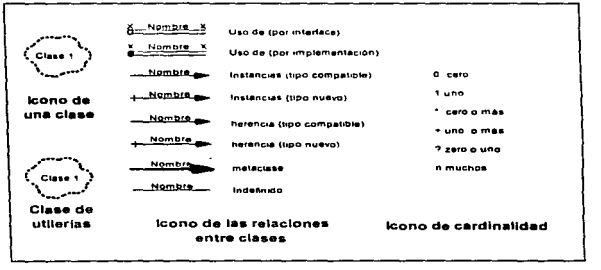
Se deberá usar diagramas de sincronización en conjunción con cada diagrama de objetos, para mostrar el orden de los mensajes que son enviados y evaluados.

6.3.1 Diagramas de clases

Muestran la existencia de clases y sus relaciones en el diseño lógico de un sistema. Un diagrama de clases representa todo o parte de la estructura de clases del sistema. Hay tres elementos de la estructura de clases: *clases, las relaciones de las clases y las utilerías de clases*.

Clases, su icono es un globo amorfo, llamado nube y está pintado con linea punteada, la cual indica que los clientes que hacen uso de él generalmente operan solamente sobre las instancias de la clase y no de la clase en si. La clase debe tener nombre, el cual es escrito dentro de la nube. Si el nombre es particularmente largo, se puede usar nemónicos, o en su defecto agrandar la nube. Cada clase debe ser única.

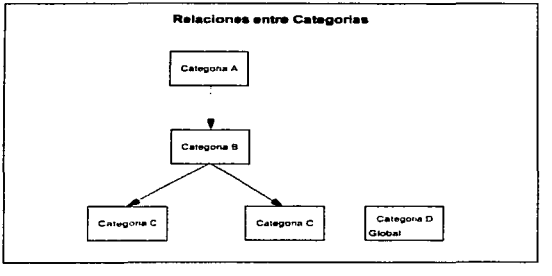
Relaciones de las Clases, los iconos para los diagramas de clases están especificados en el siguiente cuadro, cada relación debe tener etiqueta, la cual documenta el nombre de la relación entre las clases. El uso de una relación con doble línea con un círculo en uno de los extremos identifica que la clase usa recursos de otra, se dice que la relación es por interface cuando la clase utilizada es visible a todo cliente, y la relación es por implementación cuando la clase está oculta como parte del secreto del uso de una clase. El simbolo • es usado para indicar la dirección que la etiqueta debe ser leida. De igual forma los iconos para representar la cardinalidad en las relaciones entre clases se muestran en el mismo cuadro. También en casos complicados y donde se requiera, para completar la cardinalidad se pueden usar operadores relacionales: =, <>, >, <, <= y >=.



Utilerías de Clase, representa cualquier subprograma libre o una colección de programas libres, es decir que no pertenecen a una clase, y que a su vez pueden servir a aquéllas que lo requieran. Su icono es también una nube, pero además tiene una sombra, y también necesitan tener un nombre.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Categorías de Clases y su visibilidad, es una estructura de clases grande, es una forma de organizar una gran cantidad de clases, en donde cada categoría representa una colección de clases o de otras categorías de clases. Este concepto se aplica en sistemas muy grandes. Este tipo de diagramas permiten al desarrollador entender la arquitectura lógica general de un sistema, porque muestra el nivel más alto de las abstraciones que forman el vocabulario del dominio del problema. El icono para representar una categoría es un rectángulo con el nombre de la categoría dentro. Cada categoría tiene un diagrama de clases. Las relaciones entre clases son representadas con una línea. Cuando el nombre de una categoría está subrayado, quiere decir que contiene clases privadas y que otras pudieran ser importadas de otra categoría visible; cuando sólo está el nombre de la clase, representa una entidad que es privada; cuando el nombre de la categoría está en un rectángulo, representa una entidad que es contenida en una categoría de clases, pero que es exportada y por tanto visible a las demás categorías que la importan. Una librería de clases es representada como una categoría. Cuando una categoría es vista por la mayor parte de categorías es válido identificarla como global (esta palabra se pone en la parte inferior izquierda de la categoría global).



Lourdes Montiel García
Isabel Pérez Arredondo

Plantillas de Clases, es la documentación de cada clase, subclase, sus campos, y sus operaciones.

Nombre:	Identificador
Documentación:	Descripción de lo que se documenta
Visibilidad :	Exportada, Privada, Importada
Cardinalidad:	0 / 1 / n
Jerarquía:	
Superclases:	Lista de clases
Metaclases:	nombre de la clase
Parámetros:	Lista de parámetros
Interface/Implementación	
Público o Privada:	
Usos:	Lista de clases
Campos:	Lista de declaraciones de campos
Operaciones:	Lista de operaciones
Estado:	Diagrama de estados de transición
Concurrencia:	secuencial, por bloques, activo
Espacio de complejidad:	texto
Persistencia:	estático o dinámico

La visibilidad, indica si la clase es exportada, privada o importada en relación a su categoria de clases; la cardinalidad, indica cuántas instancias permitirá la clase; la jerarquía dependerá del lenguaje que se utilice para su desarrollo y del diseño del mismo, lo mismo es con los parámetros ya que

no todos los lenguajes lo permiten; la implementación o interface se refiere a la forma en que los diferentes atributos van a ser definidos, lo cual también dependerá del lenguaje que se utilice ya que no todos tienen la versatilidad del manejo de diferentes tipos; los campos deben indicarse áquellos que son propios de la clase con sus definiciones; las operaciones es una descripción general de las operaciones de la clase; usos, se refiere a las clases de las cuáles hace uso; el espacio de complejidad, se refiere al tamaño por unidad de la clase; por último la persistencia se refiere a que si la definición de la clase es estática o dinámica.

Nombre:	Identificador
Documentación:	Descripción de lo que se documenta
Visibilidad:	Exportada, Privada, Importada
Cardinalidad:	0 / 1 / n
Parámetros:	Lista de parámetros
Interface/Implementación	
Público o Privada:	
Usos:	Lista de clases
Campos:	Lista de declaraciones de campos
Operaciones:	Lista de operaciones

Plantilla de una Clase de Utilerías

Lourdes Montiel García
Isabel Perez Arredondo

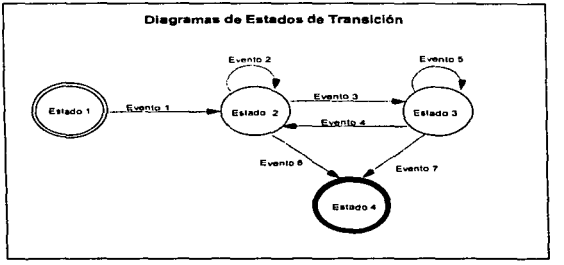
Para un diseño no tan riguroso la plantilla de operaciones incluirá el nombre de la operación, sus parámetros y su significado o descripción; sin embargo, si se requiere de mayor documentación se recomienda hacer uso de la siguiente plantilla, de la cual se podrá optar por los elementos requeridos.

Nombre:	Identificador
Documentación:	Descripción de lo que se documenta
Categoria:	texto
Requisitos	texto
Parámetros:	Lista de parámetros
Resultado:	Nombre de la clase
Precondiciones:	PDL / diagramas de objetos
Acciones:	PDL / diagramas de objetos
Postcondiciones:	PDL / diagramas de objetos
Excepciones	Lista de excepciones.
Concurrencia	Secuencial/protegido, concurrente/multiple
Tiempo de complejidad	tiempo de ocurrencia
Espacio de complejidad	alcance de la operación

Plantilla de una Operación

La categoría provee una forma de agrupar lógicamente operaciones relacionadas (con base en su clase). Los Requisitos dependen del lenguaje que se utilice para implementar, e indica el momento en que se deberá aplicar la operación: antes, despues o durante. Las precondiciones, acciones y precondiciones hacen referencia a diagramas de objetos, si se requiere mayor descripción de la operación se hace uso de ellas. Los últimos tres elementos documentan la concurrencia, la semántica y el espacio de complejidad de cada operación.

Diagramas de estados de transición, estos diagramas permiten mostrar el estado en el espacio de una clase, es decir, los eventos que causan una transición de un estado a otro, y las acciones descritas en un diagrama de estados nos podría llevar, si se requiere, a un diagrama de objetos. Un círculo representa un estado, y en él debe estar escrito el nombre (único) del estado, un estado de inicio, es representado con un doble círculo sin llenar, y un estado de término se representa con un círculo rellenado.



Evento:	Lista de identificadores
Documentación:	Descripción
Acción:	PDL/ diagrama de objectos

Plantilla de un Diagrama de Estados de Transición.

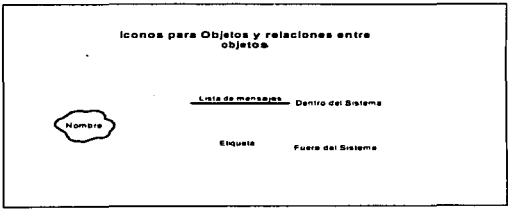
Lourdes Montiel García
Isabel Pérez Arredondo

6.3.2 Diagramas de objetos

Un diagrama de objetos representa todo o parte de la estructura de un sistema. El objetivo de cada diagrama de objetos es ilustrar la semántica de los principales mecanismos en el diseño lógico. Cada diagrama representa las interacciones que deben ocurrir entre una colección de objetos. Las operaciones usadas en un diagrama de objetos deben ser consistentes con las operaciones definidas en las clases asociadas, es decir, si se muestra que un objeto envia a otro un mensaje, entonces la operación a realizar al recibir el mensaje debe estar defindo por su clase.

Objetos, el icono para representar un objeto es semejante al de las clases, el nombre de los objetos debe ser único. Las propiedades de un objeto deben aparecer en el lado izquierdo inferior de su icono. Las más importantes propiedades que incluyen son la concurrencia y la persistencia de cada objeto, en otras palabras si el objeto es inactivo o activo durante un proceso.

Relaciones entre objetos, representan el envio de mensajes de un objeto a otro. Cuando usamos una línea, se trata de un mensaje bidireccional; cuando usamos líneas sólidas es cuando el modelo podemos implantarlo en el software del sistema, y cuando son grises son que no pueden ser implementadas en el software.



Lourdes Montiel García
Isabel Pérez Arredondo

Objeto	
Nombre:	Identificador
Documentación:	Descripción
Clase:	Nombre de la clase a la que pertenece
Persistencia:	estático / dinámico

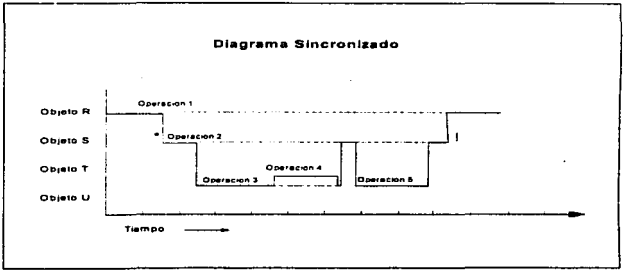
Plantilla de un Objeto.

Mensajes	
Operación	Nombre de la operación
Documentación:	Descripción
Frecuencia:	periodo o frecuencia de ejecución
Sincronización:	Simple síncrona, asíncrona.

Plantilla de las relaciones entre Objetos.

Lourdes Montiel García
Isabel Perez Arredondo

Diagramas Sincronizados, se usan para representar la dinámica del envío de mensajes de un diagrama de objetos. Un diagrama sincronizado es una gráfica que establece el tiempo en el eje de horizontal, y los objetos a través del eje vertical. Como se avance sobre el eje horizontal, el tiempo, una operación debe ser invocada. Al marcar un proceso con un asterisco, indica que se crea el objeto, y cuando se marca el proceso con un signo de admiración (!) indica la destrucción del objeto.



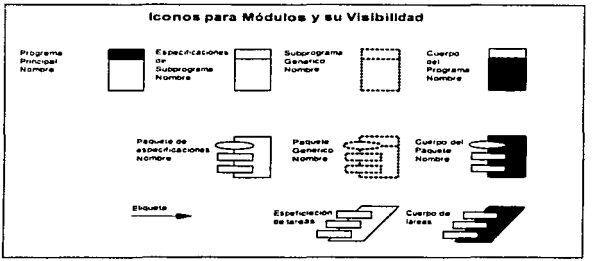
Lourdes Montiel García
Isabel Pérez Arredondo

6.3.3 Diagramas de módulos

(Boochgrams / Gradygrams)

Muestran la colocación de las clases y objetos en los módulos en el diseño físico de un sistema; un diagrama de módulos representa todo a parte de la arquitectura del sistema. Hay dos elementos en la arquitectura de módulos: los módulos y su visibilidad.

Existen varios iconos para representar los diferentes tipos de módulos que pueden existir. Los tipos de módulos corresponden a tipos de módulos que soporten los diferentes lenguajes orientados o basados en objetos.



Módulos, es la clasificación de los programas que componen un sistema. El nombre del módulo es necesario, y se debe colocar en la parte superior del rectángulo, cada nombre debe ser único, cuando el nombre del módulo está dentro de un rectángulo, indica que ese módulo es exportado por otro subsistema; en caso contrario, es decir, que sólo sea un nombre representa un módulo privado, que sólo está incluido en ese sistema; y cuando el nombre está subrayado, representa un módulo que es importado de otro sistema. La única

Lourdes Montiel García
Isabel Perez Arredondo

relación entre módulos es la dependencia de compilación, la cual es representada por la linea representada en la figura anterior.

Subsistemas, en los sistemas grandes es necesario contar con un nivel de abstracción mayor, este nivel es representado mediante los subsistemas, a través de estos diagramas un desarrollador puede comprender de forma general la arquitectura de un sistema. Cada subsistema denota otro diagrama de módulos, con su diagrama de categorías de clases, etc. El Icono de un subsistema se muestra en la siguiente figura.

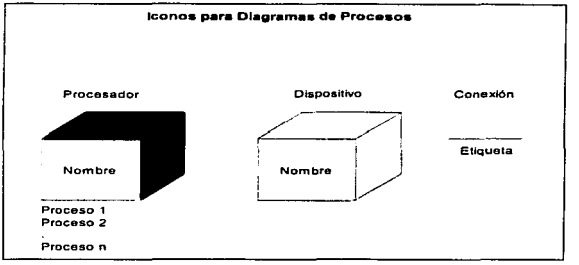
Nombre:	Nombre del Subsistema
Documentación:	Descripción
Declaraciones	Lista de declaraciones del subsistema

Plantilla de Subsistemas.

6.3.4 Diagramas de Procesos

Un sistema contiene múltiples programas ejecutándose sobre una colección de computadoras distribuidas. Los procesos de diagramas visualizan y nos ayudan a ubicar procesos y procesadores.

Un diagrama de procesos representa todo o parte de la arquitectura de procesos de un sistema. Hay tres elementos importantes para los diagramas de procesos: procesadores, dispositivos y conexiones.



Procesadores y dispositivos, es una pieza de hardware capaz de ejecutar programas, regularmente en el icono de procesodores incluye la serie de procesos que va a ejecutar.

Conexiones, representa un acoplamiento de hardware, como lo es un cable Ethernet, o fibra óptica. Esta es representada por la línea mostrada en la figura anterior, de acuerdo a la dirección requerida se puede poner una flecha.

Procesador	
Nombre:	Nombre del Procesador
Documentación:	Descripción
Características:	Características generales
Procesos:	Lista de procesos que realizarla
Horario:	Ciclos de ejecución de los procesos

Plantilla de Procesadores.

Lourdes Montiel García
Isabel Pérez Arredondo

Procesos	
Nombre:	Nombre del proceso.
Documentación:	Descripción.
Prioridad:	Grado de prioridad del proceso (numérico).

Plantilla de Procesadores.

Dispositivos y Conexiones	
Nombre:	Nombre del proceso.
Documentación:	Descripción.
Características:	Descripción.

Plantilla de Dispositivos y Conexiones.

Lourdes Montiel García
Isabel Pérez Arredondo

7. APLICACION

Planteamiento.

Para la aplicación de las tres metodologías hasta el momento planteadas, se realizará el análisis y diseño de un sistema de personal docente de una escuela que imparte clases del nivel medio superior.

A partir del requerimiento de un sistema real para la aplicación de estas metodologías seleccionamos este sistema, ya que por la colaboración que se tuvo en el desarrollo del mismo, contábamos con los requisitos del problema delimitado.

Por otro lado las metodologías antes descritas cuentas con herramientas para su aplicación en cualquier ámbito, ya que se refieren a desarrollo de Sistemas de Información en áreas diversas, como: editores de gráficos, un sistema analógico de un invernadero, el sistema de control de tráfico aéreo, sólo por mencionar algunos; por lo tanto el ámbito académico no se excluye para su aplicación.

La aplicación de las Metodologías Orientadas al Objeto serían rentables debido a que la institución es de tamaño considerable y que cuenta con tecnología de vanguardia disponible para su área de desarrollo de sistemas, además de las aportaciones de los componentes que se generarían en el análisis y diseño para su futura aplicación. Considerando este último punto el más importante, ya que se tiene un grupo de personas para desarrollar varios sistemas todos relacionados y que darán a la institución la posibilidad de crear un Sistema Integral, el cual utilizaría los elementos ya elaborados.

El sistema debe cumplir con el siguiente objetivo:

Llevar el registro de la información relacionada con el reclutamiento, selección, contratación, reclasificación y bajas docentes .

Lourdes Montiel García
Isabel Pérez Arredondo

Considérese la siguiente lista de requerimientos para el análisis y el diseño del mismo:

1. Reclutamiento y selección de aspirantes

1.1. Requisición de Personal Docente, el sistema permitirá capturar los datos referentes al personal docente que requiera para impartir clases (los datos que en ésta se ponen son los siguientes: ciclo escolar, coordinador de área que la genera, área académica y asignatura para la cual solicita al profesor, horario solicitado y observaciones).

1.2. Selección inicial de curriculum, el personal del Departamento de Personal Docente seleccionará la curricula que a su criterio tengan posibilidades de incorporarse al personal docente

1.3. Alta de aspirantes y curricula (seleccionada), el sistema permitirá la captura de los datos generales de los aspirantes a profesor.

1.4. Verificación de aspirantes y profesores, por medio del Registro Federal de Contribuyentes, el sistema verificará automáticamente la existencia de los mismos para evitar la asignación de dos claves a una sola persona.

1.5. Verificación de referencias laborales, el sistema llevará registro de las referencias, resultados y observaciones.

1.6. Registro de los aspectos evaluados durante la entrevista profunda, la calificación obtenida para cada uno tendrá la siguiente escala de calificación: 0 al 4 y observaciones.

1.7. Recepción de documentación, se llevará registro de cada uno de los documentos que entregue el aspirante, con su número de copias.

1.8. Evaluación de entrevista técnica, se permitirá el registro de la evaluación de cada aspirante.

1.9. Evaluación de Clase Modelo, se llevará el registro de las evaluaciones por aspectos: apariencia personal, seguridad en sí mismo, presentación del tema, dominio del tema, desarrollo del tema, capacidad para ejemplificar, facilidad para responder preguntas, uso adecuado del pizarrón, uso de otros materiales, voz, uso de la comunicación corporal, dominio del espacio; siendo su escala de calificación : del 0 al 4 y observaciones.

2. Contratación y bajas de profesores

2.1. Control de documentación de profesores, el sistema permitirá la actualización del expediente por profesor, en cuanto a su experiencia en docencia, laboral y académica.

2.2. Alta y asignación de clave de profesor asignado, el sistema permitirá el cambio de un aspirante a profesor, asignándole su clave de profesor automáticamente.

2.3. Emisión de Credenciales, el sistema permitirá la impresión de credenciales al personal docente, con la información correspondiente. Una vez que se dé el evento de contratación.

2.4. Emisión de Contratos, el sistema emitirá los contratos del personal docente. La emisión será cuando se dé el evento de contratación.

2.5. Control de categoría, el sistema además de los datos generales determinará la categoría, a partir del grado académico, de cada uno de los profesores.

2.6. Bajas (temporales y definitivas) y reingresos, el sistema permitirá la definición de bajas temporales y definitivas así como los reingresos de profesores.

2.7. Control de la antigüedad de profesores, el sistema calculará la antigüedad por profesor, a partir de la fecha de ingreso como profesor.

3. Información residente en la base de datos para su explotación estadística

3.1. Consultas y reportes diversos en función de los siguientes datos:

TABLA DE REQUERIMIENTOS

Area de Información	*Dato*	*Descripción*	*Valores Posibles*	*Fuente de obtención*
Información personal	Fecha de nacimiento	1. Fecha de nacimiento del profesor	DD/MM/AA	Manual
	Nacionalidad	2. Pais de origen del Profesor	Parametrico	Tabla
	Sexo	3. Sexo del Profesor	Femenino, Masculino	Clave

	Zona donde habita	4. Zona de la ciudad o del país.	Delegación o Municipio.	Tabla
	Disponibilidad de Horario	5. Horario en que puede dar clases	Paramétrico	Tabla
Formación Profesional	Licenciatura	6. Area de formación académica 7. Tipo de Institución 8. Escuela de procedencia 9. Periodo de estudios 10. Obtención de Título 11. Cédula profesional	Paramétrico Paramétrico Paramétrico mes-Año Si, Pasante, En trámite. 0-999999	Tabla Tabla Tabla Manual Manual
	Posgrado Especialidad	12. Area de formación académica 13. Tipo de Institución 14. Escuela de procedencia 15. Periodo de estudios 16. Obtención de Título	Paramétrico Paramétrico Paramétrico mes - Año Si, Pasante, En trámite, Cursando	Tabla Tabla Tabla Manual Manual
	Posgrado Maestría	17. Area de formacion academica 18. Tipo de Institución 19. Escuela de procedencia 20. Periodo de estudios 21. Obtención de Título	Paramétrico Paramétrico Paramétrico mes - Año Si, Pasante, En trámite, Cursando	Tabla Tabla Tabla Manual Manual

Lourdes Montiel García
Isabel Pérez Arredondo

Posgrado Doctorado	22. Area de formación académica 23. Tipo de Institución 24. Escuela de procedencia 25. Periodo de estudios 26. Obtención de Titulo	Paramétrico Paramétrico Paramétrico mes - Año Si, Pasante, En trámite, Cursando	Tabla Tabla Tabla Manual Manual
Estudios Complementarios	27. Area de formación académica 28. Tipo de Institución 29. Escuela de procedencia 30. Periodo de estudios 31. Obtención de Constancia	Paramétrico Paramétrico Paramétrico mes - Año Si o no	Tabla Tabla Tabla Manual Manual
Experiencia profesional	32. Tipo de Empresa 33. Nombre de la Empresa 34. Domicilio de la Empresa 35. Puesto desempeñado 36. Nombre del Jefe inmediato 37. Periodo laboral	Propia, Pública, Privada Variable Variable Variable mes - Año Variable	Manual Manual Manual Manual Manual Manual

Lourdes Montiel García
Isabel Pérez Arredondo

	Proyección profesional	38. Distinciones recibidas,	Paramétrico	Manual
		39. Organismo que otorga la distinción	Paramétrico	Manual
		40. Fecha de la distinción	mes - Año	Manual
		41. Publicaciones	Paramétrico	Manual
		42. Fecha de la publicación	mes - Año	Manual
		43. Trabajos de investigación	Paramétrico	Manual
		44. Indicación de investigación individual o en grupo	Paramétrico	Manual
		45. Fecha de la investigación	mes - Año	Manual
		46. Conferencias impartidas	Paramétrico	Manual
		47. Fecha de las conferencias impartidas	mes - Año	Manual
		48. Cursos impartidos o recibidos,	Paramétrico mes - Año	Manual Manual
		49. Periodo del curso	SI o No	Manual
		50. Membresia al SNI		
	Experiencia docente	51. Nombre de la materia que ha impartido	Paramétrico	Manual
		52. Periodo laboral como docente	mes - Año	Manual

Lourdes Montiel Garcia
Isabel Pérez Arredondo

El proceso que se realiza es el siguiente:

1. La **cartera** de los aspirantes a profesor, es una actividad constante, la cual consiste en el reclutamiento de personal clasificando en las diferentes áreas académicas y asignaturas en que podría impartir clases cada uno de los aspirantes, se les solicita su currículum, horario de disponibilidad y la indicación de las posibles materias de las diferentes áreas académicas que se tienen clasificadas (actividad desempeñada por el Departamento de Personal Docente). Se selecciona sólo aquellos aspirantes que se consideren los más adecuados.
2. Mediante **Requisiciones de Personal,** los coordinadores de área solicitan al Departamento de Personal Docente (P.D.) a un aspirante para cubrir una vacante, en la cual se indica el ciclo escolar, el área académica, asignatura requerida y horario requerido.
3. Una vez recibida la requisición en el Departamento de P.D. se le asigna un responsable para darle seguimiento y cubrir la vacante.
4. Manualmente se busca en la cartera los posibles aspirantes, en caso de no encontrar se recurre a otras medios de reclutamiento, ya sea anuncios en el periódico y en diversas bolsas de trabajo; se procede a citar a los aspirantes para realizarles una entrevista, la cual se le llama entrevista profunda, además se les pide que traigan una lista de documentos.
5. Al realizar la entrevista se les recoge los documentos solicitados; dependiendo de su desempeño se determina si prosigue con el proceso de selección. El resultado de la evaluación de la entrevista profunda es integrado con su curriculum y los documentos proporcionados en un expediente.
6. Se le solicita al aspirante las referencias laborales para investigación de sus antecendentes y desempeño laboral, si no encuentran ninguna mala referencia se procede con el proceso de selección. Los resultados de la investigación de referencias laborales también es integrado al expediente del aspirante.
7. A los aspirantes aprobados se les concerta una cita con el Coordinador de Area interesado, con la finalidad que les haga una entrevista de conocimientos, denominada "entrevista técnica", el resultado de esta evaluación también es integrado en el expediente del aspirante.

8. A los aspirantes que el Coordinador haya aprobado, les indicará que deberán preparar una clase modelo, así como el tema, la fecha y el lugar para realizarla. Al presentar la clase modelo el Coordinador del Area indicará al Departamento de P.D. qué aspirante ha sido aceptado, para proceder a realizar el proceso del contratación y envía los resultados de la evaluación para ser integrada en el expediente del aspirante.
9. Con la indicación del Coordinador del Area, se procede a dar de alta al profesor para que procedan a asignarle el grupo y materia(s) que impartirá. De igual forma se le genera la credencial que lo acredita como profesor y su contrato laboral. También se le asigna la categoría de pago que le corresponde en base a su grado académico (y antigüedad)

Categoría	Grado académico	Antigüedad	Salario
Profesor A	1. pasante	0	
Profesor B	2. Licenciatura	0	
Profesor C	2. Licenciatura	más de un año	
Profesor D	3. Especialista	0	
Profesor E	3. Especialista	más de un año	
Profesor F	5. Maestría	0	
Profesor G	5. Maestría	más de un año	
Profesor H	6. Doctorado	0	
Profesor D	6. Doctorado	más de un año	

Nota: anualmente se revisa la antigüedad de los profesores que no han sido dados de baja, y se promueve a los profesores que tengan el año cumplido de acuerdo a la tabla. En los casos de los profesores que hayan cursado u obtenido otro nivel académico, se cambia de categoría a solicitud del interesado y con los documentos que comprueben el mismo. El Sistema deberá facilitar esta reclasificación.

El proceso anteriormente descrito es manual, y mediante el nuevo sistema deberá automatizarse, ya que por el gran volumen de información representa una ventaja la explotación y consulta en línea de la información de aspirantes y

profesores. También se solicita que una vez contando con el nuevo sistema se pueda capturar la siguiente información, la cual será solcitada al profesor al ser contratado: si pertenece al Sistema Nacional de Investigadores(SNI); cursos que ha tomado e impartido; las conferencias que ha impartido; publicaciones que ha realizado; distinciones que ha recibido; historia académica: estudios de licenciatura, posgrado, maestrías, doctorados; formación docente y experiencia docente (externa).

7.1 Aplicación de la Metodología de Rebeca Wirfs

7.1.1 Búsqueda de Clases Candidatas

Aspirantes	ASPRNTS
Areas Académicas	AREACDM
Asignatura	ASGNTRA
Documentos entregados	DOCTOSENT
Documentos	DOCTOS
Evaluaciones	EVLCNES
Aspectos calificados de las evaluaciones	ASPCTOS
Evaluaciones de aspirantes	EVLASPR
Referencias laborales	EXPLBRAL
Requisiciones	RQSCNS
Coordinadores de Area	COORDN
Responsables del seguimiento de requisiciones	RSPNSBLS
Profesores	PRFSRES
Categorías	CTGRIAS
Grado Académico	GRDOACD
Aspirante - área académica - asignatura	AAREASIG
Referencias laborales	EXPLBRAL
Historia académica	HSTRIAC
Estudios	ESTDIOS
Universidades	UNVRSDD
Cursos	CURSOS
Distinciones	DSTNCNES
Publicaciones	PBLCNES
Investigaciones	INVSTGCNS
Conferencias	CNFRNCIA
Experiencia Docente	EXPDCNTE
Contratos	CNTRTOS
Bajas y Reingresos de profesores	BJASPRF

Causas de baja	CAUSABJA
Fuentes de Reclutamiento	FNTRCLT
Delegaciones y Municipios	DLGCNES
Nacionalidades	PSES
Sectores Económicos	SCTRESEC
Estados de la República	EDOS
Desarrollo Profesional	DESPROF

7.1.2 Tarjetas de Clases

Clase: *CATALOGOS*	Abstracta
Superclases: Ninguna	
Subclases: AREACDM, ASGNTRA, CTGRIAS, EVLCNES, ASPCTOS, CAUSABJA, FNTRCLT, COORDN, DOCTOS, DLGCNES, PSES, ESTDIOS, UNVRSDD, GRDOACD, SCTRESEC, RSPNSBLS, EDOS.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5, General	
Descripción: Catálogos necesarios para soportar el sistema.	
Responsabilidades Privadas: Conocer en qué catálogo se va a trabajar. Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. leeinfo () return obj_catálogo despliega () return obj_catálogo	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *AREACDM*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de las áreas académicas.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *ASGNTRA*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de las asignaturas que se imparten en la institución.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *CTGRIAS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de categorías de pago de profesores.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *COORDN*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de coordinadores de área.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *EVLCNES*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de las evaluaciones manejadas por la institución.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *ASPCTOS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de los aspectos calificados en las evaluaciones.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *PSES*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5, General	
Descripción: Catálogo de nacionalidades.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *DLGCNES*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5, General	
Descripción: Catálogo de delegaciones y municipios.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *ESTDIOS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de estudios.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *UNVRSDD*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de universidades.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *GRDOACD*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de grados académicos.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *CAUSABJA*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de causas de baja.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automatica la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *FNTRCLT*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de fuentes de reclutamiento.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *DOCTOS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de documentos.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Clase: *SCTRESEC*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de sectores económicos.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *RSPNSBLS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de responsables del seguimiento de requisiciones.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Clase: *EDOS*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5. General	
Descripción: Catálogo de los Estados de la República.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *ASPRNTS*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 3. General	
Descripción: Aspirantes a profesor que forman la cartera de Personal Docente. Permite que se den de alta los datos generales de los aspirantes.	
Responsabilidades Privadas: Generar la clave del aspirante. Permitir la captura de los datos del objeto. Seleccionar la delegación en la que vive (usa DLGCNES(2)). Seleccionar la nacionalidad (usa PSES(2)). Seleccionar el Estado de la República en el que nació (usa EDOS(2)). Seleccionar la fuente de reclutamiento (usa FNTRCLT(2)). Seleccionar el grado académico (usa GRDOACD(2)). Iniciar el proceso de dar de alta los documentos entregados (usa DOCTOSENT(5)) Iniciar el proceso de dar de alta los resultados de las evaluaciones (usa EVLASPR(6)). Inicia el proceso de dar de alta referencias laborales y sus evaluaciones (usa EXPLBRAL(7)). Realizar el cálculo del RFC. Verificar por medio del RFC para evitar duplicidad de entradas. Capturar las áreas académicas y las asignaturas que pueden impartir (usa AAREASIG(1)). Modificar la base de datos. Borrar objetos por medio de la clave de aspirante y su relación con área académica y asignaturas. Modificar la información de los objetos de aspirantes con posibilidad de recálculo del RFC.	
Contratos: 3. Visualizar objetos identificados a través de la clave de aspirante y/u otros atributos. leeaspirante ( ) return ASPRNTES despliega (ASPRNTES) 4. Asignar o cambiar el estado de pertenencia al objeto aspirantes. colocaestado (estado, clave_aspirante) return booleano	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: AAREASIG	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 3. General	
Descripción: Relación de las áreas académicas y asignaturas que un aspirante a profesor está en disponibilidad de impartir.	
Responsabilidades Privadas: Borrar objetos mediante la clave de aspirante, área académica y asignatura. Visualizar la información mediante la clave de aspirante. Modificar la base de datos. Seleccionar el área académica mediante su consulta (usa AREACDM(2)). Seleccionar la asignatura que pertenezca a esa área académica (usa ASGNTRA(2)).	
Contratos: 1. Añadir o modificar objetos a través de la clave de aspirante, área académica y asignatura. agrega () modifica (clave_asp) leeareasig ( ) return AAREASIG	

Clase: *DOCTOSENT*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 3. General	
Descripción: Registro de los documentos que ha entregado cada aspirante a profesor.	
Responsabilidades Privadas: Borrar objetos de esta clase por medio de la identificación del aspirante e indicación del documento a borrar (usa DOCTOS(2)). Modificar la base de datos. Leer y visualizar los objetos de esta clase sin modificarlos.	

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Contratos:

5. Agregar objetos de documentos entregados de acuerdo a la clave del aspirante.
Seleccionar la clase de documento a través de su clave (usa DOCTOS(2)).
Asociar la clase de documento y el aspirante por sus claves.
agregadoctoent ( ) return booleano
Validar que sólo se permita una clave de documentos por aspirante.
valida (clave_asp, clave_doctos) return booleano

Clase: *EVLASPR*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 3, General	
Descripción: Relación de los resultados de las evaluaciones de los aspirantes.	
Responsabilidades Privadas: Borrar evaluaciones indicando la clave de aspirante y la cable de la evaluación. Modificar las calificaciones y observaciones de los aspectos evaluados identificando al aspirante por su clave. Visualizar las evaluaciones y resultados por medio de la indicación de la clave de aspirante. Modificar la base de datos.	
Contratos: 6. Agregar resultados de las evaluaciones aplicadas a los aspirantes. Seleccionar las evaluaciones deseadas (usa EVLCNES(2)). Seleccionar los aspectos de evaluaciones deseadas (usa ASPCTOS(2)). Validar que sólo se permita una clave de evaluación por aspirante. validación (clave_eval,clave_asp) return boolean agregaevlaspr ( ) return booleano	

Lourdes Montiel Garcia
Isabel Perez Arredondo

Clase: *EXPLBRAL*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 4, General	
Descripción: Experiencia laboral y su evaluación como referencia laboral.	
Responsabilidades Privadas: Borrar referencias laborales de un aspirante por medio de la clave de aspirante y el folio de la referencia a borrar. Modificar las referencias laborales de un aspirante por medio de la clave de aspirante. Visualizar las referencias por medio de la indicación de la clave de aspirante. Modificar la base de datos.	
Contratos: 7. Agregar objetos de referencias laborales a través de la clave de aspirante. Agregar referencias laborales a un aspirante por su clave. agregareflab ( ) return booleano Obtener folio en base a la clave de aspirante. folioreflab (clave_asp) return string	

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *ROSCNS*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 3. General	
Descripción: Representa el requerimiento de un profesor que es emitido por un coordinador de área al Depto. de Personal Docente.	
Responsabilidades Privadas: Capturar los datos del personal docente requerido. Obtener un folio por requisición en base al ciclo escolar y un número consecutivo. Seleccionar el coordinador que genera la requisición (usa COORDN(2)). Seleccionar el área académica (usa AREACDM(2)). Seleccionar la asignatura (usa ASGNTRA(2)). Relacionar el área académica y la asignatura (usa AAREASIG (1)). Seleccionar al responsable de cubrir la vacante (usa RSPNSBLS(2)). Modificar atributos del objeto a través del ciclo y el folio de la requisición. Visualizar atributos del objeto con el ciclo y el folio de la requisición. Buscar el personal docente más adecuado (usa ASPRNTS(3)). La selección se efectuará de acuerdo a las claves del área académica y asignatura de la requisición contra los mismos datos registrados para los aspirantes. Se verificará de la selección anterior, aquéllos que cumplan con el horario requerido. Se desplegarán aquellos aspirantes que cumplan con los requisitos. Modificar la base de datos.	
Contratos: Ninguno	

Lourdes Montiel García
Isabel Perez Arredondo

Clase: *PRFSRES*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: General	
Descripción: Registro de profesores.	
Responsabilidades Privadas: Agregar objetos de la clase profesores. Generar la clave de profesor con: los dos primeros dígitos del año del sistema y los otros cuatro dígitos con un consecutivo de acuerdo a los profesores de ese año. Seleccionar el sector económico (usa SCTRESEC(2)). Seleccionar el grado académico (usa GRDOACDM(2)). Relacionar con la clave de aspirante para acceder a toda la información capturada para el aspirante (usa ASPRNTES(3)). Actualizar el estado de pertenencia del objeto aspirante (usa ASPRNTES(4)). Asignar la categoría de pago que se le corresponde (usa CTGRIAS(2)). Calcular la antigüedad del profesor. Borrar un objeto de profesor por medio de su clave. Verificar la posibilidad de borrar un objeto si es que en las clases no existe la clave de ese profesor. Modificar objetos de profesor a través de la clave de profesor y a través de la clave de aspirante modificar esos otros atributos. Modificar el grado académico y redeterminar la categoría de pago correspondiente. Visualizar atributos de profesor a través de la clave de aspirante (usa ASPRNTES(3)). Inicia el proceso de agregar nuevos objetos en su historia académica (usa HSTRIAC(8)). Inicia el proceso de agregar nuevos objetos de desarrollo profesional (usa DESPROF(9)). Modificar la base de datos.	
Contratos: 10. Leer y desplegar los atributos de profesor por clave de profesor. leeprofesor ( ) return PRFSRES despliega (PRFSRES)	

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Clase: *HSTRIAC*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: 4, General	
Descripción: Representa el compendio de la formación académica de un profesor.	
Responsabilidades Privadas: Borrar un objeto de historia académica, identificándolo con la clave de profesor y la clave de estudios. Modificar los atributos de los objetos visualizando los estudios que de él se tengan registrados, pero no se puede modificar la clave de estudio. Visualizar los atributos de la historia académica por medio de la clave de profesor. Modificar la base de datos.	
Contratos: 8. Agregar objetos de la historia académica de una persona. Seleccionar los estudios del profesor (usa ESTDIOS(2)). Seleccionar la institución educativa referida (usa UNVRSDD(2)). Capturar los datos restantes. Dar de alta un registro de historia académica. agregahistoria (HSTRIAC) return booleano	

Clase: *DESPROF*	Abstracta
Superclases: Ninguna	
Subclases: CURSOS, PBLCNES, DSTNCNES, INVSTGCNS, EXPDCNTE, CNFRNCIA.	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4, General	
Descripción: Representa las actividades profesionales complementarias de un profesor.	

Responsabilidades Privadas:
Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Modificar la base de datos.

Contratos:
9. Agregar objetos de la clase apropiada al profesor.
Dar de alta objetos de la clase apropiada.
agregades(clave_prof) return boolean
Generar el número de folio del objeto de acuerdo a la clave de profesor.
foliodes (clave_prof) return string

Clase: *CURSOS*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4, General	

Descripción: Representa dentro las actividades profesionales complementarias de un profesor a los cursos.

Responsabilidades Privadas:
Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio.
Modificar la base de datos.
Estas responsabilidades las hereda de la clase DESPROF.

Contratos:
9. Agregar objetos de la clase al profesor.
Este contrato lo hereda de la clase DESPROF.

Lourdes Montiel García
Isabel Pérez Arredondo

Clase: *DSTNCNES*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4. General	
Descripción: Representa dentro las actividades profesionales complementarias de un profesor a las distinciones recibidas.	
Responsabilidades Privadas: Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio. Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar la base de datos. Estas responsabilidades las hereda de la clase DESPROF.	
Contratos: 9. Agregar objetos de la clase al profesor. Este contrato lo hereda de la clase DESPROF.	

Clase: *PBLCNES*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4. General	
Descripción: Representa dentro las actividades profesionales complementarias de un profesor a los trabajos publicados.	
Responsabilidades Privadas: Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio. Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar la base de datos. Estas responsabilidades las hereda de la clase DESPROF.	
Contratos: 9. Agregar objetos de la clase al profesor. Este contrato lo hereda de la clase DESPROF.	

Lourdes Montiel Garcia
Isabel Perez Arredondo

Clase: *INVSTGCNS*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4. General	
Descripción: Representa dentro las actividades profesionales complementarias de un profesor a las investigaciones realizadas.	
Responsabilidades Privadas: Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio. Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar la base de datos. Estas responsabilidades las hereda de la clase DESPROF.	
Contratos: 9. Agregar objetos de la clase al profesor. Este contrato lo hereda de la clase DESPROF.	

Clase: *CNFRNCIA*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4. General	
Descripción: Representa dentro las actividades profesionales complementarias de un profesor a las conferencias.	
Responsabilidades Privadas: Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio. Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar la base de datos. Estas responsabilidades las hereda de la clase DESPROF.	
Contratos: 9. Agregar objetos de la clase al profesor. Este contrato lo hereda de la clase DESPROF.	

Lourdes Montiel Garcia
Isabel Perez Arredondo

Clase: *EXPDCNTE*	Concreta
Superclases: DESPROF	
Subclases: Ninguna	
Gráfica de Jerarquía: 2	
Gráfica de Colaboraciones: 4, General	
Descripción: Representa dentro las actividades profesionales complementarias de un profesor a la experiencia docente.	
Responsabilidades Privadas: Borrar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar objetos de un profesor de acuerdo a la clave de profesor y al folio. Desplegar objetos de un profesor de acuerdo a la clave de profesor y al folio. Modificar la base de datos. Estas responsabilidades las hereda de la clase DESPROF.	
Contratos: 9. Agregar objetos de la clase al profesor. Este contrato lo hereda de la clase DESPROF.	

Clase: *CNTRTOS*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: General	
Descripción: Representa el contrato que se emite una vez que se ha dado el evento de contratación.	
Responsabilidades Privadas: Agregar un objeto de contratos con base en la clave de profesor. Seleccionar el profesor al que pertenece el contrato (usa PRFSRES(10)). Generar el folio del contrato: los 4 primeros dígitos son el año en el que se genera el contrato y los siguientes 4 serán un consecutivo del número de contratos del año indicado. Capturar el resto de la información relacionada al contrato. Modificar la base de datos. Visualizar los objetos del profesor que se han generado.	
Contratos: Ninguno	

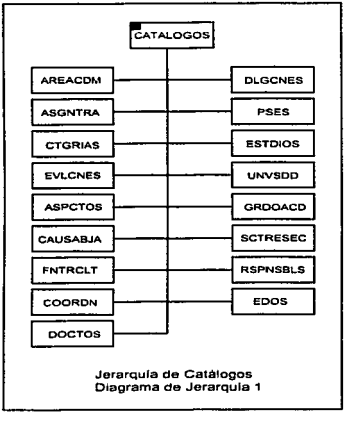
Lourdes Montiel Garcia
Isabel Perez Arredondo

Clase: *BJASPRF*	Concreta
Superclases: Ninguna	
Subclases: Ninguna	
Gráfica de Colaboraciones: General	
Descripción: Bajar y reingresos de profesores.	
Responsabilidades Privadas: Añadir registros de los ciclos en que un profesor se dá de baja sin permitir ingresar la fecha de reingreso ni la clave del coordinador que lo reingresa. Seleccionar la clave de profesor a dar de baja (usa PRFSRES(10)). Seleccionar la causa de baja (usa CAUSABJA(2)). Seleccionar el coordinador que lo dá de baja (usa COORDN(2)). Borrar objetos de un profesor que han sido registrados siempre y cuando no tenga fecha de reingreso, el acceso es a través de la clave de profesor. Modificar atributos de objetos de un profesor que han sido registrados, los atributos susceptibles de modificación son: observaciones, tipo de baja y la fecha de reingreso. Seleccionar el coordinador que reingresa a un profesor (usa COORDN(2)). Visualizar los objetos registrados de un profesor a través de la clave de profesor. Modificar la base de datos.	
Contratos: Ninguno	

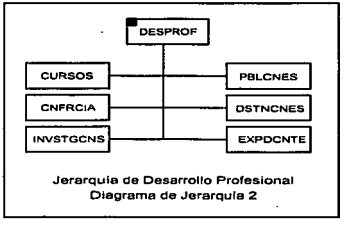
Lourdes Montiel García
Isabel Pérez Arredondo

7.1.3 Diagramas de Jerarquía

En este punto se presentan los diagramas de las jerarquías que se encontraron durante el desarrollo de la metodología.



Jerarquía de Catálogos
Diagrama de Jerarquía 1

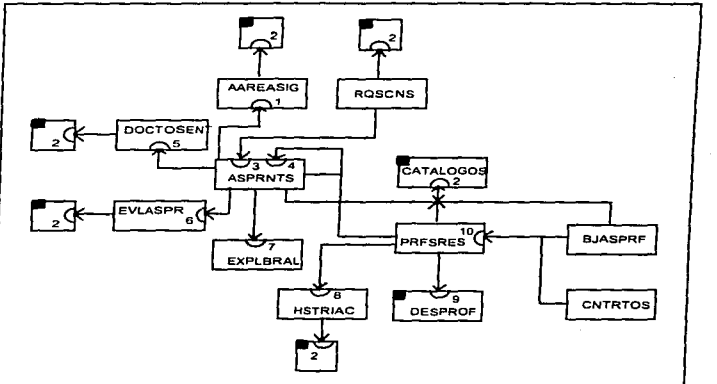


Jerarquía de Desarrollo Profesional
Diagrama de Jerarquía 2

Lourdes Montiel García
Isabel Pérez Arredondo

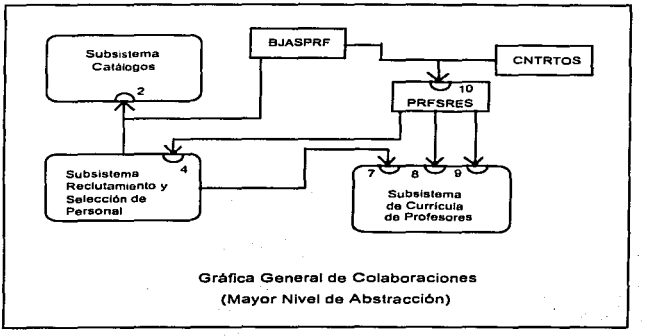
7.1.4 Gráficas de Colaboraciones

En este punto se presentan las diversas gráficas de colaboraciones derivadas del desarrollo de la metodología.



Gráfica General de Colaboraciones

Lourdes Montiel García
Isabel Pérez Arredondo

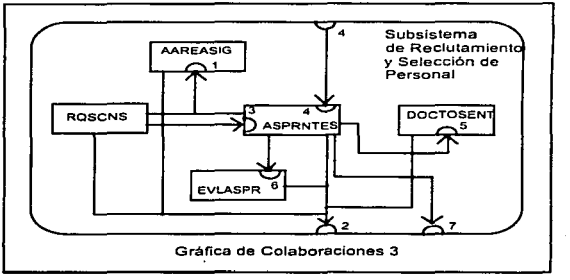


Gráfica General de Colaboraciones
(Mayor Nivel de Abstracción)

Lourdes Montiel García
Isabel Perez Arredondo

7.1.5 Diagramas y Tarjetas de Subsistemas

En este punto se muestran los diagramas de subsistemas que también muestran las colaboraciones que se dan en su interior, así como las tarjetas de subsistemas que nos permiten tener una visión más globalizadora del funcionamiento del sistema.



Gráfica de Colaboraciones 3

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Subsistema: Reclutamiento y Selección de Personal

Clases: AAREASIG, ASPRNTES, RQSCNS, DOCTOSENT, EVLASPR.

Gráfica de Colaboraciones: 3

Descripción: Se encarga de llevar el registro y control del proceso de reclutamiento y selección de personal docente. Para ello agrega, modifica, borra y/o consulta información referida a los datos generales de un aspirante, las asignaturas que está dispuesto a impartir, los documentos que la institución tiene definidos y que el aspirante ya ha entregado y los resultados de las evaluaciones aplicadas. Por otro lado, se encarga del control de las requisiciones desde su generación hasta que se ha cubierto una vacante.

Contratos:

1. Agregar objetos en los que se relacionan claves de aspirantes, área académica y asignatura para las asignaturas que un aspirante está dispuesto a impartir.
Server: AAREASIG

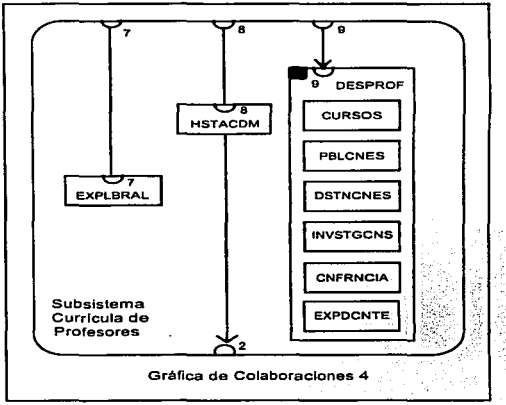
3. Consultar objetos de aspirantes identificados a través de la clave de aspirante.
Despliega información relacionada con un aspirante tanto aquélla que maneja directamente este objeto como la que es manejada por otras clases.
Server: ASPRNTS

4. Asignar o cambiar el estado de pertenencia al objeto aspirantes.
Esta operación se realiza para identificar aquellos aspirantes que ya han sido contratados (profesores) y los que no.
Server: ASPRNTS

5. Agregar objetos de documentos entregados de acuerdo a la clave del aspirante.
Permite la captura de nuevos objetos de documentos entregados por aspirante, realizando una validación para no repetir documentos por aspirante.
Server: DOCTOSENT

6. Agregar o modificar los resultados de evaluaciones aplicadas a los aspirantes.
Esta operación asigna calificaciones y comentarios a las evaluaciones de aspirantes.
Server: EVLASPR

Lourdes Montiel García
Isabel Pérez Arredondo



Gráfica de Colaboraciones 4

Lourdes Montiel García
Isabel Pérez Arredondo

Subsistema: Currícula de Profesores

Clases: EXPLBRAL, HSTACDM, DESPROF, CURSOS, PBLCNES, DSTNCNES, INVSTGCNS, CNFRNCIA, EXPDCNTE.

Gráfica de Colaboraciones: 4

Descripción: Se encarga de llevar el registro y control de la currícula de profesores. Lo que implica la actualización de los registros relacionados con un profesor de: cursos, publicaciones, investigaciones, distinciones, conferencias y experiencia docente. Además de la actualización de la historia académica así como de la experiencia laboral.

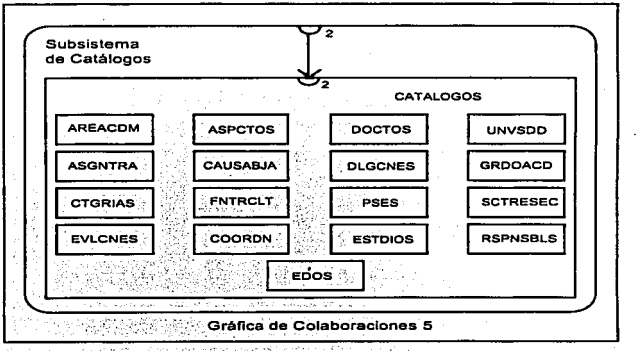
Contratos:

7. Agregar o modificar objetos de referencias laborales.
Esta operación es invocada por la clase de aspirantes con la finalidad de registrar las referencias laborales dadas por un aspirante y otorgarles una calificación, constituyendo el registro de la experiencia laboral en el futuro.
Server: EXPLBRAL

8. Agregar o modificar objetos de la historia académica de una persona.
Esta operación es invocada por la clase de profesores para llevar registro de la historia académica de un profesor.
Server: HSTRIAC

9. Dar de alta objetos relacionados con el desarrollo profesional de un profesor.
Este contrato permite actualizar los registros del desarrollo profesional de profesores, lo cual se ejecuta de acuerdo al método establecido en cada una de las clases que lo heredan.
Este contrato es definido por DESPROF y soportado por CURSOS, PBLCNES, DSTNCNES, INVSTGCNS, CNFRNCIA, EXPDCNTE.
Server: CURSOS, PBLCNES, DSTNCNES, INVSTGCNS, CNFRNCIA, EXPDCNTE.

Lourdes Montiel García
Isabel Pérez Arredondo



Gráfica de Colaboraciones 5

Lourdes Montiel García
Isabel Pérez Arredondo

Subsistema: Catálogos

Clases: AREACDM, ASGNTRA, CTGRIAS, EVLCNES, ASPCTOS, CAUSABAJA, FNTRCLT, COORDN, DOCTOS, DLGCNES, PSES, ESTDIOS, UNVSDD, GRDOACD, SCTRESEC, RSPNSBLS, EDOS.

Gráfica de Colaboraciones: 5

Descripción: Se encarga del mantenimiento de los catálogos que hacen posible el funcionamiento del sistema. También se encarga de consultar información que otras clases y otros subsistemas pueden utilizar, sin cambiarla.

Contratos:
2. Consultar información de la clase solicitada y permitir su uso por otras clases.
Permitir a las clases que requieren hacer uso de la información contenida en los catálogos hacerlo.
Este contrato es definido por CATALOGOS y soportado por AREACDM, ASGNTRA, CTGRIAS, EVLCNES, ASPCTOS, CAUSABAJA, FNTRCLT, COORDN, DOCTOS, DLGCNES, PSES, ESTDIOS, UNVSDD, GRDOACD, SCTRESEC, RSPNSBLS, EDOS.
Server: AREACDM, ASGNTRA, CTGRIAS, EVLCNES, ASPCTOS, CAUSABAJA, FNTRCLT, COORDN, DOCTOS, DLGCNES, PSES, ESTDIOS, UNVSDD, GRDOACD, SCTRESEC, RSPNSBLS, EDOS.

7.1.6 Tarjetas de Contratos

A continuación se presentan las tarjetas en donde se formaliza la definición de los contratos que se han definido a lo largo del sistema, así como los clientes y servidores de cada uno.

Contrato 1: Agregar objetos del tipo AAREASIG.

Server: AAREASIG

Cliente: ASPRNTS

Descripción: Se encarga de agregar objetos en los que se relacionan claves de aspirantes, área académica y asignatura para conocer las asignaturas que cada aspirante puede impartir.

Contrato 2: Consultar información de los catálogos y permitir su uso por otras clases.

Server: CATALOGOS

Cliente: ASPRNTS, AAREASIG, DOCTOSENT, EVLASPR, PRFSRES, HSTRIAC, BJASPRF, REQSCNS.

Descripción: Se encarga de consultar la información de los diversos catálogos, la cual es utilizada por otras clases y subsistemas en el sistema. La consulta de dicha información no implica su modificación.

Contrato 3: Consultar objetos de aspirantes y sus atributos.

Server: ASPRNTS

Cliente: RQSCNS, PRFSRES

Descripción: Se encarga de desplegar información relacionada con los aspirantes, tanto aquella información que se relaciona directamente con él, como la que se relaciona y es manejada por otras clases..

Lourdes Montiel García
Isabel Pérez Arredondo

Contrato 4: Asignar o cambiar el estado de pertenencia al objeto aspirantes.

Server: ASPRNTS

Cliente: PRFSRES

Descripción: Se encarga de cambiar el estado de pertenencia de los aspirantes con el objeto de identificar a aquéllos que ya han sido contratados (profesores) y los que no.

Contrato 5: Agregar objetos del tipo de documentos entregados por los aspirantes.

Server: DOCTOSENT

Cliente: ASPRNTS

Descripción: Se encarga de agregar objetos de documentos entregados por aspirante, realizando la validación para no permitir duplicación de documentación por aspirante.

Contrato 6: Agregar o modificar resultados de evaluaciones.

Server: EVLASPR

Cliente: ASPRNTS

Descripción: Se encarga de asignar calificaciones y comentarios a las evaluaciones aplicadas a los aspirantes a través de una asociación de la clave del aspirante, la clave del tipo de evaluación y la clave del aspecto a evaluar.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Contrato 7: Agregar o modificar objetos de referencias laborales.

Server: EXPLBRAL

Cliente: ASPRNTS

Descripción: Se encarga de agregar o registrar objetos de referencias laborales dadas por un aspirante y otorgarles una calificación, constituyendo el registro de la experiencia laboral de un profesor en el futuro.

Contrato 8: Agregar o modificar objetos de la historia académica de un profesor.

Server: HSTRIAC

Cliente: PRFSRES

Descripción: Se encarga de agregar o modificar objetos para llevar el registro de la historia académica de un profesor. Esto a través de la clave de profesor.

Contrato 9: Agregar objetos relacionados con el desarrollo profesional de un profesor..

Server: DESPROF

Cliente: PRFSRES

Descripción: Se encarga de actualizar los registros de desarrollo profesional de profesores, lo cual se ejecuta de acuerdo al método establecido en cada una de las clases que lo heredan.

Contrato 10: Consultar los objetos de profesor y sus atributos.

Server: PRFSRES

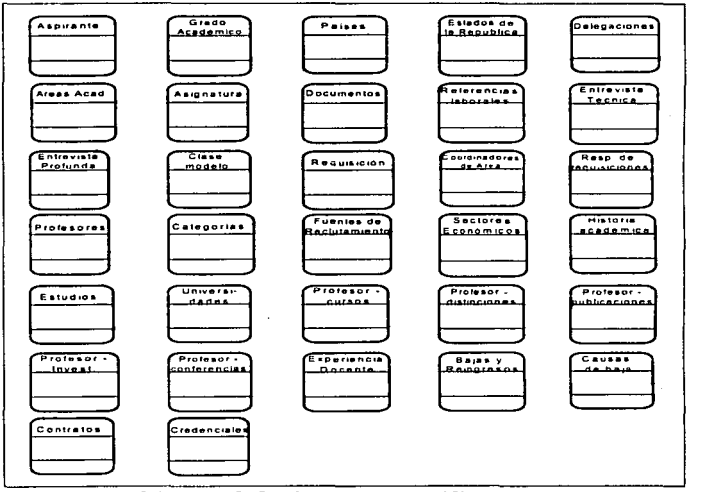
Cliente: BJASPRF, CNTRTOS

Descripción: Se encarga de permitir la consulta de los diversos atributos del objeto profesor y compartir la información con aquellas clases o subsistemas que la requieran sin permitir su modificación..

Lourdes Montiel García
Isabel Pérez Arredondo

7.2 Aplicación de la Metodología de Edward Yourdon

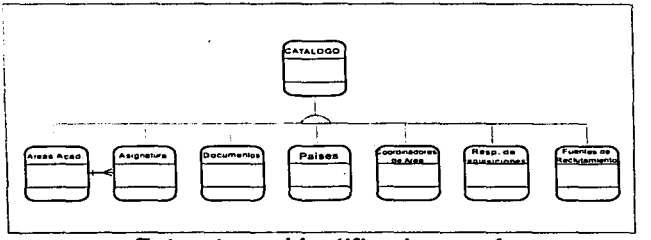
7.2.1 Identificación de Clases&Objetos



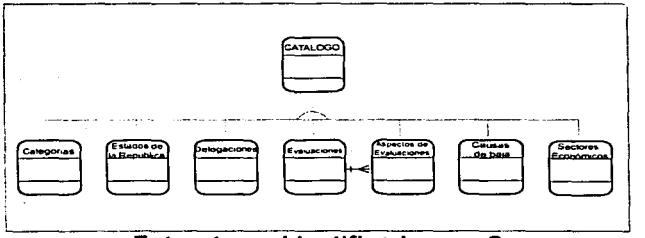
Clases&Objetos identificados

Lourdes Montiel Garcia
Isabel Pérez Arredondo

7.2.2 Identificación de Estructura



Estructuras Identificadas no. 1



Estructuras Identificadas no. 2

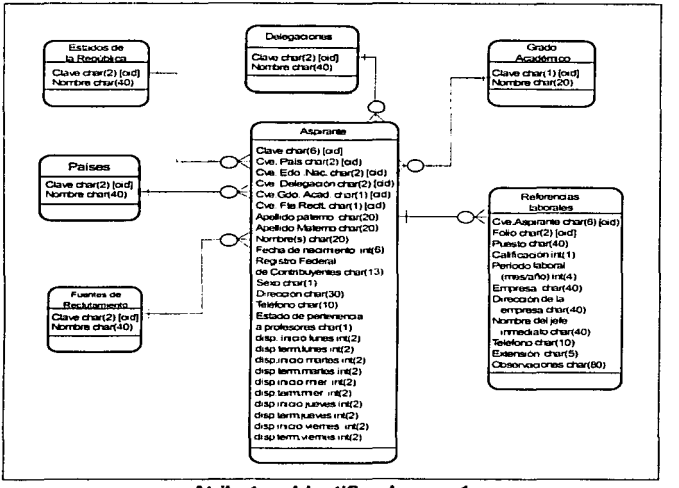
Lourdes Montiel Garcia
Isabel Perez Arredondo

7.2.3 Identificación de Materia

Materias Identificadas

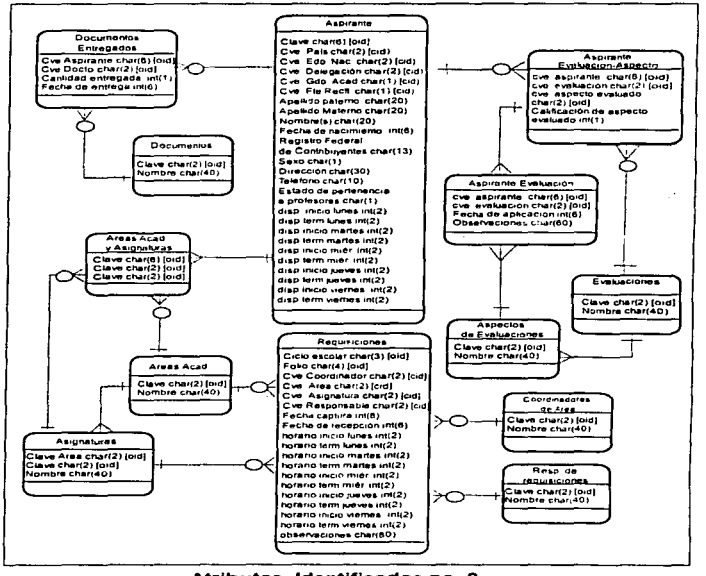
Lourdes Montiel García
Isabel Pérez Arredondo

7.2.4 Identificación de Atributos



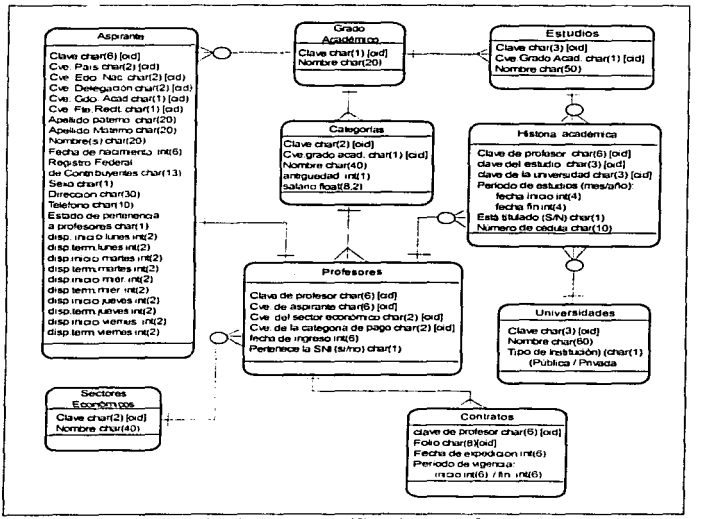
Atributos Identificados no. 1

Lourdes Montiel Garcia
Isabel Perez Arredondo



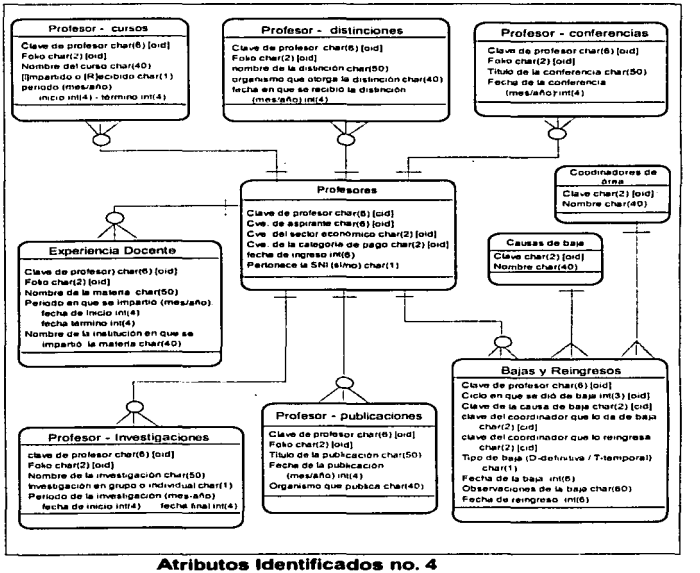
Atributos Identificados no. 2

Lourdes Montiel García
Isabel Pérez Arredondo



Atributos Identificados no. 3

Lourdes Montiel Garcia
Isabel Perez Arredondo



Atributos Identificados no. 4

Lourdes Montiel García
Isabel Pérez Arredondo

7.2.5 Identificación de Servicios

Clases&Objetos identificados como catálogos:

ESPECIFICACIÓN	DELEGACIONES
PROPÓSITO	Catálogo de delegaciones o municipios
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	PAISES
PROPÓSITO	*Catálogo de nacionalidades*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICION DATOS	Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	ESTADOS DE LA REPUBLICA
PROPÓSITO	*Catálogo de estados de la república mexicana*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

Lourdes Montiel García
Isabel Perez Arredondo

ESPECIFICACIÓN	FUENTES DE RECLUTAMIENTO
PROPÓSITO	Alta, baja, modificación y consulta
ATRIBUTOS DESCRIPTIVOS	*Catálogo de fuentes de reclutamiento*
DEFINICIÓN DATOS	Clave Char(2) [oid] Nombre Char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	*AREAS ACADEMICAS*
PROPÓSITO	*Catálogo de Areas Académicas*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	ASIGNATURAS
PROPÓSITO	Catálogo de asignaturas
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave del área académica a la que pertenece la asignatura char(2)* [oid] Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	*DOCUMENTOS*
PROPÓSITO	*Catálogo de Documentos*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(2)* [oid] *Nombre del documento char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	EVALUACIONES
PROPÓSITO	Catálogo de Evaluaciones
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(2)* [oid] *Nombre de la evaluación char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	ASPECTOS DE EVALAUCIONES
PROPÓSITO	Catálogo de Aspectos calificados de las evaluaciones
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de la evaluación char(2)* [oid] *Clave char(2)* [oid] *Nombre del aspecto a evaluar char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta y en relación a la evaluación de que son parte.

Lourdes Montiel García
Isabel Perez Arredondo

ESPECIFICACIÓN	*COORDINADORES DE AREA*
PROPÓSITO	Catálogo de Coordinadores de Area
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(2)* [oid] *Nombre del coordinador char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	RESPONSABLES DE SEGUIMIENTO DE REQ.
PROPÓSITO	Catálogo de Responsables del seguimiento de requisiciones
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(2)* [oid] *Nombre del responsable char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	SECTORES ECONOMICOS
PROPÓSITO	Catálogo de Sectores económicos
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(2)* [oid] *Nombre del sector económico char(40)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	CATEGORIAS
PROPÓSITO	Catálogo de categorías de pago de profesores
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICION DATOS	*Clave char(2)* [oid] clave del grado académico char(1) [cid] *Nombre char(40)* *antigüedad int(1)* *salario float(8,2)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	CAUSAS DE BAJA
PROPÓSITO	Catálogo de Causas de baja
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICION DATOS	Clave char(2) [oid] Descripción de la causa de baja char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

ESPECIFICACIÓN	GRADOS ACADEMICOS
PROPÓSITO	Catálogo de grados académicos
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICION DATOS	*Clave char(1)*[oid] *Nombre char(20)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	ESTUDIOS PROFESIONALES
PROPÓSITO	Catálogo de Estudios profesionales
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(3)* [oid] Clave de grado académico char(1)[cid] *Nombre char(50)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta, en relación su grado académico.

ESPECIFICACIÓN	UNIVERSIDADES
PROPÓSITO	Catálogo de Universidades
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*Clave char(3)* [oid] *Nombre de la institución char(60)* *Tipo de Institución (Pública / Privada) (char(1)*
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

De las Clase&Objetos anteriormente mostrados se ha definido la siguiente plantilla de los servicios que tienen, y se ha generalizado debido a que su comportamiento es común:

Servicios	Descripción
Alta	Se deberá generar la clave de la clase involucrada, un consecutivo, el cual deberá ser de uno a hasta tres dígitos de acuerdo a la definición de cada clase y permitirá la captura de la descripción. Al finalizar se almacenarán a éstos en el medio físico. En los casos que la clase involucrada tenga relación con otra se deberá hacer uso de ésta en forma de consulta, y poder hacer la referencia.

Modificación	Mediante el despliegue de los objetos de cada clase se deberá permitir la selección del objeto a modificar, del cual sólo se podrá modificar la descripción. Al finalizar la edición se almacenarán a éstos en el medio físico. En los casos de que la clase involucrada tenga relación con otra, no deberá permitir el cambio de la clave de ésta cuando sea identificada como oid.
Baja	Este evento implica borrar el objeto identificado mediante la clave de la clase involucrada; al identificar el registro se deberá verificar en las clases que hagan uso de ella, si existe no se permitirá la eliminación del objeto. Una vez identificado el registro se procederá a eliminar el mismo del almacenador físico.
Consulta	Se podrá visualizar o utilizar la información de la clase sin modificarla

Lourdes Montiel García
Isabel Pérez Arredondo

Clases&Objetos principales:

ESPECIFICACIÓN	*ASPIRANTES*
PROPÓSITO	*Aspirantes a profesor, forman la cartera de Personal Docente*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave char(6) [oid] Clave de delegación / municipio char(2) [cid] Clave de país/nacionalidad char(2) [cid] Clave de estado de nacimiento char(2) [cid] Clave fuente de reclutamiento char(2) [cid] clave del grado académico char(1) [cid] Apellido paterno char(20) Apellido Materno char(20) Nombre(s) char(20) Fecha de nacimiento int(6) Registro Federal de Contribuyentes char(13) Sexo char(1) Dirección char(30) Teléfono char(10) Estado de pertenencia a profesores char(1) disp. inicio lunes int(2) disp.term.lunes int(2) disp.inicio martes int(2) disp.term.martes int(2) disp.inicio miér. int(2) disp.term.miér. int(2) disp.inicio jueves int(2) disp.term.jueves int(2) disp.inicio viernes int(2) disp.term.viernes int(2)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue de los aspirantes en la cartera, los cuales son los que tienen el valor 'N' en el estado de pertenenecia a profesores.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Servicios	Descripción
Alta	Para dar de alta objetos en la clase de aspirantes se deberá generar la clave del aspirante con la siguiente nomenclatura: los dos primeros dígitos serán el año de la fecha del sistema y los últimos cuatro dígitos serán un consecutivo de acuerdo a los aspirantes que se tengan en la clase del año que indica los dos primeros dígitos. Deberá permitir la captura de los atributos de la clase(todos son requeridos) al obtener los elementos necesarios para obtener el RFC, mediante el siguiente mecanismo se obtendrá: tomará las dos primeras letras del apellido paterno, la primer letra del apellido materno, y la primer letra del nombre, si tiene más de un nombre y el primer nombre es MARIA o JOSE deberá tomar la primer letra del segundo nombre; a estas letras se le concatenará la fecha de nacimiento con el siguiente orden: año, mes, dia. Con el RFC obtenido se realizará una búsqueda con la finalidad de identificar al aspirante o profesor que tenga el mismo RFC, sino existe algún objeto con éste procederá la captura de los atributos faltantes del objeto y le asignará al estado de pertenencia de profesores una 'N'. Para los atributos que hacen referencia a otras clases se deberá hacer uso de ellas en modo de lectura. Una vez capturados los datos generales se procederá a capturar las áreas académicas y asignaturas en que puede impartir clase, una vez que se termine se procederá a guardar el objeto capturado.
Baja	Se deberá identificar el objeto a borrar por medio de la clave de aspirante y sólo se podrán borrar objetos de la clase al detectar si no hay información en las clases de documentos entregados, evaluaciones y referencias laborales. Si procede el borrado deberá borrarse los objetos de áreas académicas y asignaturas del objeto.

Modificación	Se deberá identificar el objeto a modificar por medio de la clave de aspirante y procederá a permitir el cambio de los diferentes atributos, cuando se modifique el nombre o fecha de nacimiento se deberá indicar que se modificará el RFC y se volverá a validar al sujeto, con opción de cancelar o continuar, en caso de encontrar otro objeto con el RFC obtenido deberá indicarse y regresar los valores anteriores. Al finalizar los cambios deberán almacenarse en el medio físico. También deberá permitir la modificación de los objetos de áreas académicas y asignaturas del aspirante indicado.
Consulta	Se deberá identificar el objeto a consultar por medio de la clave de aspirante y mostrar los atributos del objeto solicitado, así como las áreas académicas y asignaturas que tiene registradas.

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	*AREAS - ASIGNATURAS DE ASPIRANTES*
PROPÓSITO	*Relación de las áreas académicas y asignaturas que puede dar un aspirante*
ATRIBUTOS DESCRIPTIVOS	Alta, baja y consulta
PARAMETRO	Clave de aspirante
DEFINICIÓN DATOS	Clave de aspirante char(6) [oid] *Clave del area académica a la que pertenece la asignatura char(2)* [oid] Clave de la asignatura char(2) [oid]
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las áreas académicas y asignaturas que puede dar un aspirante (o profesor).

Servicios	Descripción
Alta	Se permitirá la selección de las áreas académicas mediante su consulta, una vez indicada la clave se permitirá seleccionar las asignaturas de la área académica correspondiente; una vez seleccionada se guardarán asociadas a la clave del aspirante indicada. Al finalizar se almacenarán a éstos en el medio físico relacionadas con la clave de aspirante indicada. Las clases de áreas académicas y asignaturas deberán ser utilizadas en forma de consulta.
Baja	Este evento implica borrar el objeto identificado mediante la clave de aspirante y las claves de área académica y asignaturas, la cual será mediante el despliegue de las mismas por aspirante y selección de éstas. Una vez identificado el registro se procederá a eliminar el mismo del almacenador físico.
Consulta	Se podrá visualizar o utilizar la información de la clase mediante la clave de aspirante la cual no podrá ser modificada.

Lourdes Montiel Garcia
Isabel Perez Arredondo

ESPECIFICACIÓN	*DOCUMENTOS ENTREGADOS DE ASPIRANTES*
PROPOSITO	*Relación de los documentos que han entregado los aspirantes*
ATRIBUTOS DESCRIPTIVOS	Alta, baja y consulta
DEFINICIÓN DATOS	Clave de aspirante char(6) [oid] *(clave del documento*) char(2)[oid] *Cantidad entregada int(1)* *Fecha de entrega int(6)*
ATRIBUTOS SIEMPRE DERIVABLES	Documentos entregados por los aspirantes (o profesores) con la fecha y cantidad del mismo.

Servicio	Descripción
Alta	Al dar de alta se debe indicar el aspirante al cual se le registrarán documentos, esto es por medio de la clave de aspirante, posteriormente se hará referencia a la clase de documentos por medio de la clave, se debe asegurar que sólo permita una clave de documentos por aspirante de forma única, permitir la actualización de los demás datos, para la fecha de entrega el sistema propondrá la del sistema y para el total de copias inicializará con 1.
Baja	Para borrar objetos de esta clase será por medio de la identificación del aspirante e indicación del documento a borrar. Se realziará mediante el despliegue y selección de los documentos asociados a la clave de aspirante.
Consulta	Por medio de la clave del aspirante se visualizarán los objetos de esta clase, sin permitir su modificación.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

ESPECIFICACIÓN	*EVALUACIONES - ASPIRANTES*
PROPÓSITO	*Relación de las evaluaciones de los aspirantes*
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de aspirante char(6)* [oid] *clave de la evaluación char(2)* [oid] *Fecha de aplicación int(6)* *Observaciones char(60)*
ATRIBUTOS SIEMPRE DERIVABLES	Resultados de las evaluaciones y en su momento de un promedio de las mismas por aspirante (o profesor).

ESPECIFICACIÓN	*ASPECTOS EVALUACIONES - ASPIRANTES*
PROPÓSITO	*Calificaciones de los Aspectos evaluados de las evaluaciones de los aspirantes*
ATRIBUTOS DESCRIPTIVOS	Alta, modificación y consulta
DEFINICIÓN DATOS	*clave de aspirante char(6)* [oid] *clave de la evaluación char(2)* [oid] *clave del aspecto evaluado char(2)* [oid] *Calificación de aspecto evaluado int(1)*
ATRIBUTOS SIEMPRE DERIVABLES	Resultados de las evaluaciones y en su momento de un promedio de las mismas por aspirante (o profesor).

Lourdes Montiel García
Isabel Perez Arredondo

Servicios	Descripción
Alta	Al dar de alta se debe indicar el aspirante al cual se le registrarán evaluaciones, esto es por medio de la clave de aspirante, posteriormente se hará referencia a la clase de evalauciones por medio de la clave, se debe asegurar que sólo permita una clave de la evaluación por aspirante de forma única, al dar de alta la evaluación por medio de un proceso se deberán dar de alta los aspectos de ésta en referencia al aspirante, y finalmente permitir la actualización de los demás datos, para las evaluaciones los valores son del 1 al 4.
Baja	Se debe indicar el aspirante al cual se le borrarán evaluaciones, esto es por medio de la clave de aspirante y la clave de la evaluación, al confirmar la operación se deberá eliminar los objetos de los aspectos de la evaluación y del aspirante indicados.
Modificación	Se debe indicar el aspirante al cual se desea modificar la información de la evaluaciones, sólo se permitirán modificar las observaciones y calificaciones de los aspectos evaluados.
Consulta	Se permitirán visualizar las evaluaciones por medio de la indicación de la clave del aspirante y la evaluación, sin permitir modificaciones.

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	EXPERIENCIA LABORAL
PROPÓSITO	Experiencia laboral y la evaluación de la misma como referencia laboral
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*ASPRNTS (clave de aspirante) char(6)* [oid] *Folio char(2)* [oid] *Puesto char(40)* *Calificación int(1)* *Periodo laboral (mes/año) int(4)* *Empresa char(40)* *Dirección de la empresa char(40)* *Nombre del jefe inmediato char(40)* *Teléfono char(10)* *Extensión char(5)* *Observaciones char(80)*
ATRIBUTOS SIEMPRE DERIVABLES	Listado de la experiancia laboral de aspirantes (o profesores)

Servicios	Descripción
Alta	Se debe indicar al aspirante, al cual se le añadirán las referencias laborales, y se procederá permitir capturar los atributos restantes de la clase, para cada objeto se tendrá que obtener un folio con base en la clave de aspirante. La calificación debe ser de 1 a 4. Todos los atributos son requeridos excepto la extensión.
Baja	Se debe indicar al aspirante, al cual se le borrarán referencias laborales, se deberá por medio del folio que les corresponda identificar la referencia a borrar.
Modificación	Por medio de la clave del aspirante se podrá visualizar y seleccionar el objeto a modificar.
Consulta	Se permitirán visualizar las referencias por medio de la indicación de la clave del aspirante, sin permitir modificaciones.

Lourdes Montiel Garcia
Isabel Perez Arredondo

ESPECIFICACIÓN	*REQUISICIONES*
PROPÓSITO	Requisiciones de personal docente
ATRIBUTOS DESCRIPTIVOS	Alta, modificación y consulta
DEFINICIÓN DATOS	Ciclo escolar char(3) [oid] Folio char(4) [oid] *clave del coordinador char(2)* [cid] *clave del área académica a la que pertenece la asignatura char(2)* [cid] clave de la asignatura char(2) [cid] clave del responsable char(2) [cid] Fecha captura int(6) Fecha de recepción int(6) horario inicio lunes int(2) horario term.lunes int(2) horario inicio martes int(2) horario term.martes int(2) horario inicio miér. int(2) horario term.miér. int(2) horario inicio jueves int(2) horario term.jueves int(2) horario inicio viernes int(2) horario term.viernes int(2) int(2) observaciones char(80)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las requisiciones emitidas por ciclo, coordinador de área, por período de emisión, por responsables, sin asignación de responsables, por área académica y por asignatura.

Operación:	Descripción
Alta	Se solicitará el ciclo escolar en que se genera la requisición de personal, con base en el ciclo se obtendrá automáticamente un folio o un consecutivo, y se permitirá la captura de los atributos restantes, de aquellos atributos que hacen uso de otras clases deben ser en forma de consulta. La fecha de captura se actualizará con la fecha del sistema. Las altas de requisiciones sólo la realizarán los coordinadores.

Modificación	Con el ciclo y el folio de la requisición se podrá realizar la modificación de los siguientes atributos: horario requerido, observaciones, el responsable de la requisición, al actualizar este dato se deberá actualizar el campo de la fecha de recepción con la fecha del sistema, una vez asignado se podrá cambiar el responsable pero la fecha no se deberá actualizar.
Consulta	Con el ciclo y el folio de la requisición se podrá realizar la visualización de los atributos del objeto.

ESPECIFICACIÓN	*PROFESORES*
PROPÓSITO	Registro de Profesores
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave de profesor char(6) [oid] *clave de aspirante char(6)* [cid] clave del sector económico char(2) [cid] clave de la categoria de pago char(2) [cid] fecha de ingreso int(6) Pertenece al SNI (Si o No) char(1)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de los profesores que existen así como la referencia de los datos de evaluaciones, documentos entregados, experiencia laboral en base a su clave de aspirante.

Lourdes Montiel García
Isabel Pérez Arredondo

Servicio	Descripción
Alta	Para dar de alta objetos en la clase de profesores se deberá generar la clave con la siguiente nomenclatura: los dos primeros dígitos serán el año de la fecha del sistema y los últimos cuatro dígitos serán un consecutivo de acuerdo a los profesores que se tengan en la clase del año que indica los dos primeros dígitos. Le solicitará la clave del aspirante que será dado de alta como profesor, por medio de esta clave la información capturada como aspirante será accedida, se deberá permitir la captura de los atributos que faltan de la clase de profesores (todos son requeridos). Para los atributos que hacen referencia a otras clases se deberá hacer uso de ellas en modo de consulta. Se actualizará el estado de pertenencia al objeto de aspirante para identificarlo como profesor, asignandole una 'S'. Una vez capturados los datos generales se procederá a capturar las áreas académicas y asignaturas en que puede impartir clase, una vez que se termine se procederá a guardar el objeto capturado usando la clave de aspirante correspondiente. Al asignar el grado académico se deberá determinar la categoría correspondiente, esto será identificando la categoria con el grado académico indicado y de acuerdo a la fecha de ingreso se determinará si es la antigüedad en años requerida.
Baja	Se deberá solicitar la clave de profesor, para poder borrar un objeto de esta clase se deberá verificar que no haya ninguna clase que haga uso del objeto involucrado (historia académica, cursos, distinciones, investigaciones, etc) por lo que se deberá verificar que en las clases no exista la clave de profesor indicada, si se puede borrar se deberá cambiar el estado del objeto como aspirante de pertenencia a profesores con 'N'.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Modificación	Con la clave de profesor se podrá hacer las modificaciones a los atributos de la clase de profesor y con la clave de aspirante que tiene asignado se podrán hacer modificaciones a los atributos de la clase de aspirante, de igual forma se podrán hacer modificaciones a las áreas académicas y asignaturas con la clave de aspirante indicada. Al modificar el grado académico se deberá determinar la categoría correspondiente, esto será identificando la categoria con el grado académico indicado y de acuerdo a la fecha de ingreso se determinará si es la antigüedad en años requerida.
Consulta	Se podrán visualizar los atributos de profesor y aspirantes se hará por medio de la clave de profesor y a su vez con la clave de aspirante.

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	HISTORIA ACADEMICA
PROPÓSITO	Historia académica de Profesores
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] clave del estudio char(3) [oid] clave de la universidad char(3) [cid] Período de estudios (mes/año): fecha inicio int(4) fecha fin int(4) Está titulado (S/N) char(1) Número de cédula char(10)
ATRIBUTOS SIEMPRE DERIVABLES	Lista de estudios, especialidades grado académicos por estudio del profesor

Servicios	Descripción
Alta	Con la clave de profesor y la clave de los estudios se añadirán los objetos de la clase y se procederá a indicar la universidad, por medio de la clave y se permitirá completar los atributos restantes.
Baja	Con la clave de profesor y la clave de los estudios se identificará el objeto a eliminar de la clase.
Modificación	Se solicitará la clave de profesor y se visualizarán los objetos de los diferentes estudios que se tenga registrados, de los cuales se permitirá la edición, la clave del estudio no puede ser modificada.
Consulta	Por medio de la clave de profesor se podrán visualizar los atributos de la historia académica sin modificarla.

Lourdes Montiel García
Isabel Pérez Arredondo

ESPECIFICACIÓN	CURSOS
PROPÓSITO	Profesor - cursos que ha tomado o impartido
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	clave de profesor char(6) [oid] Folio char(2) [oid] Nombre del curso char(40) [I]mpartido o [R]ecibido char(1) periodo (mes/año): inicio int(4) - término int (4)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de los cursos por profesor, o el total de cursos por profesor.

ESPECIFICACIÓN	DISTINCIONES
PROPÓSITO	Profesor - distinciones que ha recibido
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	clave de profesor char(6) [oid] Folio char(2) [oid] nombre de la distinción char(50) organismo que otorga la distinción char(40) fecha en que se recibió la distinción (mes/año) int(4)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las distinciones por profesor, o el total de distinciones por profesor.

ESPECIFICACIÓN	PUBLICACIONES
PROPÓSITO	Profesor - publicaciones que ha realizado
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] Folio char(2) [cid] Título de la publicación char(50) Fecha de la publicación (mes/año) int(4) Organismo que publica char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las publicaciones por profesor, o el total de publicaciones por profesor.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

ESPECIFICACIÓN	INVESTIGACIONES
PROPÓSITO	Profesor - Investigaciones que ha realizado
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] Folio char(2) [oid] Nombre de la investigación char(50) Investigación en grupo o individual char(1) Período de la investigación (mes-año) fecha de inicio int(4) fecha final int(4)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las investigaciones por profesor, o el total de investigaciones por profesor.

ESPECIFICACIÓN	CONFERENCIA
PROPÓSITO	Profesor - conferencias que ha impartido
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] Folio char(2) [oid] Título de la conferencia char(50) Fecha de la conferencia (mes/año) int(4)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las conferencias por profesor, o el total de conferencias por profesor.

ESPECIFICACIÓN	EXPERIENCIA DOCENTE
PROPÓSITO	Profesor - Experiencia Docente
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	clave de profesor char(6) [oid] Folio char(2) [oid] Nombre de la materia char(50) Período en que se impartió (mes/año): fecha de inicio int(4) fecha término int(4) Nombre de la institución en que se impartió la materia char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de la experiencia docente por profesor, o el total de las ocasiones que impartieron alguna clase por profesor.

Lourdes Montiel García
Isabel Pérez Arredondo

De las últimas 6 clases: cursos, distinciones, publicaciones, investigaciones, conferencias y experiencia docente, sólo se presentan las siguientes plantillas de operación debido a que su funcionamiento es semejante y que las variaciones están indicadas en su respectiva plantilla de clase.

Servicio	Descripción
Alta	Con la clave de profesor se generará un folio o consecutivo y se procederá a completar los atributos restantes.
Baja	Con la clave de profesor se visualizarán los objetos del profesor y se permitirá seleccionar cuál borrar con respecto a su folio.
Modificación	Se solicitará la clave de profesor y se visualizarán los objetos del profesor y se permitirá seleccionar, con respecto a su folio, cuál editar en sus diferentes atributos.
Consulta	Por medio de la clave de profesor y se visualizarán los objetos del profesor sin modificarla.

Lourdes Montiel Garcia
Isabel Perez Arredondo

ESPECIFICACIÓN	CONTRATOS
PROPÓSITO	Registro de contratos emitidos
ATRIBUTOS DESCRIPTIVOS	Alta y consulta
DEFINICIÓN DATOS	*PRFSRES(clave de profesor) char(6)* [oid] Folio char(8)[oid] Fecha de expedición int(6) Periodo de vigencia: inicio int(6) / fin int(6)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de los contratos emitidos durante un período, y por maestro

Servicio	Descripción
Alta	Se solicitará la clave de profesor, el periodo de vigencia del contrato. El proceso para la elaboración de los contratos será el siguiente: El folio del contrato se obtendrá con base en la clave de profesor: los primeros cuatro dígitos son el año en que se genera el contrato y los siguientes cuatro dígitos serán un consecutivo de contratos del año indicado, la fecha de expedición será la fecha del sistema. Los demás datos serán los capturados.
Consulta	Por medio de la clave de profesor se visualizarán los objetos del profesor sin modificarla, podrá visualizar los contratos que para el profesor se han generado.

Lourdes Montiel García
Isabel Pérez Arradondo

ESPECIFICACIÓN	BAJAS Y REINGRESOS DE PROFESORES
PROPÓSITO	Bajas y Reingresos de profesores
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] Ciclo en que se dió de baja int(3) [oid] *clave de la causa de baja char(2)* [cid] *COORDN (clave del coordinador que lo da de baja) char(2)* [cid] *COORDN (clave del coordinador que lo reingresa) char(2)* [cid] Tipo de baja (D-definitiva / T-temporal) char(1) Fecha de la baja int(6) Observaciones de la baja char(60) Fecha de reingreso int(6)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de profesores que se direon de baja de un periodo, por coordinador que los dá de baja y por causa de baja, así como por el tipo de baja: definitivas y temporales. Listado de profesores que reingresaron en un periodo agrupado por el coordinador de área en el que reingresaron.

Servicio	Descripción
Alta	Con la clave de profesor y el ciclo en que se da de baja añadirá el objeto y se procederá completar los atributos restantes. Los atributos que hacen referencia a otras clases deben ser accedidos en modo de lectura, y no debe permitir capturar los datos: fecha de reingreso y clave del coordinador que lo reingresa.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

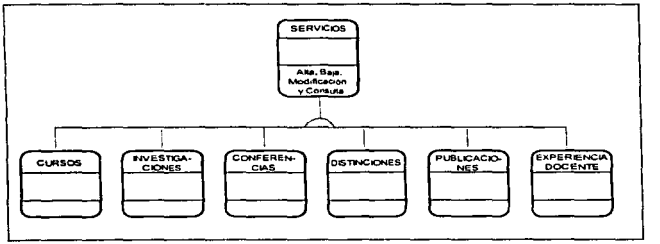
Baja	Con la clave de profesor permitirá visualizar los objetos del profesor que han sido registrados, es decir los diferentes ciclos en que haya sido dado de baja, y deberá permitir seleccionar cuál registro borrar, siempre y cuando no tenga fecha de reingreso.
Modificación	Con la clave de profesor permitirá visualizar los objetos del profesor que han sido registrados, es decir los diferentes ciclos en que haya sido dado de baja, y deberá permitir seleccionar cuál modificar, sólo podrá modificar los siguientes atributos: las observaciones, el tipo de baja y la fecha de reingreso .
Consulta	Por medio de la clave de profesor se visualizarán los objetos del profesor sin modificarla.

Lourdes Montiel García
Isabel Pérez Arredondo

7.2.6 Componente Dominio del Problema (CDP)

Se identifica como Clase Raíz a la denomina como CATALOGO, la cual fué identificada en la estructura del sistema (diagramas 1 y 2 de identificación de estructura).

Se ideintifica una clase de generalización de servicios para las clases de CURSOS, INVESTIGACIONES, CONFERENCIAS, DISTINCIONES, PUBLICACIONES y EXPERIENCIA DOCENTE, los servicios son Alta, Baja, Modificación y Consulta:



Clase de Generalización de Servicios

7.2.7 Componente de Interacción Humana (CIH)

1. Clasificación del CIH
   - Coordinadores de Area
   - Jefe del Departamento de Personal Docente
   - Responsables de Seguimiento de Requisiciones (operadores)

Lourdes Montiel Garcia
Isabel Perez Arredondo

2. Descripción de los humanos y sus tareas

Quién:	Coordinador de Area
Propósito:	Administrar las asignaciones de profesores a grupos - materia (entre otras)
Caracteristicas:	Alto nivel, tiene a su cargo subordinados
Factores Criticos de éxito:	Asignar oportunamente profesores a los grupos - materia

Quién:	Jefe del Departamento de Personal Docente
Propósito:	Proveer a los coordinadores de Area al personal docente capacitado identificado como aspirantes, así como vigilar la actualización de la curricula de profesores, emisiones de contratos y credenciales. Y asignar a los responsables para dar seguimiento a las requisiciones de personal docente. Coordinar las convocatorias para reclutamiento de personal docente.
Caracteristicas:	Alto nivel, tiene a su cargo subordinados
Factores Criticos de éxito:	Contar con una cartera de aspirantes basta y suficiente para dar el tiempo de respuesta requerido.

Quién:	Responsables de Seguimiento de Requisiones
Propósito:	Capturar la información de la curricula de profesores, así como la cartera de aspirantes; y dar seguimiento a las requisiciones, lo cual es identificar a los aspirantes adecuados (con ayuda del sistema) y concertar y preparar agenda para evaluaciones. Participar en el reclutamiento de personal docente
Caracteristicas:	Subordinados
Factores Criticos de éxito:	Contar con la información necesaria para captura y seguimiento de personal docente

3. Jerarquía de Comandos

	Usuario
1. Cartera de Aspirantes	Personal Docente
2. Requisiciones	Coordinadores de Area y Personal Docente
3. Búsqueda Automática de Aspirantes	Personal Docente
4. Altas de Profesores y Currícuta	Personal Docente
5. Generación de Contratos y Credenciales	Personal Docente
6. Bajas y reingresos de Profesores	Personal Docente
7. Catálogos	Personal Docente

7.2.8 Componente de Dirección de Tareas (CDT)

Tarea	Descripción
Despliege de catálogo de selección	En cualquier Clase&Objeto en el que se haga uso de otra Clase&Objeto en forma de consulta para hacer referencia a sus objetos, se deberá incluir el uso de catálogos de éstas para permitir la selección y referencias más rápidas y fácil para el usuario, y se creará por evento, es decir por medio de la activiación de botones o de cierta combinación de teclas.
Búsqueda de Personal Docente en base a la información de requisiciones	Ésta representa una tarea importante del Sistema, ya que por medio de ésta se realizará la búsqueda de aspirantes en la cartera de forma más rápida; el proceso es el siguiente: Se deberá partir del ***ciclo*** de la(s) requisición(es) que se desean procesar, este dato es ***requerido***. Se solicitará un *período* del cual se desean las requisiciones, este dato será *opcional*.

Lourdes Montiel García
Isabel Perez Arredondo

Búsqueda de Personal Docente en base a la información de requisiciones	Se solicitará el *coordinador de área* del que se desean procesar las requisiciones, esta dato será *opcional.* Se solicitará el *Responsable* de seguimiento de la requisición, este dato es *opcional.* Se solicitará el folio de la requisición que se desea procesar, al indicar este dato se deberán inhibir la captura del periodo y la clave del coordinador de área. Una vez con estos datos, se deberá seleccionar de acuerdo a: 1. Si el folio de la requisición se indicó, se seleccionará el objeto de la clase de requisiciones con el ciclo y el folio indicados 2. Si no se indicó el folio se deberán seleccionar las requisiciones que sean del ciclo; si se indicó el periodo se seleccionarán las requisiciones que se encuentren en ese intervalo de acuerdo a la fecha de recepción (sólo áquellas que ya hayan sido asignadas a un responsable); si se indicó el coordinador del área se seleccionarán sólo las requisiciones del coordinador indicado; si se indica la clave del responsable de seguimiento se deberá seleccionar las requisiciones con la clave de responsable de seguimiento indicada. 3. De la requisición se seleccionarán de acuerdo a las claves del área académica y asignatura de la requisición a los aspirantes que tengan indicado las mismas claves (de área y asignatura solicitadas). Se verificará de los aspirantes con este requisito si tienen el horario requerido, comparando fechas de inicio de disponibilidad contra la requerida y las fechas de término de disponibilidad y requerida, finalmente se desplegarán aquellos aspirantes que se han seleccionado con las condiciones indicadas anteriormente..

Lourdes Montiel García
Isabel Pérez Arredondo

El Componente de Dirección de Datos, no se incluye debido a que no se va a desarrollar la aplicación, sin embargo, de acuerdo al lenguaje de desarrollo dependerán los procesos para almacenar los mismos.

Lourdes Montiel Garcia
Isabel Perez Arredondo

7.3 Aplicación de la Metodología de Grady Booch

7.3.1 Identificación de clases y objetos

Aspirante	Clave
ASPRNTS	Apellido paterno
	Apellido Materno
	Nombre(s)
	Fecha de nacimiento
	Registro Federal de Contribuyentes
	Sexo
	Nacionalidad
	Estado de Nacimiento
	Dirección
	Teléfono
	Disponibilidad del horario
	Fuente de reclutamiento
	Grado Académico
Areas Académicas	Clave
AREACDM	nombre
Asignatura	Clave
ASGNTRA	Nombre
	Area a la que pertenece
Documentos entregados	Clave de Aspirante
	Clave de documento
	Cantidad entregada
	Fecha de entrega
Documentos	Clave
DCMNTOS	Nombre
Evaluaciones	Clave
EVLCNES	Nombre de la Evaluación
Aspectos calificados de las evaluaciones	Clave
	Descripción del aspecto a evaluar
ASPCTOS	Clave de la evaluación a la que pertenece
Evaluaciones de aspirantes	Clave de aspirante
	Clave de la Evaluación
	Clave de los aspectos a evaluar
	Evaluación por aspecto evaluado
	observaciones generales

Lourdes Montiel García
Isabel Pérez Arredondo

Referencias laborales	Clave de aspirante
EXPLBRAL	Puesto
	Calificación
	Período laboral
	Empresa
	Dirección de la empresa
	Nombre del jefe inmediato
	Teléfono
	Extensión
	Observaciones
Requisiciones	Ciclo escolar
RQSCNS	Folio
	Coordinador de Area
	Area solicitada
	Asignatura
	horario requerido
	observaciones
	Responsable del seguimiento de la requisición
Coordinadores de Area	Clave
COORDN	Nombre
Responsables del seguimiento	Clave
de requisiciones	Nombre
RSPNSBLS	
Profesores	Clave
PRFSRES	Apellido paterno
	Apellido Materno
	Nombre(s)
	Fecha de nacimiento
	Registro Federal de Contribuyentes
	Sexo
	Nacionalidad
	Estado de Nacimiento
	Dirección
	Teléfono
	Disponibilidad del horario
	Sector económico al que se dedican
	fecha de ingreso
	Fuente de reclutamiento
	Pertenece al SNI
	Categoria

Categorías	Clave
CTGRIAS	descripción
	grado académico
	antigüedad
	salario
Grado Académico	clave
GRDOACD	Nombre del grado académico
Profesor - área académica - asignatura	clave de profesor
	clave de area académica
	clave de asignatura
Documentos entregados de profesores	Clave de profesor
	Clave de documento
	Cantidad entregada
	Fecha de entrega
Evaluaciones del reclutamiento de profesores	Clave de profesor
	Clave de la Evaluación
	Clave de los aspectos a evaluar
	Evaluación por aspecto evaluado
	observaciones generales
Referencias laborales	Clave de profesor
EXPLBRAL	Puesto
	Calificación
	Periodo laboral
	Empresa
	Dirección de la empresa
	Nombre del jefe inmediato
	Teléfono
	Extensión
	Observaciones
Historia académica	Clave de profesor
HSTRIAC	Clave del estudio
	Escuela de procedencia (universidades)
	Periodo de estudios
	Está titulado (S/N)
	Número de cédula
Estudios	Clave
ESTDIOS	Nombre del estudio
	Grado académico
Universidades	Clave
UNVRSDD	Nombre la institución
	Tipo de institución

Lourdes Montiel García
Isabel Pérez Arredondo

Profesor - cursos	Clave del profesor
CURSOS	Nombre del curso
	impartido o recibido
	periodo del curso
Profesor - distinciones	Clave del profesor
DSTNCNES	nombre de la distinción
	organismo que otorga la distinción
	fecha en que se recibió la distinción
Profesor - publicaciones	Clave del profesor
PBLCNES	Título de la publicación
	Fecha de la publicación
	Organismo que publica
Profesor - Investigaciones	Clave del profesor
INVSTGCNS	Nombre de la investigación
	Investigación en grupo o individual
	Periodo de la investigación (mes-año)
Profesor - conferencias	Clave del profesor
CNFRNCIA	Título de la conferencia
	Fecha de la conferencia
Experiencia Docente	Clave de profesor
EXPDCNTE	Nombre de la materia
	Periodo en que se impartió
	Nombre de la institución en que se impartió la materia
Contratos	Folio
	Expediente (clave del profesor)
	Fecha de expedición
	Periodo de contratación
Credenciales	Clave de profesor
	Nombre del profesor
	Fecha de ingreso
Bajas y Reingresos de profesores	Profesor
BJASPRF	Ciclo en que se dió de baja
	Clave de la causa de baja
	Tipo de baja (D-definitiva / T-temporal)
	Fecha de la baja
	Observaciones de la baja
	Clave del coordinador que lo da de baja
	Fecha de reingreso
	Clave del coordinador que lo reingresa
Causas de baja	Clave
CAUSABJA	Descripción de la causa de baja

Lourdes Montiel Garcia
Isabel Perez Arredondo

Debido a que las estructuras de una base de datos representan una generalidad de los elementos que intervienen en un sistema, cada una de las tablas identificadas son clases y los registros que contengan son los objetos, es decir la información en específico serán las instancias y por consecuencia sus objetos.

El siguiente diagrama de clases es la representación general de clases que intervienen en el Sistema.

Lourdes Montiel García
Isabel Pérez Arredondo

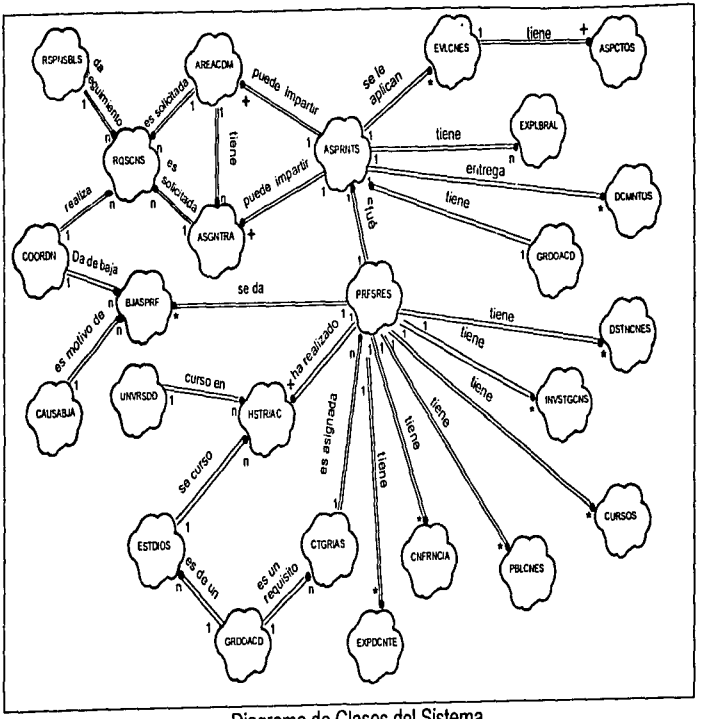


Diagrama de Clases del Sistema

7.3.2 Semántica e implantación de Clases

La implementación de todas las clases se considera Pública, ya que sus atributos pueden ser visualizados por todos que lo requieran, así como la persistencia de todas las clases se considera estática y la concurrencia es secuencial.

En cada una de las clases se ha añadido en algunos atributos la indicación [P] la cual quiere decir que forma parte de la llave primaria de la clase y [S] la cual indica que no forma parte de la llave primaria de la clase, pero que es la llave primaria de otra clase la cual sólo se está usando.

En primer plano se presentarán las plantillas de las clases que se identificaron como catálogos y que se visualizan en el siguiente diagrama:

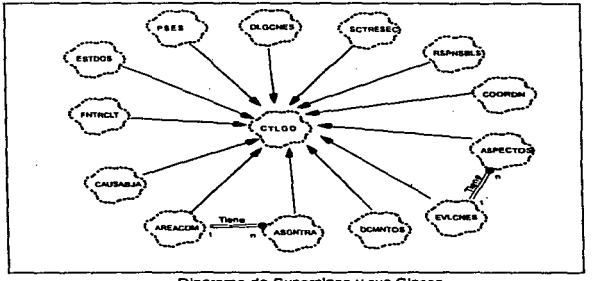


Diagrama de Superclase y sus Clases

Lourdes Montiel García
Isabel Pérez Arredondo

Nombre:	DLGCNES
Documentación:	*Catálogo de delegaciones y municipios*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición no 1
Espacio de complejidad:	Longitud por registro 42 bytes

Nombre:	PSES
Documentación:	*Catálogo de nacionalidades*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No 1
Espacio de complejidad:	Longitud por registro 42 bytes

Lourdes Montiel Garcia
Isabel Perez Arredondo

Nombre:	ESTDOS
Documentación:	*Catálogo de estados de la república mexicana*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No.1
Espacio de complejidad:	Longitud por registro 42 bytes

Nombre:	FNTRCLT
Documentación:	*Catálogo de fuentes de reclutamiento*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave Char(2) [P] Nombre Char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Longitud por registro 42 bytes

Nombre:	*AREACDM*
Documentación:	*Catálogo de Areas Académicas*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 42 bytes

Nombre:	ASGNTRA
Documentación:	Catálogo de asignaturas
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	*AREACDM (area académica a la que pertenece la asignatura) char(2)* [P]
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 44 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Nombre:	*DCMNTOS*
Documentación:	*Documentos*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(2)* [P] *Nombre del documento char(40)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 42 bytes*

Nombre:	EVLCNES
Documentación:	Catálogo de Evaluaciones
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(2)* [P] *Nombre de la evaluación char(40)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 42 bytes*

Lourdes Montiel García
Isabel Perez Arredondo

Nombre:	ASPCTOS
Documentación:	Catálogo de Aspectos calificados de las evaluaciones
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*EVLCNES (clave de la evaluación) char(2)* [P]
Campos:	*Clave char(2)* [P] *Nombre del aspecto a evaluar char(40)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 44 bytes*

Nombre:	*COORDN*
Documentación:	Catálogo de Coordinadores de Area
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(2)* [P] *Nombre del coordinador char(40)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 42 bytes*

Lourdes Montiel García
Isabel Perez Arredondo

Nombre:	RSPNSBLS
Documentación:	Catálogo de Responsables del seguimiento de requisiciones
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(2)* [P]
	Nombre del responsable char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 42 bytes*

Nombre:	SCTRESEC
Documentación:	Catálogo de Sectores económicos
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(2)* [P]
	Nombre del sector económico char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 42 bytes*

Lourdes Montiel Garcia
Isabel Perez Arredondo

Nombre:	CTGRIAS
Documentación:	Catálogo de categorías de pago de profesores
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	GRDOACD (clave del grado académico) char(1) [S]
Campos:	*Clave char(2)* [P] *Nombre char(40)* *antigüedad int(1)* *salario float(8,2)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No.1*
Espacio de complejidad:	*Por registro 49 bytes*

Nombre:	CAUSABJA
Documentación:	Catálogo de Causas de baja
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	Clave char(2) [P] Descripción de la causa de baja char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 42 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Los siguientes también son catálogos pero no comparten las mismas longitudes de los catálogos mencionados anteriormente.

Nombre:	GRDOACD
Documentación:	Catálogo de grados académicos
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(1)*[P] *Nombre char(20)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 21 bytes*

Nombre:	ESTDIOS
Documentación:	Catálogo de Estudios
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	GRDOACD (Clave de grado académico) char(1)[S]
Campos:	*Clave char(3)* [P] *Nombre char(50)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 54 bytes*

Nombre:	UNVRSDD
Documentación:	Catálogo de Universidades
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*Ninguna*
Campos:	*Clave char(3)* [P] *Nombre de la institución char(60)* *Tipo de Institución (Pública / Privada) (char(1)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	*Diagrama de estados de transición No. 1*
Espacio de complejidad:	*Por registro 64 bytes*

De las plantillas anteriores no se incluyen plantillas de operación ya que con las indicaciones proporcionadas en el diagrama de estados de transición y las mismas plantillas de clases queda claramente definido su comportamiento.

Las siguientes plantillas son de las clases que almacenarán la información general del sistema, en las cuales algunas hacen uso de las clases denominadas "catálogos".

Nombre:	*ASPRNTS*
Documentación:	*Aspirantes a profesor, forman la cartera de Personal Docente*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	DLGCNES (delegación / municipio) char(2) [S] PSES (nacionalidad) char(2) [S] ESTDOS (estado de nacimiento) char(2) [S] FNTRCLT (fuente de reclutamiento) char(2) [S] GRDOACD (clave del grado académico) char(1) [S]
Campos:	Clave char(6) [P] Apellido paterno char(20) Apellido Materno char(20) Nombre(s) char(20) Fecha de nacimiento int(6) Registro Federal de Contribuyentes char(13) Sexo char(1) Dirección char(30) Teléfono char(10) Estado de pertenencia a profesores char(1) disp. inicio lunes int(2) disp.term.lunes int(2) disp.inicio martes int(2) disp.term.martes int(2) disp.inicio miér. int(2) disp.term.miér. int(2) disp.inicio jueves int(2) disp.term.jueves int(2) disp.inicio viernes int(2) disp.term.viernes int(2)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 2
Espacio de complejidad:	Por registro 152 bytes

Lourdes Montiel Garcia
Isabel Perez Arredondo

Operación:	Alta
Documentación:	Para dar de alta objetos en la clase de aspirantes se deberá generar la clave del aspirante con la siguiente nomenclatura: los dos primeros dígitos serán el año de la fecha del sistema y los últimos cuatro dígitos serán un consecutivo de acuerdo a los aspirantes que se tengan en la clase del año que indica los dos primeros dígitos. Deberá permitir la captura de los atributos de la clase(todos son requeridos) al obtener los elementos necesarios para obtener el RFC, mediante el siguiente mecanismo se obtendrá: tomará las dos primeras letras del apellido paterno, la primer letra del apellido materno, y la primer letra del nombre, si tiene más de un nombre y el primer nombre es MARIA o JOSE deberá tomar la primer letra del segundo nombre; a estas letras se le concatenará la fecha de nacimiento con el siguiente orden: año, mes, día. Con el RFC obtenido se realizará una búsqueda con la finalidad de identificar al aspirante o profesor que tenga el mismo RFC, sino existe algún objeto con éste procederá la captura de los atributos faltantes del objeto y le asignará al estado de pertenencia de profesores una 'n'. Para los atributos que hacen referencia a otras clases se deberá hacer uso de ellas en modo de lectura. Una vez capturados los datos generales se procederá a capturar las áreas académicas y asignaturas en que puede impartir clase, una vez que se termine se procederá a guardar el objeto capturado.

Operación:	Baja
Documentación:	Se deberá identificar el objeto a borrar por medio de la clave de aspirante y sólo se podrán borrar objetos de la clase al detectar si no hay información en las clases de documentos entregados, evaluaciones y referencias laborales. Si procede el borrado deberá borrarse los objetos de áreas académicas y asignaturas del objeto.

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Modificación
Documentación:	Se deberá identificar el objeto a modificar por medio de la clave de aspirante y procederá a permitir el cambio de los diferentes atributos, cuando se modifique el nombre o fecha de nacimiento se deberá indicar que se modificará el RFC y se volverá a validar al sujeto, con opción de cancelar o continuar, en caso de encontrar otro objeto con el RFC obtenido deberá indicarse y regresar los valores anteriores. Al finalizar los cambios deberán almacenarse en el medio físico. También deberá permitir la modificación de los objetos de áreas académicas y asignaturas del aspirante indicado.

Operación:	Consulta
Documentación:	Se deberá identificar el objeto a consultar por medio de la clave de aspirante y mostrar los atributos del objeto solicitado, así como las áreas académicas y asignaturas que tiene registradas.

Nombre:	*AAREASIG*
Documentación:	*Relación de las áreas académicas y asignaturas que puede dar un aspirante*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Parámetros:	Clave de aspirante char(6)
Interface/Implementación	
Uso de otras clases:	ASPRNTS (clave de aspirante) char(6) [P] AREACDM (clave del area académica a la que pertenece la asignatura) char(2) [P] ASGNTRA (clave de la asignatura) char(2) [P]
Campos:	Ninguno
Operaciones:	Alta, baja y consulta
Estado:	Diagrama de estados de transición No. 3
Espacio de complejidad:	Por registro 10 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Alta
Documentación:	Se permitirá la selección de las áreas académicas mediante su consulta, una vez indicada la clave se permitirá seleccionar las asignaturas de la área académica correspondiente; una vez seleccionada se guardarán asociadas a la clave del aspirante indicada. Al finalizar se almacenarán a éstos en el medio físico relacionadas con la clave de aspirante indicada. Las clases de áreas académicas y asignaturas deberán ser utilizadas en forma de consulta.

Operación:	Baja
Documentación:	Este evento implica borrar el objeto identificado mediante la clave de aspirante y las claves de área académica y asignaturas, la cual será mediante el despliegue de las mismas por aspirante y selección de éstas. Una vez identificado el registro se procederá a eliminar el mismo del almacenador físico.

Operación:	Consulta
Documentación	Se podrá visualizar o utilizar la información de la clase mediante la clave de aspirante la cual no podrá ser modificada.

Lourdes Montiel García
Isabel Pérez Arredondo

Nombre:	*DCTOSENT*
Documentación:	*Relación de los documentos que han entregado los aspirantes*
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	ASPRNTS (clave de aspirante) char(6) [P] *DCMNTOS(clave del documento)* char(2)[P]
Campos:	*Cantidad entregada int(1)* *Fecha de entrega int(6)*
Operaciones:	Alta, baja y consulta
Estado:	Diagrama de estados de transición No. 3
Espacio de complejidad:	Por registro 22 bytes

Operación:	Alta
Documentación:	Al dar de alta se debe indicar el aspirante al cual se le registrarán documentos, esto es por medio de la clave de aspirante, posteriormente se hará referencia a la clase de documentos por medio de la clave, se debe asegurar que sólo permita una clave de documentos por aspirante de forma única, permitir la actualización de los demás datos, para la fecha de entrega el sistema propondrá la del sistema y para el total de copias inicializará con 1.

Operación:	Baja
Documentación:	Para borrar objetos de esta clase será por medio de la identificación del aspirante e indicación del documento a borrar.

Operación:	Consulta
Documentación:	Por medio de la clave del aspirante se visualizarán los objetos de esta clase, sin permitir su modificación.

Lourdes Montiel García
Isabel Perez Arredondo

Nombre:	*EVLASPR*
Documentación:	*Relación de las evaluaciones de los aspirantes*
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*ASPRNTS (clave de aspirante) char(6)* [P] *EVLCNES (clave de la evaluación) char(2)* [P]
Campos:	*Fecha de aplicación int(6)* *Observaciones char(60)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 70 bytes

Nombre:	*ASPASPR*
Documentación:	*Calificaciones de los Aspectos evaluados de las evaluaciones de los aspirantes*
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*ASPRNTS (clave de aspirante) char(6)* [P] *EVLCNES (clave de la evaluación) char(2)* [P] *ASPCTOS (clave del aspecto evaluado) char(2)* [P]
Campos:	*Calificación de aspecto evaluado int(1)*
Operaciones:	Alta, modificación y consulta
Estado:	Diagrama de estados de transición No. 4
Espacio de complejidad:	Por registro 12 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Alta
Documentación:	Al dar de alta se debe indicar el aspirante al cual se le registrarán evaluaciones, esto es por medio de la clave de aspirante, posteriormente se hará referencia a la clase de evaluaciones por medio de la clave, se debe asegurar que sólo permita una clave de la evaluación por aspirante de forma única, al dar de alta la evaluación por medio de un proceso se deberán dar de alta los aspectos de ésta en referencia al aspirante, y finalmente permitir la actualización de los demás datos, para las evaluaciones los valores son del 1 al 4.

Operación:	Baja
Documentación:	Se debe indicar el aspirante al cual se le borrarán evaluaciones, esto es por medio de la clave de aspirante y la calve de la evaluación, al confirmar la operación se deberá eliminar los objetos de los aspectos de la evaluación y del aspirante indicados.

Operación:	Modificación
Documentación:	Se debe indicar el aspirante al cual se desea modificar la información de la evaluaciones, sólo se permitirán modificar las observaciones y calificaciones de los aspectos evaluados.

Operación:	Consulta
Documentación:	Se permitirán visualizar las evaluaciones por medio de la indicación de la clave del aspirante y la evaluación, sin permitir modificaciones.

Lourdes Montiel García
Isabel Perez Arredondo

Nombre:	EXPLBRAL
Documentación:	Experiencia laboral y la evaluación de la misma como referencia laboral
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*ASPRNTS (clave de aspirante) char(6)* [P]
Campos:	*Folio char(2)* [P] *Puesto char(40)* *Calificación int(1)* *Periodo laboral (mes/año) int(4)* *Empresa char(40)* *Dirección de la empresa char(40)* *Nombre del jefe inmediato char(40)* *Teléfono char(10)* *Extensión char(5)* *Observaciones char(80)*
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 267 bytes

Operación:	Alta
Documentación:	Se debe indicar al aspirante, al cual se le añadirán las referencias laborales, y se procederá permitir capturar los atributos restantes de la clase, para cada objeto se tendrá que obtener un folio con base en la clave de aspirante. La calificación debe ser de 1 a 4. Todos los atributos son requeridos excepto la extensión.

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Baja
Documentación:	Se debe indicar al aspirante, al cual se le borrarán referencias laborales, se deberá por medio del folio que les corresponda identificar la referencia a borrar.

Operación:	Modificación
Documentación:	Por medio de la clave del aspirante se podrá visualizar y seleccionar el objeto a modificar.

Operación:	Consulta
Documentación:	Se permitirán visualizar las referencias por medio de la indicación de la clave del aspirante, sin permitir modificaciones.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Nombre:	*RQSCNS*
Documentación:	Requisiciones de personal docente
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*COORDN (clave del coordinador) char(2)* [S] *AREACDM (clave del área académica a la que pertenece la asignatura) char(2)* [S] ASGNTRA (clave de la asignatura) char(2) [S] RSPNSBLS (clave del responsable) char(2) [S]
Campos:	Ciclo escolar char(3) [P] Folio char(4) [P] Fecha captura int(6) Fecha de recepción int(6) horario inicio lunes int(2) horario term.lunes int(2) horario inicio martes int(2) horario term.martes int(2) horario inicio miér. int(2) horario term.miér. int(2) horario inicio jueves int(2) horario term.jueves int(2) horario inicio viernes int(2) horario term.viernes int(2) observaciones char(80)
Operaciones:	Alta, modificación, consulta y búsqueda
Estado:	Diagrama de estados de transición No. 6
Espacio de complejidad:	Por registro 119 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Alta
Documentación:	Se solicitará el ciclo escolar en que se genera la requisición de personal, con base en el ciclo se obtendrá automáticamente un folio o un consecutivo, y se permitirá la captura de los atributos restantes, de aquellos atributos que hacen uso de otras clases deben ser en forma de consulta. La fecha de captura se actualizará con la fecha del sistema. Las altas de requisiciones sólo la realizarán los coordinadores.

Operación:	Modificación
Documentación:	Con el ciclo y el folio de la requisición se podrá realizar la modificación de los siguientes atributos: horario requerido, observaciones, el responsable de la requisición, al actualizar este dato se deberá actualizar el campo de la fecha de recepción con la fecha del sistema, una vez asignado se podrá cambiar el responsable pero la fecha no se deberá actualizar.

Operación:	Consulta
Documentación:	Con el ciclo y el folio de la requisición se podrá realizar la visualización de los atributos del objeto.

Operación:	Búsqueda de Personal Docente
Documentación:	Ésta representa una operación importante, ya que por medio de ésta se realizará la búsqueda de aspirantes en la cartera de forma más rápida; el proceso es el siguiente: Se deberá partir del ***ciclo*** de la(s) requisición(es) que se desean procesar, este dato es ***requerido.*** Se solicitará un *periodo* del cual se desean las requisiciones, este dato será *opcional.*

Lourdes Montiel Garcia
Isabel Perez Arredondo

	Se solicitará el *coordinador de área* del que se desean procesar las requisiciones, esta dato será *opcional*. Se solicitará el *Responsable* de seguimiento de la requisición, este dato es *opcional* . Se solicitará el folio de la requisición que se desea procesar, al indicar este dato se deberán inhibir la captura del período y la clave del coordinador de área. Una vez con estos datos, se deberá seleccionar de acuerdo a: 1. Si el folio de la requisición se indicó, se seleccionará el objeto de la clase de requisiciones con el ciclo y el folio indicados 2. Si no se indicó el folio se deberán seleccionar las requisiciones que sean del ciclo; si se indicó el período se seleccionarán las requisiciones que se encuentren en ese intervalo de acuerdo a la fecha de recepción (sólo áquellas que ya hayan sido asignadas a un responsable); si se indicó el coordinador del área se seleccionarán sólo las requisiciones del coordinador indicado; si se indica la clave del responsable de seguimiento se deberá seleccionar las requisiciones con la clave de responsable de seguimiento indicada. 3. De la requisición se seleccionarán de acuerdo a las claves del área académica y asignatura de la requisición a los aspirantes que tengan indicado las mismas claves (de área y asignatura solicitadas). Se verificará de los aspirantes con este requisito si tienen el horario requerido, comparando fechas de inicio de disponibilidad contra la requerida y las fechas de término de disponibilidad y requerida, finalmente se desplegarán aquellos aspirantes que se han seleccionado con las condiciones indicadas anteriormente.

Lourdes Montiel García
Isabel Pérez Arredondo

Nombre:	*PRFSRES*
Documentación:	Registro de Profesores
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*ASPRNTS (clave de aspirante) char(6)* [S] SCTRESEC (clave del sector económico) char(2) [S] CTGRIAS (clave de la categoría de pago) char(2) [S]
Campos:	Clave de profesor char(6) [P] fecha de ingreso int(6) Pertenece al SNI (Si o No) char(1)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 2
Espacio de complejidad:	Por registro 19 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Alta
Documentación:	Para dar de alta objetos en la clase de profesores se deberá generar la clave con la siguiente nomenclatura: los dos primeros dígitos serán el año de la fecha del sistema y los últimos cuatro dígitos serán un consecutivo de acuerdo a los profesores que se tengan en la clase del año que indica los dos primeros dígitos. Le solicitará la clave del aspirante que será dado de alta como profesor, por medio de esta clave la información capturada como aspirante será accedida, se deberá permitir la captura de los atributos que faltan de la clase de profesores (todos son requeridos). Para los atributos que hacen referencia a otras clases se deberá hacer uso de ellas en modo de lectura. Se actualizará el estado de pertenencia al objeto de aspirante para identificarlo como profesor, asignandole una 's'. Una vez capturados los datos generales se procederá a capturar las áreas académicas y asignaturas en que puede impartir clase, una vez que se termine se procederá a guardar el objeto capturado usando la clave de aspirante correspondiente. Al asignar el grado académico se deberá determinar la categoría correspondiente, esto será identificando la categoría con el grado académico indicado y de acuerdo a la fecha de ingreso se determinará si es la antigüedad en años requerida.

Operación:	Baja
Documentación:	Se deberá solicitar la clave de profesor, para poder borrar un objeto de esta clase se deberá verificar que no haya ninguna clase que haga uso del objeto involucrado, por lo que se deberá verificar que en las clases no exista la clave de profesor indicada, si se puede borrar se deberá cambiar el estado del objeto como aspirante de pertenencia a profesores con 'n'.

Lourdes Montiel García
Isabel Pérez Arredondo

Operación:	Modificación
Documentación:	Con la clave de profesor se podrá hacer las modificaciones a los atributos de la clase de profesor y con la clave de aspirante que tiene asignado se podrán hacer modificaciones a los atributos de la clase de aspirante, de igual forma se podrán hacer modificaciones a las áreas académicas y asignaturas con la clave de aspirante indicada. Al modificar el grado académico se deberá determinar la categoría correspondiente, esto será identificando la categoría con el grado académico indicado y de acuerdo a la fecha de ingreso se determinará si es la antigüedad en años requerida

Operación:	Consulta
Documentación:	Se podrán visualizar los atributos de profesor y aspirantes se hará por medio de la clave de profesor y a su vez con la clave de aspirante.

Nombre:	HSTRIAC
Documentación:	Historia académica de Profesores
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P] ESTDIOS (clave del estudio) char(3) [P] UNVRSDD (clave de la universidad) char(3) [S]
Campos:	Periodo de estudios (mes/año): fecha inicio int(4) fecha fin int(4) Está titulado (S/N) char(1) Número de cédula char(10)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 27 bytes

Operación:	Alta
Documentación:	Con la clave de profesor y la clave de los estudios se añadirán los objetos de la clase, y se procederá a indicar la universidad, por medio de la clave, y se permitirá completar los atributos restantes.

Operación:	Baja
Documentación:	Con la clave de profesor y la clave de los estudios se identificará el objeto a eliminar de la clase.

Operación:	Modificación
Documentación:	Se solicitará la clave de profesor y se visualizarán los objetos de los diferentes estudios que se tenga registrados, de los cuales se permitirá la edición, la clave del estudio no puede ser modificada.

Operación:	Consulta
Documentación:	Por medio de la clave de profesor se podrán visualizar los atributos de la historia académica sin modificarla.

Nombre:	CURSOS
Documentación:	Profesor - cursos que ha tomado o impartido
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] Nombre del curso char(40) [I]mpartido o [R]ecibido char(1) periodo (mes/año): inicio int(4) - término int (4)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No.1
Espacio de complejidad:	Por registro 53 bytes

Nombre:	DSTNCNES
Documentación:	Profesor - distinciones que ha recibido
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] nombre de la distinción char(50) organismo que otorga la distinción char(40) fecha en que se recibió la distinción (mes/año) int(4)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No.1
Espacio de complejidad:	Por registro 100 bytes

Nombre:	PBLCNES
Documentación:	Profesor - publicaciones que ha realizado
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] Título de la publicación char(50) Fecha de la publicación (mes/año) int(4) Organismo que publica char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 100 bytes

Nombre:	INVSTGCNS
Documentación:	Profesor - Investigaciones que ha realizado
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] Nombre de la investigación char(50) Investigación en grupo o individual char(1) Periodo de la investigación (mes-año) fecha de inicio int(4) fecha final int(4)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 63 bytes

Nombre:	CNFRNCIA
Documentación:	Profesor - conferencias que ha impartido
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] Titulo de la conferencia char(50) Fecha de la conferencia (mes/año) int(4)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 58 bytes

Nombre:	EXPDCNTE
Documentación:	Profesor - Experiencia Docente
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(2) [P] Nombre de la materia char(50) Periodo en que se impartió (mes/año): fecha de inicio int(4) fecha término int(4) Nombre de la institución en que se impartió la materia char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 1
Espacio de complejidad:	Por registro 102 bytes

Lourdes Montiel García
Isabel Pérez Arredondo

De las últimas 6 clases: cursos, distinciones, publicaciones, investigaciones, conferencias y experiencia docente, sólo se presentan las siguientes plantillas de operación debido a que su funcionamiento es semejante y que las variaciones están indicadas en su respectiva plantilla de clase.

Operación:	Alta
Documentación:	Con la clave de profesor se generará un folio o consecutivo y se procederá a completar los atributos restantes.

Operación:	Baja
Documentación:	Con la clave de profesor se visualizarán los objetos del profesor y se permitirá seleccionar cuál borrar con respecto a su folio.

Operación:	Modificación
Documentación:	Se solicitará la clave de profesor y se visualizarán los objetos del profesor y se permitirá seleccionar, con respecto a su folio, cuál editar en sus diferentes atributos.

Operación:	Consulta
Documentación:	Por medio de la clave de profesor y se visualizarán los objetos del profesor sin modificarla.

Nombre:	CNTRTOS
Documentación:	Registro de contratos emitidos
Cardinalidad:	*n*
Jerarquia:	
Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P]
Campos:	Folio char(8)[P]
	Fecha de expedición int(6)
	Periodo de vigencia: inicio int(6) / fin int(6)
Operaciones:	Alta y consulta
Estado:	Diagrama de estados de transición No 5
Espacio de complejidad:	Por registro 20 bytes

Operación:	Alta
Documentación:	Se solicitará la clave de profesor, el periodo de vigencia del contrato. El proceso para la elaboración de los contratos será el siguiente: El folio del contrato se obtendrá con base en la clave de profesor: los primeros cuatro digitos son el año en que se genera el contrato y los siguientes cuatro dígitos serán un consecutivo de contratos del año indicado, la fecha de expedición será la fecha del sistema. Los demás datos serán los capturados.

Operación:	Consulta
Documentacion:	Por medio de la clave de profesor se visualizarán los objetos del profesor sin modificarla, podrá visualizar los contratos que para el profesor se han generado.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Nombre:	BJASPRF
Documentación:	Bajas y Reingresos de profesores
Cardinalidad:	*n*
Jerarquía: Superclases:	*Ninguna*
Interface/Implementación	
Uso de otras clases:	*PRFSRES(clave de profesor) char(6)* [P] CAUSABJA*(clave de la causa de baja) char(2)* [S] *COORDN (clave del coordinador que lo da de baja) char(2)* [S] *COORDN (clave del coordinador que lo reingresa) char(2)* [S]
Campos:	Ciclo en que se dió de baja int(3) [P] Tipo de baja (D-definitiva / T-temporal) char(1) Fecha de la baja int(6) Observaciones de la baja char(60) Fecha de reingreso int(6)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No. 4
Espacio de complejidad:	Por registro 79 bytes

Operación:	Alta
Documentación:	Con la clave de profesor y el ciclo en que se da de baja añadirá el objeto y se procederá completar los atributos restantes. Los atributos que hacen referencia a otras clases deben ser accedidos en modo de lectura, y no debe permitir capturar los datos: fecha de reingreso y clave del coordinador que lo reingresa.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Operación:	Baja
Documentación:	Con la clave de profesor permitirá visualizar los objetos del profesor que han sido registrados, es decir los diferentes ciclos en que haya sido dado de baja, y deberá permitir seleccionar cuál registro borrar, siempre y cuando no tenga fecha de reingreso.

Operación:	Modificación
Documentación:	Con la clave de profesor permitirá visualizar los objetos del profesor que han sido registrados, es decir los diferentes ciclos en que haya sido dado de baja, y deberá permitir seleccionar cuál modificar, sólo podrá modificar los siguientes atributos: las observaciones, el tipo de baja y la fecha de reingreso .

Operación:	Consulta
Documentación:	Por medio de la clave de profesor se visualizarán los objetos del profesor sin modificarla.

Lourdes Montiel García
Isabel Perez Arredondo

Diagramas de Estados de Transición

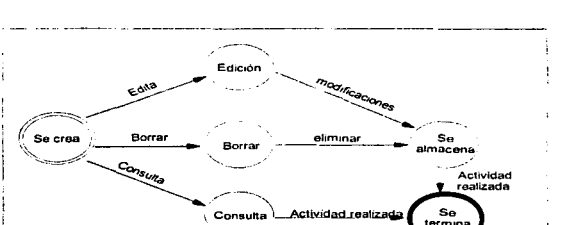


Diagrama de Estados de Transición No . 1

Evento:	Edita
Documentación :	El evento de edición implica el añadir o modificar objetos, para el caso de añadir se deberá generar la clave de la clase involucrada (un consecutivo de hasta dos dígitos, de acuerdo a la definición de cada clase) y permitirá la captura o modificación de la descripcion.

Evento:	Modificaciones
Documentación :	Al finalizar la edición (añadir o modificar) se almacenarán a éstos en el medio físico.

Lourdes Montiel Garcia
Isabel Perez Arredondo

Evento:	Borrado
Documentación:	Este evento implica borrar el objeto identificado mediante la clave de la clase involucrada; al identificar el registro se deberá verificar en las clases que hagan uso de ella, si existe no se permitirá la eliminación del objeto.

Evento:	Eliminar
Documentación:	Una vez identificado el registro se procederá a eliminar el mismo del almacenador físico.

Evento:	Consulta
Documentación:	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento:	Actividad realizada
Documentación:	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

Lourdes Montiel García
Isabel Perez Arredondo

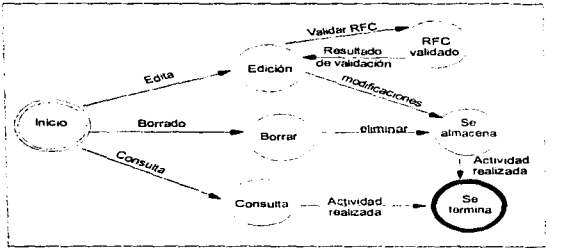


Diagrama de Estados de Transición No . 2

Evento:	Edita
Documentación :	El evento de edición implica el añadir o modificar objetos, para el caso de añadir se deberá generar la clave de aspirante o profesor y se procedan a editar los atributos faltantes. Al obtener los apellidos, nombre(s) y fecha de nacimiento se generará el RFC, con el cual ejecutará el evento de validación del RFC.

Evento:	Validar RFC
Documentación :	Con el RFC obtenido se verificará si exite algún objeto con el RFC indicado; si existiera, se le indicará al usuario la clave de aspirante o profesor que la tiene, dando la oportunidad de proseguir con la captura. Ya que el RFC obtenido no cuenta con la homoclave, en el momento que se valide con este elemento y se identifique a un objeto con el mismo RFC, se indicará que el RFC es idéntico y completo y no permitirá su captura, más la indicación mencionda.

Evento:	Resultado de la validación
Documentación:	Del resultado del evento de validación del RFC, se indicará se continúa o no con la captura de los datos restantes.

Evento:	Modificaciones
Documentación:	Al finalizar la edición (añadir o modificar) se almacenará a éstos en el medio físico.

Evento:	Borrado
Documentación:	Este evento implica borrar el objeto identificado mediante la clave de aspirante o profesor, según sea la clase involucrada; al identificar el registro se deberá verificar en las clases que hagan uso de ella, si alguna clase está haciendo uso del objeto a borrar no se permitirá la eliminación del mismo.

Evento:	Eliminar
Documentación:	Si se pude borrar el objeto de acuerdo a la indicaciones anteriores se procederá a eliminar el mismo del almacenador físico.

Evento:	Consulta
Documentación:	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento	Actividad realizada
Documentación:	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

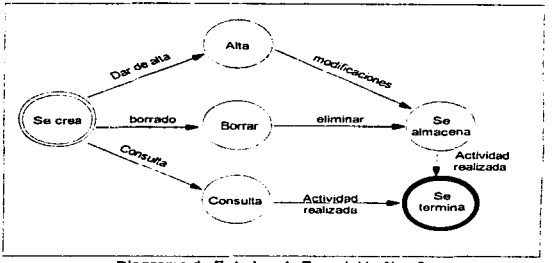


Diagrama de Estados de Transición No . 3

Evento:	Dar de alta
Documentación :	Se deberá dar de alta los objetos mediante la refencia o uso de otras clases, lo cual involucra el uso de los identificadores de las clases, por lo cual no debe permitir modificaciones.

Evento:	Modificaciones
Documentación :	Al finalizar el alta del objeto se almacenará en el medio físico.

Evento:	Borrado
Documentación :	Este evento implica borrar el objeto identificado mediante la clave de la clase involucrada.

Lourdes Montiel García
Isabel Pérez Arredondo

Evento:	Eliminar
Documentación:	Una vez identificado el registro se procederá a eliminar el mismo del almacenador físico.

Evento:	Consulta
Documentación:	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento:	Actividad realizada
Documentacion:	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

Lourdes Montiel García
Isabel Perez Arredondo

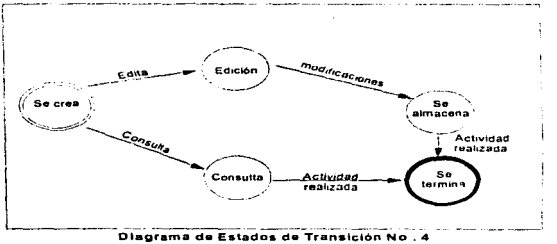


Diagrama de Estados de Transición No . 4

Evento:	Edita
Documentación :	El evento de edición implica el añadir o modificar objetos, para el caso de añadir se hace uso o referencia a otras clases y se procede a editar los atributos faltantes.

Evento:	Modificaciones
Documentación :	Al finalizar el alta del objeto se almacenará en el medio físico.

Evento:	Consulta
Documentación :	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento:	Actividad realizada
Documentación :	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

Lourdes Montiel García
Isabel Perez Arredondo

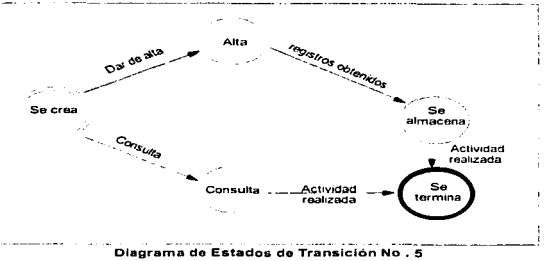


Diagrama de Estados de Transición No . 5

Evento:	Dar de alta
Documentación :	Mediante un proceso se generar los objetos que contenga la clase.

Evento:	Modificaciones
Documentación :	Al finalizar la identificación de los objetos a dar de alta se almacenarán en el medio fisico.

Evento:	Consulta
Documentacion :	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento:	Actividad realizada
Documentación :	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

Lourdes Montiel García
Isabel Pérez Arredondo

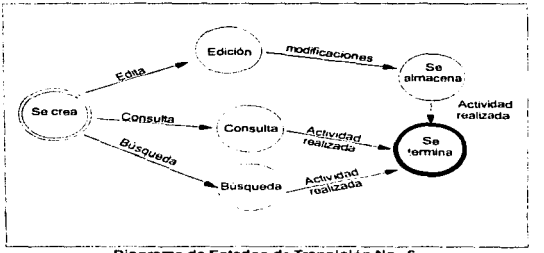


Diagrama de Estados de Transición No . 6

Evento:	Edita
Documentación :	El evento de edición implica el añadir o modificar objetos, para el caso de añadir se hace uso o referencia a otras clases y se procede a editar los atributos faltantes.

Evento:	Modificaciones
Documentación :	Al finalizar el alta del objeto se almacenará en el medio fisico.

Evento:	Consulta
Documentación :	Se podrá visualizar o utilizar la información de la clase sin modificarla

Evento:	Actividad realizada
Documentación :	Una vez terminado cualquier evento se termina cerrando archivos, eliminando la memoria utilizada, etc.

Lourdes Montiel Garcia
Isabel Pérez Arredondo

Evento:	Búsqueda
Documentación :	Proceso de búsqueda de aspirantes que tengan el área académica, asignatura y horario de disponibilidad requisitados.

Diagramas de Módulos

Por útimo los siguientes diagramas nos dan la forma en que el Sistema propuesto se compone modularmente.

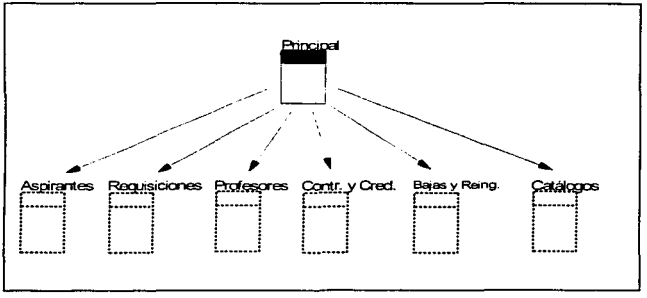


Diagrama de Módulos

Lourdes Montiel García
Isabel Pérez Arredondo

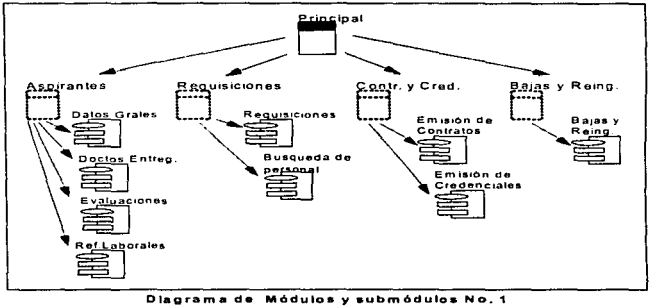


Diagrama de Módulos y submódulos No. 1

Lourdes Montiel García
Isabel Pérez Arredondo

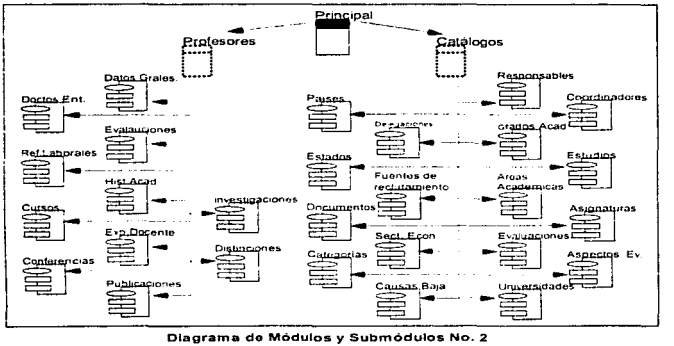


Diagrama de Módulos y Submódulos No. 2

Lourdes Montiel Garcia
Isabel Perez Arredondo

CONCLUSIONES

Del estudio y aplicación de las metodologías expuestas podemos concluir que las metodologías orientadas al objeto son el resultado de una evolución de las metodologías tradicionales o estructuradas, la cual se dió de la necesidad de tener un método de ingeniería de software que de nacimiento a sistemas que al liberarse no sean obsoletos y con visión a continuar su crecimiento y que conceptualizaran desde el análisis y diseño la aplicación para desarrollarla con lenguajes orientados al objeto.

Su objetivo es el incrementar el capital de software mediante componentes reutilizables y con visión de crecimiento, lo cual implica que la aplicación de metodologías orientadas al objeto darán e incrementarán sus resultados en el desarrollo de varios sistemas, y por lo tanto a largo plazo.

Las metodologías estudiadas visualizan el ciclo de vida de un sistema de forma cíclica, es decir creciente, pero cada uno lo representa de forma distinta, sin embargo manejan el mismo concepto.

De forma general las metodologías involucran los siguientes pasos:

Metodología de Rebeca Wirfs	Fase Exploratoria Búsqueda de Clases Búsqueda de Relaciones Identificación de Colaboraciones entre Clases Fase Analítica Definición de Jerarquías Definición de Subsistemas Definición de Protocolos

Lourdes Montiel García
Isabel Pérez Arredondo

Metodología de Edward Yourdon	1.	Identificación de Clases&Objetos
	2.	Identificación de estructura
	3.	Identificación de la materia
	4.	Identificación de los atributos
	5.	Identificación de los servicios
	6.	diseño del componente del dominio del problema
	7.	diseño del componente de interacción humana
	8.	diseño del componente de dirección de tareas
	9.	diseño del componente de dirección de datos
Metodología de Grady Booch	1.	Identificación de clases y objetos
	2.	Identificar la semántica de clases y objetos
	3.	Identificar las relaciones entre clases y objetos
	4.	Implantar clases y objetos

Para la identificación de las clases y objetos los autores dan una serie de recomendaciones, entre ellas se tienen criterios semejantes y no contradictorios. Durante la aplicación (caso práctico desarrollado en el capitulo 7) inicialmente utilizamos los criterios de identificación de clases y objetos recomendados por Wirfs y Yourdon, pero en los casos que nos fué difícil la identificación recurrimos a las propuestas de clasificación mencionadas por Booch (categorización clásica, agrupación conceptual y teoria del prototipo). Por lo cual recomendamos el estudio de las tres, ya que nos amplian el criterio de identificación y definición de clases y objetos, lo cual nos facilitará la aplicación de cualquier metodología elegida.

En la metodología de Rebeca Wirfs se identificó que no abarca todo el ciclo de vida de los sistemas, ya que sólo contempla el diseño; lo cual representa una limitante para su aplicación; por otro lado, propone el manejo de tarjetas para la identificación y descripción de clases, durante la aplicación de esta metodología, inicialmente resulto práctico el manejo de las tarjetas, pero a medida que se avanzaba en el desarrollo y el número de consultas a otras tarjetas se incrementaba, se dificultaba su manipulación aún considerando que el sistema no es de grandes

dimensiones. Además en ellas no se contempla la descripción de los atributos de las diferentes clases que se identifiquen. Como muestra citamos el caso de la clase DLGCNES (delegaciones), de la cual se muestran a continuación, en primer término el de Wirfs-Brock, después el de Yourdon y por último el de Booch. En ellas se nota la carencia de la definición de atributos en las tarjetas de Wirfs-Brock, ya que dichos atributos hacen más completa la definición de una clase.

Clase: *DLGCNES*	Concreta
Superclases: CATALOGOS	
Subclases: Ninguna.	
Gráfica de Jerarquía: 1	
Gráfica de Colaboraciones: 5, General	
Descripción: Catálogo de delegaciones y municipios.	
Responsabilidades Privadas: Crear nuevos registros en los catálogos. Modificar registros del catálogo. Borrar registros en el catálogo. Generar de manera automática la clave del objeto de la base. Todas estas responsabilidades las hereda de la superclase CATALOGOS.	
Contratos: 2. Consultar información de la clase solicitada y permitir su uso por otras clases. Este contrato lo hereda de la superclase CATALOGOS.	

ESPECIFICACIÓN	DELEGACIONES
PROPÓSITO	Catálogo de delegaciones o municipios
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	Clave char(2) [oid] Nombre char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Despliegue o reporte del contenido del catálogo mediante su consulta

Nombre:	DLGCNES
Documentacion:	*Catálogo de delegaciones y municipios*
Cardinalidad:	*n*
Jerarquia:	
Superclases:	*CTLGO (catalogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición no 1
Espacio de complejidad:	Longitud por registro 42 bytes

Con base en estas observaciones sugerimos la aplicación de esta metodología cuando se traten de sistemas pequeños y que a las tarjetas se les añada la descripción de los atributos de cada clase.

En la metodología de Edward Yourdon hace la indicación que para empezar a conocer el problema se tiene que estudiar la documentación existente del mismo y que si se cree necesario se puede acudir con el usuario, lo cual deja al usuario como una fuente de información para el análisis en un segundo término; lo que a nuestro criterio no debe ser, ya que el problema, su comportamiento, las necesidades, los conflictos reales, y la problemática a resolver es identificada totalmente de las indicaciones que el usuario haga, y en su momento como se sustenta en la metodología de Grady Booch, el usuario lo menciona, pero el experto deberá identificarlo, sin que necesariamente el usuario lo identifique como un problema.

Como es notable en los pasos que cada autor ha identificado para el desarrollo de sistemas, la metodología de Edward Yourdon es la más completa, no solo por ser la que incluye el mayor número de pasos a realizar (los cuales no

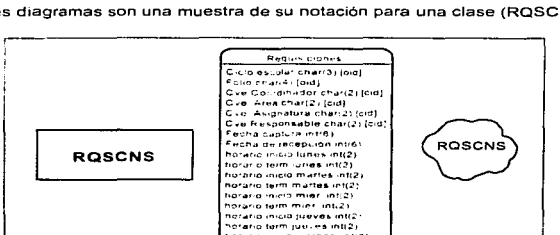
necesariamente deben ser en ese orden), sino porque nos indica los diferentes aspectos que se deben considerar en el desarrollo, lo cual nos ayuda a no dejar aspectos sin evaluar o sin contemplar, además nos da la pauta de asegurarnos de que los diferentes componentes que conforman un sistema sean definidos.

En el desarrollo de la aplicación mediante la metodología de Yourdon se identificaron y definieron más aspectos que en las otras metodologías, ya que contamos con lo que él llama los Componentes del Diseño. Cabe mencionar que el alcance de las metodologías de Wirfs-Brock y Booch llega hasta la que Yourdon denomina Componente del Dominio del Problema. Un ejemplo de esto es la definición del Componente de Interacción Humana, el cual nos ayuda a definir la jerarquía de comandos de acuerdo a las necesidades de este componente hacia el sistema. El diagrama de Jerarquía de Comandos se muestra a continuación:

	Usuario
1. Cartera de Aspirantes	Personal Docente
2. Requisiciones	Coordinadores de Area y Personal Docente
3. Búsqueda Automática de Aspirantes	Personal Docente
4. Altas de Profesores y Currícula	Personal Docente
5. Generación de Contratos y Credenciales	Personal Docente
6. Bajas y reingresos de Profesores	Personal Docente
7. Catálogos	Personal Docente

La notación que sugiere Edward Yourdon, especificamente para la definición de clases&objetos y sus atributos y con ellos las relaciones que existen entre ellas, es muy semejante a las manejadas en herramientas CASE comerciales, lo cual nos da la posibilidad de realizar la documentación de éstas por medio del uso de herramientas también utilizadas en el desarollo tradicional de sistemas, esto representa una ventaja, ya que el esfuerzo e incluso el costo para cambiar de

metodología de desarrollo al hacer uso de estas herramientas sería menor. Una ventaja que encontramos en la notación de Yourdon, respecto a las clases y objetos es que por medio de los diagramas de atributos se visualizan rápidamente las clases, las relaciones entre ellas y sus atributos; en las otras metodologías para poder visualizar la misma información hay que hacer referencia a diagramas o tarjetas (Booch), o bien no se cuenta con toda la información (Wirfs-Brock). Los siguientes diagramas son una muestra de su notación para una clase (RQSCNS):



Para mayor detalle, como lo es las relaciones entre clases, referirse a las páginas 197(wirfs-Brock), 213 (Youdon) y 251 (Booch) de este ejemplar.

Las plantillas que sugiere para la documentación complementaria de las Clases&objetos, son de fácil manejo, sin embargo le añadiríamos la indicación de la longitud por registro y su dinámica de crecimiento de la Clase&objeto que se describe, ya que en el caso de tener un manejador de bases de datos, se requerirá de esta información para la creación y su definición de los parámetros de almacenamiento para el crecimiento de la clase. La misma observación hacemos para la metodología de Wirfs-Brock, que tampoco los incluye. El siguiente es un

ejemplo de una plantilla en la cual se visualiza el aspecto de longitud del registro como sugiere Booch:

Nombre:	ESTDOS
Documentación:	*Catálogo de estados de la república mexicana*
Cardinalidad:	*n*
Jerarquía:	
Superclases:	*CTLGO (catálogo)*
Interface/Implementación	
Uso de otras clases:	Ninguna
Campos:	Clave char(2) [P] Nombre char(40)
Operaciones:	Alta, baja, modificación y consulta
Estado:	Diagrama de estados de transición No.1
Espacio de complejidad:	Longitud por registro 42 bytes

En el caso de la metodología de Grady Booch, la notación para representar las clases y objetos, no es del todo práctica, ya que el elaborarla nos implicaría un mayor esfuerzo, si es que no contamos con una herramienta CASE que maneje esta notación, además que no es tan clara como la anterior, como ya mencionamos no nos indica los atributos que cada una tiene por lo que en el desarrollo de la aplicación tuvimos que hacer referencia a las plantillas de las clases y de objetos para visualizar el sistema en general.

Las plantillas que sugiere son completas e indica el autor la libertad de aumentar o quitar conceptos de acuerdo a las necesidades y requerimientos del sistema involucrado. Aceptando la invitación que se hace aumentaríamos el concepto de atributos siempre derivables que se maneja en la metodología de Yourdon, ya que esto nos da la pauta para conocer que otros atributos podemos obtener a través de la clase. Se muestra una de las plantillas de Yourdon donde aparece este concepto:

ESPECIFICACIÓN	PUBLICACIONES
PROPOSITO	Profesor - publicaciones que ha realizado
ATRIBUTOS DESCRIPTIVOS	Alta, baja, modificación y consulta
DEFINICIÓN DATOS	*clave de profesor char(6)* [oid] Folio char(2) [cid] Título de la publicación char(50) Fecha de la publicación (mes/año) int(4) Organismo que publica char(40)
ATRIBUTOS SIEMPRE DERIVABLES	Listado de las publicaciones por profesor, o el total de publicaciones por profesor.

La notación de la metodología de Grady Booch es muy amplia, la cual consta de diagramas de clases y objetos, de modulos y procesos, de categorías de clases, diagramas de estados de transición de clases, diagramas de sincronización y de subsistemas. De los cuales: el diagrama de categorías de clases, de módulos, de procesos y de subsistemas serían de gran utilidad para documentación en el desarrollo de sistemas muy grandes, ya que representan una gran ayuda para su mayor entendimiento y dar la posibilidad por medio de su uso a un nuevo elemento en el desarrollo de sistemas para entender la integración de los sistemas desarrollados, así como las posibles categorías de clases existentes.

Cada uno de los autores siempre dejan ver como una invitación a los desarrolladores la modificación y mejora de los diferentes elementos que integran su metodología, sin embargo nos dan un camino bien definido para su aplicación en el cual de un paso a otro siempre se pretende continuar con la definición adecuada de los elementos que integran el sistema (clases y objetos). Es por ello que nos hemos tomado la libertad de hacer las sugerencias presentadas en esta sección.

Con base en lo anterior comprobamos que las metodologías orientadas al objeto si promueven el uso de una rica notación para la documentación del análisis y diseño de sistemas de información y mediante su aplicación se obtienen componentes reutilizables, concretamente en este caso, las plantillas y diagramas

pueden volverse a utilizar en sistemas de la institución académica involucrada, y posiblemente en instituciones del mismo giro.

También comprobamos que para el modelado de los objetos y las clases se ha modificado los enfoques de abstracción (encapsulación, ocultamiento de información, comportamiento y atributos) para obtener componentes que nos permitan una constante evolución.

Nos aventuramos a asegurar que a largo plazo la reutilización se extenderá a componentes de código reutilizable, aunándose estos a los componentes de análisis y diseño antes mencionados.

Anteriormente se ha descrito metodologías y técnicas para la definición de sistemas de información con la tecnología orientada al objeto, sin embargo, hay que contemplar el entorno de cualquier sistema de información, es decir, antes de poder identificar y definir clases y objetos, hay que entender y conocer la situación actual del sistema que se desea desarrollar mediante aspectos tales como: el personal que involucra, los flujos de información, los recursos con los que cuenta la organización para la realización de diferentes procesos, para que a partir de éstas se esté en posibilidad de identificar cómo, mediante un sistema de información (software), se puedan optimizar sus procesos y dar a todos aquellos involucrados la solución esperada.

Técnicamente hablando, la documentación de este proceso de asimilación de una situación actual de un sistema y de los cambios que puedan darse por la implantación de otro, con tecnología de cómputo, se sugiere se realice mediante el uso de una de las herramientas de la Ingeniería de Software tradicional que es el Diagrama de Flujo de Datos (DFD).

De entre los factores que se requieren para la aplicación de esta "nueva" tecnología se encuentra el factor humano. De él dependerá con gran medida el éxito o fracaso de su aplicación, es decir, no basta con personal entusiasta por

incursionar en la orientación a objetos, sino también será necesario que cuente con preparación académica, ya que existen organizaciones con personal de bajo nivel que apenas están incursionando en la aplicación de las metodologías tradicionales o que empieza a identificar la necesidad de tener una metodología para el desarrollo de sistemas incluso algunas veces creen que su aplicación no es necesaria, ya que justifican ser más rápidos sin necesidad de documentar y seguir una normatividad definida.

De todo lo anterior se concluye que para la exitosa aplicación de la metodología orientada a objetos se requiere que el personal encargado de toma de decisiones, del liderazgo de los proyectos y los colaboradores (analistas, diseñadores y programadores) asimilen correctamente el nuevo enfoque a utilizar, así como sus alcances y lo que ello demanda.

Recordemos que de acuerdo a la tendencia actual el desarrollo de aplicaciones debe irse involucrando en la tecnología orientada a objetos, debido a la utilización de esta tecnología cada vez más frecuente en sistemas operativos, lenguajes de programación y herramientas de diseño, sólo por mencionar algunas.

Para concretar un sistema orientado a objetos se requiere después de terminar el diseño continuar con la programación a través de un lenguaje de programación orientado a objetos. Para la selección del lenguaje OO más adecuado a los reqerimientos y recursos de la organización, deben atander una serie de puntos, entre ellos estudiar la variedad de productos que ofrece el mercado para la plataforma utilizada, sus caracteristicas, los conceptos utilizados en la metodología aplicada en el análisis y diseño soportadas (como: definición de objetos y clases, herencia simple o múltiple, polimorfismo, etc.) y el grado de complejidad que su manejo implica. Como se puede inferir, todas estos puntos requieren de un estudio más profundo, dicho estudio se encuentra fuera del alcance de este trabajo de tesis.

BIBLIOGRAFIA

Ivar Jacobson
Maynus Christerson Patrik jonsson
Object Oriented Software Engineering
A use CASE Driven Approach
Gunnar Overgaard
Ed. ACM Press

Grady Booch
Object Oriented Desing with Aplications
The Benjamin/Cummings Publishing Company Inc.

Rebeca Wirfs-Brock,
Brian Wilkerson,
Lauren Wiener
Designing Object Oriented Software

Edward Yourdon,
Coad Peter
Object Oriented Analisys
Prentice Hall 1990
New Jersey.

Edward Yourdon,
Coad Peter
Object Oriented Desing
Prentice Hall 1990
New Jersey.

Roger S. Pressman
Ingeniería del Software, Un Enfoque Práctico
2a Edición.
Ed. Mc Graw Hill
Traducción José María Troya, Luis Hernández Yáñez
México 1989.

Roger S. Pressman
Ingeniería del Software, Un Enfoque Práctico
3a Edición.
Ed. Mc Graw Hill
Traducción José María Troya, Luis Hernández Yáñez
México 1993.

Robert G. Murdick con John C. Munson
Sistemas de Información Administrativa
2a. Edición
Ed. Prentice Hall
Traducción Rosa María Rosas Sánchez
México 1988.
Personal Computing México
Septiembre 94

Special Silver Anniversary Supplement
Celebrating The History of Object Oriented Technology
SIGS Publications
Peter Wegner, Brown University
"Dimensions of Object Oriented Modeling"
Computer
Octubre 1992
Pag. 12 a 20

Dave Thomas
"What's in an Object"
Byte
Marzo 1989
Pag. 231 a 240

Dra. Hanna Oktaba
"Programación Orientada a Objetos moda o realidad?"
Soluciones Avanzadas, Tecnologías de Información y Estrategias de Negocios.
Abril-Mayo 1993
Pag. 39 a 42

M.C. Gloria Quintanilla
"El Impacto en la Ingeniería de Software la Programación Orientada a Objetos"

Soluciones Avanzadas, Tecnologías de Información y Estrategías de Negocios.
Abril-Mayo 1993
Pag. 43 a 46

Vladimir Estivill Castro
"Un Panorama de las Bases de Datos Orientados a Objetos"
Soluciones Avanzadas, Tecnologías de Información y Estrategías de Negocios.
Junio 1994
Pag. 43 a 50

M.C. Amparo Gaona y M.C. Guadalupe Ibergüengoitia
"Técnica de Modelado de Objetos OMT"
Soluciones Avanzadas, Tecnologías de Información y Estrategías de Negocios.
Junio 1994
Pag. 56 a 59

Salvador de la Mora
"Los Objetos llegaron ya"
Personal Computing México Septiembre 1994
Septiembre 1994
Pag. 46 a 48