30
25

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"CAMPUS ARAGON"

ANALISIS Y DISEÑO ORIENTADO A OBJETOS PARA EL SISTEMA INTEGRAL DEL CONTROL DE PUESTOS Y RECURSOS HUMANOS EN LA SUBSECRETARIA DE HACIENDA Y CREDITO PUBLICO.

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

ALEJANDRO PALMA ACLIXQUEÑO

ASESOR: ING. ERNESTO PEÑALOZA ROMERO

MÉXICO

1999

A mamá, que me enseñó que la honestidad, la constancia y el trabajo son los ingredientes del éxito.

Con especial cariño a papá, mis hermanos y al pequeño Edgar.

Con respeto y profundo agradecimiento a mi casa; la Universidad.

"Este palacio es fábrica de los dioses, pensé primeramente. Exploré los inhabitados recintos y comenté: Los dioses que lo edificaron han muerto. Noté sus peculiaridades y dije: Los dioses que lo edificaron estaban locos."

J.L. Borges. El inmortal.

"...Instruyéndome no había conseguido más que descubrir mi propia ignorancia..."

René Descartes. El Discurso del método.

Índice

Capítulo I. La Ingeniería de Software

Capitulo II. Fase de Concepto del SICPRH

Capitulo III. Análisis Orientado a Objetos del SICPRH

Capitulo IV. Diseño Orientado a Objetos del SICPRH

Capítulo V. Implementación

Capítulo I
La Ingeniería de Software

I.1 La crisis del software

Desde mediados de la década pasada se ha evidenciado el hecho de que el software ha superado al hardware como la clave del éxito de muchos sistemas basados en computadoras; actualmente la competitividad entre las empresas requiere de información completa, veraz y disponible en cualquier momento.

Sin embargo, no siempre ha sido así: durante las tres primeras décadas de la era de la informática (1950 a 1980) el principal desafío en la industria de la computación consistía en el desarrollo del hardware, de tal forma que se redujera el costo de procesamiento y almacenamiento de información. El desarrollo del software de entonces estaba limitado por el poder del equipo de cómputo y el diseño de sistemas se hacía en función de optimizar tiempo de procesamiento y espacio en disco o en cintas. La mayoría del software se construía y era utilizado por la misma persona u organización; el "arte" de implementar sistemas consistía en pruebas de ensayo y error, de esta manera podíamos afirmar que el desarrollo del software se realizaba a través de estas únicas fases: codificación, ejecución y depuración.

Con el paso del tiempo la industria del hardware ha experimentado una tremenda evolución, esto ha hecho posible la aparición de equipo de cómputo más poderoso y barato, al mismo tiempo ha incrementado la demanda de productos de programación. No es exagerado decir que el crecimiento de la industria del hardware dio origen al software como industria.

Lamentablemente no podemos decir que la Industria del Software ha crecido en forma semejante a la del hardware. Toda una serie de problemas endémicos han impedido el desarrollo en esta área; estos problemas causan que los productos de software no sean entregados a tiempo, se excedan en el presupuesto, sean difíciles de mantener, y sobre todo, no satisfagan los requerimientos del cliente. Esto se ha visto agravado con la presencia continua de aspectos indeseables como altos costos de producción y mantenimiento, bajo rendimiento y bajo nivel de confiabilidad del software.

Mucha gente de esta industria ha caracterizado a los problemas asociados con el desarrollo del software como una crisis. Se ha argumentado que la construcción de nuevos sistemas que complazcan tanto al usuario como al cliente y que no contengan errores resulta un problema mayor. Este problema sin resolver ha sido denominado la "Crisis del Software".

Contrario a esta opinión, Pressman afirma que "La palabra crisis se podría definir como un punto decisivo en el curso de algo... Sin embargo, para el software no ha habido ningún punto decisivo, ningún momento crítico, solamente un lento cambio evolutivo. En la Industria del Software hemos tenido una crisis que ha estado con nosotros cerca de 30 años... Es bastante más preciso describir lo que hemos estado aguantando las tres últimas décadas como una aflicción crónica que como una crisis"[1].

Toda esta problemática dio origen a la Ingeniería del Software, la cual considera que se podría atenuar estos inconvenientes si pensamos que los sistemas de software son productos

de Ingeniería, de la misma forma como los aviones, edificios, carros, televisores o cualquier otro objeto que requiera un alto grado de habilidades para transformar materia prima en un producto útil. Es decir, el software debe ser construido empleando principios y metodologías de Ingeniería al igual que como se construye el hardware.

Así entonces, se ha definido a la Ingeniería del Software como "La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software"[2]. Es una disciplina que utiliza técnicas de Ingeniería y Administración para la fabricación y mantenimiento de productos de programación, de tal forma que éstos sean desarrollados y modificados a tiempo y dentro de un presupuesto definido.

1.2 Métodos, herramientas y técnicas de la Ingeniería de Software

Tratar de exponer los métodos, herramientas y técnicas de los que se vale la Ingeniería de software para el desarrollo de sistemas resulta una labor colosal y va más allá del objetivo del presente trabajo. Sin embargo, he intentado evidenciar la necesidad de la aplicación de principios y metodologías de Ingeniería en la construcción de sistemas de software. Por lo tanto, mencionaré brevemente algunos aspectos esenciales de la Ingeniería de Software, mismos de los que hecho mano en la construcción del Sistema Integral de Control de Puestos y Recursos Humanos (SICPRH).

---

[1] Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico.* 3a Edición. McGraw Hill, p.p.18

[2] *IEEE, Standard Glossary of Software Engineering Terminology.* Std. 610.12-1990. pp.67 presentado en *Software Engineering, IEEE Standards Collection.* Institute of Electrical and Electronics Engineers, Inc. Los Alamitos, California, Computer Society Press 1993

1.2.1 Metodologías

La Ingeniería de Software soslaya que los sistemas de programación han de ser considerados como productos de Ingeniería. Si esto es cierto, ¿Cómo se construye un producto de Ingeniería?, ¿Cuántas alternativas de construcción tenemos para realizar tal producto?, y si existen varias maneras de hacerlo, ¿Cuál es la mejor?.

Un producto de Ingeniería se erige a través de la aplicación de una metodología sistemática y rigurosa, esta metodología divide el desarrollo del producto en fases, en donde cada fase "construye" una parte del producto con la ayuda de métodos, técnicas y herramientas. Existen varias maneras para fabricar un producto, de acuerdo al número de metodologías que se apliquen, pero no existe alguna que sea mejor a las demás *bajo cualquier circunstancia*, aunque debemos mencionar, hay algunas que prevalecen sobre las otras porque generalmente son más adecuadas.

Así pues, para fabricar un producto de programación debemos aplicarle una metodología utilizando los métodos y técnicas afines.

Hagamos ciertas precisiones: un **método** es un proceso disciplinado para generar un conjunto de modelos que describen algunos aspectos del sistema a desarrollar usando una notación bien definida. Los métodos son guías generales que gobiernan la ejecución de ciertas actividades, constituyendo enfoques ordenados, rigurosos y sistemáticos. Son importantes por varias razones, primero porque introducen disciplina al desarrollo de sistemas de software complejos. También definen los productos que sirven como vehículos comunes para la comunicación entre miembros del equipo de trabajo. Además, los métodos definen los *milestones*[3] necesarios para la administración del proyecto, medición del progreso y manejo de riesgos.

[3] A lo largo del desarrollo de sistemas se identifican metas que diferencian el fin de una actividad y el comienzo de otra implicando la generación de un producto intermedio. Cuando cumplimos estas metas calendarizadas decimos que hemos alcanzado un *milestone*.

Las **técnicas** resultan de la composición de teorías, conocimientos, y en ocasiones de métodos, que unifican y construyen una vista muy bien delimitada de la realidad. Son más mecánicas que los métodos y tienen una aplicabilidad más limitada. Un conjunto consistente de técnicas da lugar a lo que se denomina una tecnología de software.

Por su parte, una **metodología** es una colección (conjunto) de métodos y técnicas aplicados a todo el ciclo de vida del sistema, éstos métodos están relacionados y unificados por un enfoque de solución. Una metodología indica la forma de abordar un problema (sistema) y de solucionarlo a lo largo de todo su ciclo de vida.

Puesto que las metodologías están compuestas por métodos y técnicas, éstos deben ofrecer una estructura abierta y una notación consistente, en el sentido que sean capaces de aceptar entradas de cualquier otro método.

Algunos métodos de diseño fueron propuestos durante los 70`s para modelar la creciente complejidad. El más importante fue el diseño estructurado. Este método fue influenciado notoriamente por la topología de los lenguajes tradicionales de la época, tales como FORTRAN y COBOL. En estos lenguajes, la unidad fundamental de descomposición es la subrutina o subprograma, y el programa resultante toma la forma de un árbol en la cual las subrutinas realizan su trabajo por llamadas a otros subprogramas. Esto es exactamente el enfoque del diseño estructurado: aplicamos una descomposición algorítmica para convertir un problema mayor en varios problemas menores.

Los métodos han evolucionado como respuesta al crecimiento en complejidad de los sistemas de software. A partir del abaratamiento en los precios del hardware y del creciente poder de procesamiento de los equipos de cómputo fue deseable y factible automatizar más aplicaciones de complejidad creciente. Los lenguajes de programación de alto nivel se convirtieron en herramientas importantes. Tales lenguajes incrementaron la productividad de los desarrolladores de software, presionándonos a crear sistemas de mayor complejidad.

Como lo mencionamos anteriormente, en Ingeniería no existe una metodología que nos garantice ser la mejor *bajo cualquier circunstancia*, y las metodologías que se basan en el diseño estructurado no son la excepción. El uso del diseño estructurado no ha cambiado, a pesar de que se han presentado numerosas dificultades. Se ha hecho notorio que "la programación estructurada parece caerse a pedazos cuando las aplicaciones exceden las 100,000 líneas de código"[4].

Desde la década pasada hemos notado que un impresionante número de autores ha intentado encontrar el mejor camino de solución en sistemas de software, inventando sus propios métodos y los más arriesgados, sus propias metodologías. Paradójicamente, el mundo de la informática se ha estado inundando de formas de solución y terminologías que crean gran confusión entre los analistas, diseñadores e implementadores de software. Más recientemente docenas de métodos de análisis y diseño han sido propuestos, la mayoría de ellos, para lidiar con los defectos del diseño estructurado. Estos métodos pueden ser catalogados dentro de estos tipos:

- Diseño estructurado

El diseño estructurado está ejemplificado en los trabajos de Yourdon y Constantine, Myers y Page Jones. Los fundamentos de este método se derivan de Wirth, Dahl, Dijkstra y Hoare, Probablemente la mayoría del software ha sido escrito usando estos métodos de diseño.

- Diseño de flujo de datos (*Data driven Design*)

Fue expuesto por Jackson y en los métodos de Warnier y Orr. En esos métodos la estructura del sistema de software se deriva de un mapeo de las entradas hasta convertirse en las salidas del sistema. El diseño de flujo de datos ha sido exitosamente aplicado a un reducido dominio, particularmente a los sistemas de información.

---

[4] Stein, J. *Object Oriented Programming and Database Design*. Marzo 1988, citado por Booch, Grady. *Object-Oriented Analisys and Design with Applications*. 2ª edición. p.p. 18

- La orientación a objetos

El análisis y diseño orientado a objetos descansa en la premisa de que debiéramos modelar sistemas de software como una colección de objetos cooperantes, involucrando objetos individuales como instancias de una clase con una determinada jerarquía.

I.2.2 Modelos de ciclo de vida de los sistemas

El concepto del ciclo de vida del sistema es fundamental en los métodos de Ingeniería de software. El propósito de una definición del ciclo de vida es entender el proceso de la Ingeniería de software sobre el desarrollo del sistema para que éste pueda ser monitoreado y controlado.

El ciclo de vida de un sistema de software se define como "el periodo de tiempo que comienza cuando un producto de software es concebido y termina cuando el software ya no está disponible para usarse. Típicamente, el ciclo de vida del software incluye una fase de concepto, una fase de requerimientos, fase de diseño, fase de implementación, fase de pruebas, fase de instalación, fase de operación y mantenimiento y, algunas veces, una fase de retiro... Estas fases pueden traslaparse o ser desarrolladas iterativamente"[5].

Diferentes modelos de ciclo de vida hacen hincapié en distintos aspectos del ciclo, pero ninguno es apropiado para todos los productos. Es esencial definir un modelo de ciclo de vida para cada proyecto de programación, puesto que permite clasificar y controlar las diferentes actividades necesarias para el desarrollo y mantenimiento del producto.

Existen muchos modelos de ciclo de vida de productos de software, sin embargo, estos caen básicamente en dos categorías:

---

[5] IEEE, *Standard Glossary of Software Engineering Terminology*, Std. 610.12-1993 p.p. 68. presentado en *Software Engineering, IEEE Standards Collection.* Institute of Electrical and Electronics Engineers, Inc. Los Alamitos, California, Computer Society Press, 1993

a) Modelo secuencial.

Los modelos secuenciales dividen el ciclo de vida del producto de programación en una serie de actividades sucesivas; cada fase requiere información de entrada, procesos y resultados, todos ellos bien definidos. Se necesitan recursos para terminar los procesos de cada fase, y cada una de ellas se efectúa mediante la aplicación de métodos explícitos, herramientas y técnicas. Dentro de los modelos secuenciales ocupa un lugar especial el modelo de cascada.

b) Modelo iterativo.

A diferencia del modelo de cascada, estos modelos no requieren de un completo conocimiento de los requerimientos del software para empezar el desarrollo del mismo. Al igual que los modelos secuenciales, están formados por un número definido de fases, pero permiten hacer ciclos repetitivos en el proceso de desarrollo hasta que se alcanza el producto final. Algunos modelos incluyen en el mantenimiento del producto las mismas fases repetitivas usadas para el desarrollo. Sin embargo todos los modelos iterativos ponen especial énfasis en la retroalimentación. Como ejemplos de estos modelos podemos mencionar al modelo espiral, modelo evolucionario, el modelo del eterno desarrollo, el enfoque de fases intercaladas, etc.

I.3 Administración de Proyectos de Software

Puesto que la tecnología de Ingeniería de software por sí sola es necesaria pero no suficiente para el éxito de los proyectos, ésta debe estar acompañada de la administración de Ingeniería en Software.

Al comenzar el presente trabajo hablamos de ciertos aspectos indeseables que se han presentado en el desarrollo e implementación de sistemas. Algunos de estos problemas son síntomas de una inadecuada administración de los proyectos de software:

- Sobrecostos
  - Del 50 al 100% es común.
- Retrasos en los tiempos de entrega
  - Retrasos del 20 al 50%
- Calidad inaceptable
  - Funcionalidad y desempeño inadecuado.
  - Inaceptables consecuencias causadas por los errores.

La Industria del Software está viciada de fallas en trazar, implementar y congelar los planes de desarrollo de software de acuerdo a los requerimientos del proyecto (*baselines*), fallas en la planeación de tareas, en calendarización, fallas en los presupuestos, en mecanismos de control y estado, mal manejo de la configuración del software, fallas o nulidad en el aseguramiento de la calidad, en la documentación, asignación inapropiada e inadecuada de personal (*staff*), etc.

¿Por qué es inadecuada, por no decir pobre, esta tarea de administración? La respuesta la podemos encontrar en la dificultad de administrar proyectos de software. Primeramente debemos mencionar que un producto de software es invisible; es una entidad lógica (en comparación a las entidades físicas que producen las otras Ingenierías), además, los requerimientos cambian frecuentemente. Si a esto aunamos el hecho de que el desarrollo de software es una actividad de equipo intensamente inteligente y desgastante, de que los expertos en la administración de proyectos de software son escasos y en que la demanda por sistemas de software más grandes y complejos se está acelerando, entonces podemos darnos cuenta de la dificultad con que nos encontramos.

Explotando el hecho de que la administración desempeña la misma función, independientemente de la posición en la organización o del tipo de compañía manejado, y puesto que la administración de proyectos de software es similar a la administración de cualquier otra Ingeniería, las técnicas clásicas de la administración de proyectos se han adaptado a los aspectos únicos de la Ingeniería de Software.

La tecnología de Ingeniería de Software se ha enriquecido con la disciplina de la Administración, de tal forma que los proyectos de software utilizan el proceso administrativo en esta forma básica:

- Planeación.

  Se determinan aspectos importantes del sistema, tales como actividades, tiempos para su realización y los recursos necesarios TTR (tareas-tiempos-recursos).

- Organización.

  Se conciben los roles necesarios para realizar las tareas y se adquiere el personal adecuado para cubrir estos perfiles.

- Dirección.

  Se guía al equipo de trabajo para que acaten sus roles y cumplan los objetivos.

- Control.

  Se monitorean las actividades de trabajo y se aplican las acciones correctivas.

La tabla de la siguiente página lista las principales tareas administrativas en el desarrollo de un proyecto de programación.

PASOS DEL PROCESO ADMTIVO.	TAREAS BÁSICAS	TAREAS ESPECÍFICAS
Planeación.	Definición del problema	Descripción de la situación actual.
	Requerimientos del proyecto	Determinación de objetivos y metas del proyecto. Adoptar políticas y normas.
	Proceso técnico (TTR)	Identificar y dividir las actividades de trabajo Desarrollar calendarios y presupuestos (estimación de costos). Desarrollar planes para las funciones de soporte (por ejemplo aseguramiento de la calidad, verificación y validación, etc.).
	Análisis de riesgos	Conducir actividades para detectar riesgos. Decidir cuáles riesgos son aceptables. Desarrollar un plan de contingencia.
Organización	Organización del proyecto.	Seleccionar una estructura organizacional (tipo proyecto. funcional o matricial) y una estructura de equipo (jerárquica, democrática, etc.). Establecer cualidades para las posiciones del proyecto. Definir responsabilidades y autoridades para las posiciones y tareas. Establecimiento de rutas de comunicación para los clientes, jefes inmediatos y consultores.
	Asignación (*staff*).	Llenar con el personal adecuado las posiciones (roles) definidos anteriormente.
Control.	Visibilidad del proyecto	Establecimiento de un sistema para mantener una visibilidad efectiva del proyecto a través de los *baselines*.
	Estado del proyecto	Establecimiento de un sistema de revisiones técnicas. Análisis del presupuesto, calendarios, calidad y productividad contra el plan.
	Control de cambios y acciones. correctivas	Determinar la necesidad de una acción correctiva. Iniciar las acciones correctivas necesarias. Rastrear las acciones correctivas hasta su culminación.
Dirección.	Dirección del proyecto.	Proveer un liderazgo efectivo. Motivación al personal. Resolución de conflictos entre miembros del equipo de trabajo.

De todas estas actividades cabe resaltar la planeación. La planeación es una fase administrativa que impacta a todo el desarrollo del proyecto; resulta de tal importancia que en algunas metodologías se le considera como la primera fase en el ciclo de vida del software.

Debo mencionar el hecho de que dos de los productos de la planeación se consideran como productos intermedios en el ciclo de vida del software, éstos son el Plan para la Administración de Proyectos de Software y la Especificación de Requerimientos.

El Plan para la Administración de Proyectos de Software "PAPS" (SPMP por las siglas en inglés de *Software Project Management Plan*) define el proceso administrativo y técnico necesario para satisfacer los requerimientos del proyecto, tales como relaciones organizacionales, roles y responsabilidades del equipo de desarrollo, métodos y metodología que se emplearán, mecanismos de control y reporte, estipulación de tareas, calendarios, recursos y presupuestos.

La especificación de Requerimientos establece las características del producto a desarrollar.

Los planes típicos en los proyectos de software son:

- Plan para la administración de proyectos de software.
- Plan para la verificación y validación.
- Plan para la prueba del software.
- Plan para la administración y configuración del software.
- Plan para el aseguramiento de la calidad.

I.4. Verificación y Validación

La Verificación y Validación (V&V) es una de las herramientas de la Ingeniería del Software que se aplican a todo el ciclo de vida del producto de programación. Los objetivos de las actividades de V&V son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software.

La V&V se ha definido como "el proceso de determinar si los requerimientos para un sistema son completos y correctos, si los productos de cada fase de desarrollo satisfacen los requerimientos o condiciones establecidas por la fase previa y si el sistema final cumple con los requerimientos especificados"[6]

Aunque la V&V tiene la misma función en cuanto a probar y detectar errores, cada término tiene su propio significado. La **verificación** se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica, mientras que la **validación** se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

Para definir de manera sencilla a estos conceptos, un buen número de autores de libros sobre Ingeniería de Software citan a Boehm, quien precisa estos términos a través de las siguientes preguntas:

"Verificación: ¿Estamos construyendo el producto correctamente?
Validación: ¿Estamos construyendo el producto correcto?"[7]

Las actividades más comunes en la V&V del software son:

- Recorridos e inspecciones.

[6] IEEE *Standard Glossary of Software Engineering Terminology*. Std. 610.12-1990. p.p. 81
[7] Citado en Fairley, Richard E. *Ingeniería de Software*.1ª ed. McGraw Hill. México, D.F., 1992. p.p.286

- Pruebas de unidad de código.
- Depuración.
- Pruebas de integración de módulos.
- Pruebas de aceptación del software.
- Ejecución simbólica del producto, etc.

I.5 Aseguramiento de la Calidad

Uno de los objetivos principales de la Ingeniería de Software es la construcción de "sistemas de calidad". Veamos qué significa esto; para Pressman, la calidad de un sistema está en función de la concordancia que guarde éste con tres aspectos básicos:

"Los *requisitos explícitos del software* son la base de las medidas de la calidad. La falta de concordancia con los requisitos significa una falta de calidad.

Los *estándares especificados* definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería de Software. Si no se siguen estos criterios, casi siempre habrá falta de calidad."[8]

Por último, "existe un conjunto de *requisitos implícitos* que a menudo no se mencionan (por ejemplo, el deseo de un fácil mantenimiento). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software queda en entredicho"[9]

[8] Pressman, Roger S. Op.cit. p.p..576
[9] Idem.

Otro punto de vista sostiene que la calidad de un sistema de software está en función de indicadores, tales como la portabilidad del producto, grado de confiabilidad, eficiencia, exactitud, niveles de error, solidez, corrección, etc.

Ambos puntos de vista son correctos y, lo que es mejor, son complementarios.

Por su parte, la herramienta que utiliza la Ingeniería de Software para crear sistemas de calidad es el Aseguramiento o Garantía de la Calidad (SQA por las siglas en inglés de *Software Quality Assurance*). Éste, al igual que la V&V, es aplicable a todo el ciclo de vida del software y opera sobre dos puntos básicos: el producto de programación y el proceso de desarrollo del producto.

El Aseguramiento de la Calidad es "un modelo planeado y sistemático de todas las acciones necesarias para proveer la confianza adecuada en cuanto a que el producto de programación satisface los requerimientos técnicos establecidos. Significa además un conjunto de actividades diseñadas para evaluar el proceso por el cual los productos son desarrollados o manufacturados"[10]

Las actividades del Aseguramiento de la Calidad incluyen:

- Realización de revisiones técnicas formales.
- Prueba del software.
- Ajuste a los estándares.
- Control de cambios.
- Registro y realización de informes.

Como habrá de observarse, la V&V del software comparte intereses con el Aseguramiento de la Calidad, sin embargo, una organización de software madura incluirá ambos procesos en el desarrollo de un proyecto.

---

[10] IEEE, *Standard Glossary of Software Engineering Terminology*. Std. 610.12-1990 p.p.60

I.6. La Orientación a Objetos en la Ingeniería de Software

Para abatir los efectos de la crisis del software, los primeros pasos en Ingeniería fueron aplicar conceptos de administración de proyectos tales como asignación de tareas, presupuestos, calendarización y medición de recursos. Se propusieron ciclos de vida de desarrollo de sistemas dividiendo los proyectos en fases. Se crearon metodologías para la implementación de sistemas y se obtuvieron productos intermedios. Se desarrollaron modelos de verificación y validación por fase y por proyecto entero, así como el aseguramiento de la calidad del producto.

No obstante lo anterior, el estado actual de la mayor parte de la Ingeniería de software está algo atrasado con respecto a las demás áreas de la Ingeniería. "Cuando la mayoría de los ingenieros completan su trabajo y éste se vende, esperamos que funcione de manera correcta. No esperamos que los motores de aviones exploten o que los edificios se derrumben, pero no nos sorprende que cierto software se comporte de manera extraña. La mayoría de los productos de hardware tienen garantía, pero la mayoría de los productos de software llevan una renuncia a la garantía"[11].

Actualmente el gran crecimiento en la velocidad y capacidad de las computadoras a muy bajo costo impulsa la demanda de software muy complejo e incrementa las expectativas del consumidor. Encontramos aplicaciones que exhiben una amplia gama de comportamiento, tales como sistemas en línea, aplicaciones que mantienen la integridad de cientos de miles de registros de información mientras actualizan datos o realizan búsquedas, sistemas que controlan entidades del mundo real, etc.

[11] Martin, James and Odell, James J. Análisis y Diseño Orientado a Objetos. Prentice Hall, México, D.F., 1994. p.p.33

Sistemas como éstos son de trascendencia considerable porque varios usuarios dependen de su correcto funcionamiento y generalmente todos sus detalles de implementación no pueden ser comprensibles por una sola persona.

La complejidad de los nuevos sistemas en algunos casos ha creado modelos monstruosos. los diagramas de diseño y análisis son difíciles de comprender. a menudo no se contemplan todas las posibilidades en los seguimientos de eventos y lo que es peor, los sistemas han crecido tanto en líneas de código que es muy difícil, y en algunos casos imposible, mantenerlos.

Lo anterior llevó a los ingenieros de software, diseñadores y analistas de sistemas a replantear totalmente la manera en cómo analizar, diseñar e implementar sistemas de una manera más natural y por consiguiente, más apegada a la realidad. Surgió así el Paradigma Orientado a Objetos.

Para poder comprender este paradigma es necesario remontarse a la época del surgimiento de la informática. Desde esos tiempos los programadores tenían que pensar como una computadora para poder resolver problemas de computación. Esta técnica parecía servir incluso cuando aprendíamos a programar, puesto que escribíamos programas muy sencillos. Sin embargo, en sistemas grandes, los ciclos y las ramificaciones nos proveen de una gran cantidad de combinación de caminos, de modo que no podemos pensar en todos ellos.

Para controlar la complejidad de los sistemas se comenzó a utilizar la programación estructurada. Ésta redujo el espagueti de código, pero la programación seguía basándose en una secuencia esperada de instrucciones de ejecución. El esfuerzo por diseñar y depurar programas, pensando en el orden que la computadora sigue para hacer las cosas desembocó en un software que no se entendía del todo. La aparición de los lenguajes declarativos para bases de datos agrava el problema, pues a menudo violan los principios de la programación estructurada, insertando saltos en el flujo natural del código.

Christopher Hoare, catedrático de computación en la Universidad de Oxford, comenta acerca del diseño estructural tradicional: "El intento de construir una disciplina de Ingeniería de Software sobre bases tan imperfectas como éstas está condenado al fracaso; como si se quisiera basar la Ingeniería Química en la teoría del Flogisto, o la astronomía en la hipótesis de la Tierra plana"[12]

El paradigma orientado a objetos es una nueva forma de conceptualizar la realidad. Las técnicas orientada a objetos permiten que el software se construya a partir de objetos de comportamiento específico. Los propios objetos se pueden construir a partir de otros, que a su vez pueden estar formados por otros objetos. Esto nos recuerda una maquinaria compleja, construida por partes complejas, subpartes, sub-subpartes, etc.

El mundo orientado a objetos tiene una mayor disciplina que el de las técnicas estructuradas convencionales. Los defensores de las técnicas orientadas a Objetos consideran que esto nos llevará a un mundo de clases reutilizables, donde la mayor parte del proceso de construcción de software consistirá en el ensamblaje de clases ya existentes y probadas. Sostienen que "las técnicas Orientadas a Objetos ligadas a las herramientas CASE con un generador de códigos y un depósito (también orientado a objetos) constituyen el mejor camino conocido para construir una verdadera Ingeniería de Software." [13].

I.6.1 Fundamentos del paradigma de la Orientación a Objetos

El paradigma de la orientación a objetos (OO) descansa sobre los conceptos que se definen abajo. Los defensores del enfoque OO argumentan que este proceso es "natural"

[12] Hoare, C.A.R., *The Engineering of Software: A Startling Contradiction.* Computer Bulletin, Diciembre de 1975, citado en Martin, James. op. cit. p.p. 34
[13] Martin, James. op.cit. p.p.35

puesto que los términos que definen la base del software OO son similares a los que forman la base de todos los seres vivos.

OBJETOS

Un objeto es cualquier cosa con límites o fronteras conceptuales bien definidas; un objeto tiene ciertos atributos (datos) y comportamiento (métodos o funciones). Los objetos pueden ser reales o abstractos, por ejemplo, una factura, una organización, una figura en un programa de dibujo, un mecanismo en un dispositivo de robótica, una pantalla con la que interactúa un usuario, etc.

El comportamiento de un objeto denota la forma en cómo actúa y reacciona en términos de sus cambios de **estado** (cambio en los valores de sus atributos) y paso de mensajes. Una **operación** o **método** es cualquier acción que el objeto realiza sobre sí mismo; un *mensaje* consiste en una operación que un objeto desarrolla para otro a través de una petición o solicitud.

El paradigma OO intenta modelar la realidad como un conjunto de objetos interactuando entre sí en un ambiente dado (escenario). Las técnicas OO permiten que el software se construya a partir de objetos de comportamiento específico.

ENCAPSULACION

He mencionado que un objeto posee atributos (datos) y comportamiento (métodos), el empaque conjunto de datos y métodos se llama **encapsulado**. El objeto esconde sus datos de los demás objetos y permite el acceso a los datos mediante sus propios métodos. Esto recibe el nombre de **ocultamiento de la información**.

Si un objeto oculta su información, ¿Cómo es posible para otros objetos comunicarse o realizar una petición a éste?

Los objetos están compuestos por una **interface** y una **implementación**. La interface de un objeto revela su vista externa: los otros objetos pueden comunicarse y solicitar operaciones a nuestro objeto a través de ella. La implementación constituye la vista interna de nuestro objeto; oculta al mundo externo los detalles de cómo son realizadas las operaciones propias. De esta manera, un objeto cliente conoce las operaciones que puede solicitar a un objeto servidor pero desconoce los detalles de cómo se lleva a cabo dichos métodos y menos aún, puede modificar tales estructuras (operaciones y/o datos del objeto servidor).

CLASES

"Una clase representa un conjunto de objetos que comparten una estructura y comportamiento común."[14] Mientras que un objeto representa una entidad concreta con existencia en tiempo y espacio, una clase representa solo una abstracción, la *esencia* del objeto.

En el mundo real nosotros vemos objetos; por ejemplo, para la S.H.C.P. es importante que IBM, HP, Compaq, Juan, José y María paguen impuestos. Estos objetos tan diferentes tienen una estructura y comportamientos comunes reunidos en la clase contribuyentes.

HERENCIA

Uno de los puntos fuertes del enfoque OO descansa en que una clase puede especializarse en clases de menor nivel, de la misma forma que un objeto puede tener subtipos. En el ejemplo anterior la clase contribuyentes tiene la subclase contribuyentes morales y contribuyentes físicos. María, Juan y José son instancias (objetos) de la subclase contribuyentes físicos mientras que IBM, HP y Compaq son instancias de la subclase contribuyentes morales.

---

[14] Booch, Grady. *Object-Oriented Analysis and Design with applications*. 2a. ed. Benjamin Cummings Publishing Co.,U.S.A. 1994., p.p. 103

Existe una jerarquía de tipos, subtipos, sub-subtipos, etc. Para conseguir esto, una subclase hereda propiedades de su clase padre. En nuestro ejemplo las subclases contribuyentes morales y contribuyentes físicos heredan características de la clase contribuyentes, tales como razón, base gravable, adeudos, etc.

La herencia es una relación entre clases u objetos donde una de éstas comparte la estructura (datos) y/o comportamientos (métodos) definida en otra (herencia simple) u otras clases (herencia múltiple). La herencia nos permite jerarquizar nuestras abstracciones y ordenar la complejidad.

TIPIFICACION

Es el enforzamiento que hace una clase para evitar que objetos de diferentes tipos puedan ser invocados en forma indebida. La clase hace un chequeo de tipos de tal forma que sólo el objeto del tipo correcto sea activado.

Para esclarecer este concepto Booch propone un ejemplo claro utilizando unidades físicas, "cuando dividimos distancia sobre tiempo esperamos algún valor que denote velocidad, no peso. Similarmente, multiplicar temperatura por una unidad de fuerza carece de sentido, pero multiplicar masa por fuerza si lo tiene. Ambos son ejemplos de tipificación fuerte, donde las reglas de nuestro dominio enforzan combinaciones legales de abstracción"[15]

Si los tipos de objetos son fijados en tiempos de compilación tenemos una **tipificación estática** (*static binding*). Si los tipos de objetos son conocidos hasta tiempos de ejecución del programa decimos que estamos ante una *tipificación dinámica* (*dinamic binding*).

[15] Booch, Grady. op. cit. p.p. 66

POLIMORFISMO

Permite que un nombre sencillo (como una declaración de una variable) pueda denotar objetos de diferentes clases que son relacionados por una superclase común. Cualquier objeto invocado con ese nombre está habilitado para responder a un conjunto de operaciones en diferentes formas.

CONCURRENCIA

Es la propiedad que distingue a un objeto activo de uno que no lo está. Un sistema concurrente es aquel que permite a varios objetos actuar al mismo tiempo.

PERSISTENCIA

Es la propiedad de un objeto que le permite existir aún cuando haya terminado el proceso que lo originó.

I.6.2 La Metodología de Booch

El análisis y alguna parte del diseño del SICPRH sigue los métodos descritos por Grady Booch en su más reciente libro *Object Oriented Analisys and Design with Applications* (1994), debido a esto, describiré brevemente algunos aspectos de esta metodología.

Como hemos mencionado, en Ingeniería de Software no existe algún método que nos asegure que su estricto seguimiento nos permitirá realizar un análisis óptimo y/o un diseño robusto *bajo cualquier circunstancia*. Las técnicas OO no son la excepción, aún más, no son tan específicas como las técnicas estructuradas. "El proceso de análisis y diseño OO no puede ser descrito como recetas de cocina"[16]. Sin embargo, los métodos de análisis

[16] Booch, Grady. op. cit., p.p. 229

orientado a objetos (AOO) y diseño orientado a objetos (DOO) están bien definidos y se enriquecen con las heurísticas y la experiencia del equipo de desarrollo.

Según Booch, un sistema OO debe tomar ventaja de las virtudes de los sistemas con ciclos de vida bien definidos y de aquellos que no los tienen.

En esas organizaciones donde no hay lugar para ciclos de vida bien definidos habrá lugar para la anarquía; a través del trabajo duro de algunos elementos del equipo de desarrollo se podría obtener algo de valor, sin embargo el producto sería impredecible, no habría una calendarización confiable y ciertamente no habría calidad. En el otro extremo no habría anarquía, pero las rígidas políticas y estándares para el desarrollo del software aniquilaría la creatividad del equipo de trabajo. En el primer caso sobra creatividad pero no hay control; en el segundo sobra control pero no hay creatividad.

Indudablemente no hay proceso perfecto que una estas dos situaciones, no obstante, podemos conciliar la necesidad de creatividad e innovación con prácticas, estándares y políticas para controlar el ciclo de vida del sistema a través del ejercicio de lo que Booch llama micro y macroproceso de desarrollo.

"El **microproceso** está relacionado con el ciclo de vida en espiral de Boehm y sirve para estructurar un enfoque de desarrollo iterativo e incremental. El macroproceso de desarrollo está ligado fuertemente con el ciclo de vida en cascada y es utilizado para controlar las actividades del microproceso"[17]

El microproceso de desarrollo aporta gran flexibilidad al sistema, es un proceso incremental en diferentes niveles de abstracción y refinamiento. Representa las actividades diarias de un desarrollador o un grupo pequeño de éstos aplicándose a todo el ciclo de vida del proyecto. La figura 1.1 muestra las actividades que lo conforman.

[17] Booch, Grady, op. cit., p.p. 234.

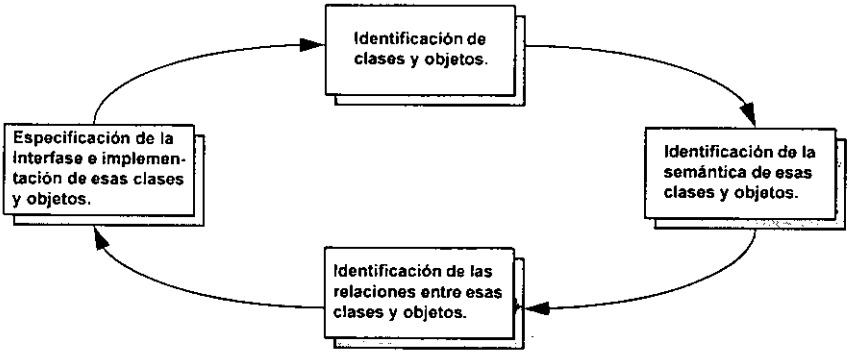


Fig. 1.1

El **macroproceso** implica un número de productos y actividades mensurables que permiten a un equipo de desarrolladores estimar riesgos y realizar correcciones al microproceso. El macroproceso incluye las actividades del equipo de desarrollo completo, en la escala de semanas a meses. Tiene un enfoque sobre los riesgos y la visión arquitectónica, es decir, sobre los dos elementos que tienen el más alto impacto en la calendarización, calidad y entrega del sistema. La figura 1.2 muestra las fases que componen el macroproceso de desarrollo de un sistema orientado a objetos.

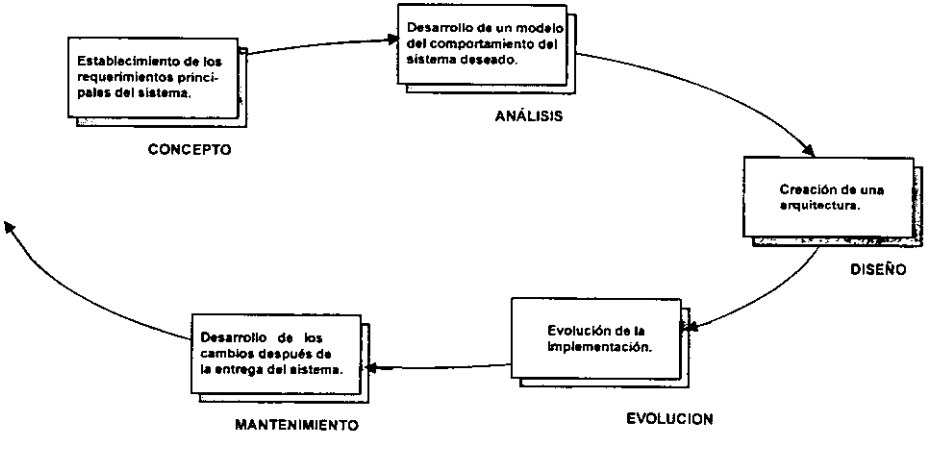


Fig. 1.2

Para Booch un sistema OO intenta hallar un equilibrio entre las actividades de un microproceso informal y altamente creativo combinándolas con un macroproceso altamente formal y con funciones de control. Booch aclara que la filosofía básica del macroproceso es de naturaleza incremental, al igual que el microproceso, sin embargo, reconoce la relación del primero con los modelos en cascada y aún más, señala que "una de las características de una organización de desarrollo madura es saber cuándo romper las reglas"[18].

I.7 Particularidades del SICPRH

La Ingeniería de Software no es una teoría rígida, al contrario, es una disciplina flexible compuesta de muchos enfoques y métodos adaptables a casi cualquier situación. En este sentido es notable la integración del paradigma de la orientación a objetos a las técnicas y herramientas de la Ingeniería de Software. A propósito de esto podemos mencionar el hecho de que a principios de la presente década aparecieron numerosas metodologías orientadas a objetos (OO) para la implementación de sistemas, los trabajos de Grady Booch, Coad y Yourdon, Rumbaugh, Shaler y Mellor, etc. son sólo algunos ejemplos de esto.

La intención original del presente trabajo, además de la implementación de un sistema real, es mostrar el proceso de fabricación de un sistema siguiendo una metodología orientada a objetos integrada en un proceso de Ingeniería de Software.

¿Por qué orientada a objetos? Por dos razones; la primera radica en el nivel de complejidad del sistema. El SICPRH es de una complejidad media, abarca las cuatro subdirecciones que conforman la Dirección Técnica Operativa, con esto quiero decir que un buen número de sistemas que no pertenecen a la Subdirección de Recursos Humanos, hacen

[18] Booch, Grady. Op. cit. p.p. 250

uso de la Plantilla de personal. Además hay tal cantidad de detalles y comportamientos del sistema deseado que resulta un tanto difícil para una sola persona pensar en todos ellos.

La segunda razón consiste en la necesidad de un diseño altamente flexible. A menudo se necesitan cambios en la denominación de plazas, en los atributos de éstas, en la creación, eliminación y/o reacomodo de áreas presupuestales, subdirecciones, departamentos y oficinas. La emisión de una gran cantidad y variedad de reportes exige un diseño robusto, pues en ocasiones se solicitan listados que son difíciles de realizar porque la estructura de los datos es rígida.

Para dominar la complejidad del sistema utilizo la OO, mi propósito es sacarle el máximo provecho a la forma "natural" de analizar y modelar sistemas orientados a objetos, además de obtener una estructura que me permita realizar cambios de la manera menos difícil.

Como mencioné anteriormente, mi idea original era implementar un sistema OO, lamentablemente la tecnología de las bases de datos orientadas a objetos (DBOO) no ha madurado aún, actualmente es muy difícil encontrar en México sistemas manejadores de bases de datos orientadas a objetos (ODMBS) comerciales.

Esta es la razón por la cual el SICPRH es un sistema híbrido, trato de aprovechar el poder del análisis y diseño OO, la madurez y uso de los sistemas manejadores de bases de datos relacionales (RDBMS) y el hecho de que es posible la migración de un análisis y diseño OO hacia el diseño de un sistema de bases de datos relacional. Esta es la estrategia en el ciclo de vida del SICPRH.

Las etapas OO del SICPRH siguen los métodos de Grady Booch, esto se debe a que, a diferencia de algunas metodologías como GOOD[19] y HOOD[20], la metodología de Booch es independiente de los lenguajes de programación.[21]. El trabajo de Booch es ampliamente conocido en el medio de las metodologías OO, aporta una gran cantidad de consejos y

[19] General Object-Oriented Design desarrollado por Seidewitz y Stark para la NASA.
[20] Hierarchical Object-Oriented Design, desarrollado por la Agencia Espacial Europea.

heurísticas para identificar clases, objetos, métodos y mecanismos de activación, así como para realizar un análisis de calidad y un diseño robusto. Además la notación que utiliza parece completa.

El ciclo de vida del SICPRH, equivalente al macroproceso de Booch, es el siguiente:

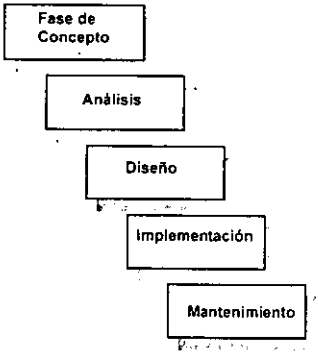


Fig. 1.3

Como se puede notar es un modelo en cascada, pero es similar a las fases del macroproceso. El SICPRH tiene una fase de concepto, una fase de AOO y DOO donde se aplica el microproceso. La fase de implementación es diferente al modelo de Booch y se realizará en un SMBD relacional. La fase de mantenimiento implica retomar el modelo del ciclo de vida del SICPRH empezando desde la primera fase.

El contenido del presente trabajo documentará la manera en cómo se realiza cada fase. Se verá que "no hay nada inherentemente orientado a objetos en la fase de concepto"[22], se aplicarán los métodos de Booch para el análisis y alguna parte del diseño, se detallará la migración de una estructura OO a un diseño de bases de datos relacional, por último, describiremos la fase de implementación del SICPRH.

---

[21] GOOD y HOOD están muy asociados a ADA.
[22] Booch, Grady. Op.cit. pp. 251

Capítulo II

FASE DE CONCEPTO DEL SICPRH

II.1. Introducción

El modelo de cascada para ciclos de vida de sistemas de software, unido al análisis y diseño estructurado ha sido muy popular. "Diferentes" modelos de cascada han sido usados en la construcción de sistemas, la mayoría de las veces comenzaban con una fase de definición del sistema, pasando por planeación, análisis, diseño, implementación y mantenimiento. En otras se empezaba por la planeación y se seguía con un análisis de requerimientos, diseño arquitectónico, diseño detallado, instrumentación, pruebas de software, etc. Sin embargo, de uno de estos modelos a otro existían fases con diferentes nombres, pero con el mismo significado, las mismas actividades, los mismos productos y las mismas funciones, de tal forma que podíamos asegurar que todos los modelos de cascada con métodos estructurados eran una misma cosa.

Es de todos conocido el hecho de que las técnicas y métodos actuales descansan en la teoría precedente, de tal forma que es necesario retomar la manera en como se comenzaba un proyecto de software bajo el enfoque estructurado. Según Fairley[1], un modelo de cascada de cinco fases era suficiente para describir un proyecto de software.

[1] Fairley, Richard. op. cit., p.p.40

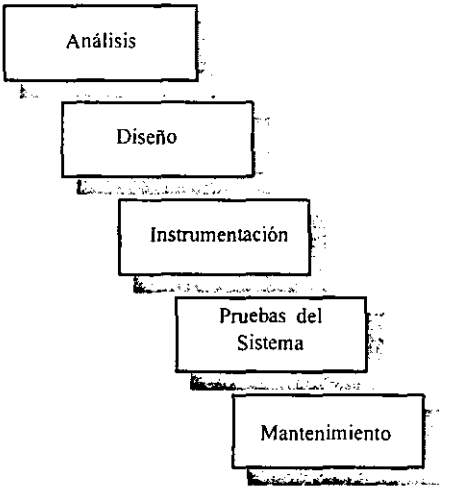


Fig. 2.1

El sistema se construye empezando con el análisis. Esta fase está dividida en dos subfases; planeación y definición de los requisitos del sistema.

Las actividades principales durante la **planeación** incluyen la comprensión del problema del cliente, estudio de factibilidad, estimaciones de tiempos, tareas y recursos, determinación de los criterios de aceptación, etc., siendo los productos de esta fase los documentos de definición del sistema, plan de proyecto[2] y manual de usuario preliminar.

La subfase de **Definición de los requisitos del sistema** consiste en la especificación técnica de las restricciones y características que debe cumplir el producto. El objetivo de esta subfase es especificar total y consistentemente los requerimientos técnicos del producto de una manera concisa y sin ambigüedades. El producto de esta subfase es el documento de especificación de requisitos para la producción del software.

[2] El plan de proyecto consiste en lo que hemos llamado PAPS (Plan de Administración de Proyectos de software), veáse capítulo I.

Como se puede notar para este enfoque, el análisis de un sistema involucra muchas actividades, casi todas tan importantes como laboriosas. Cabe mencionar el hecho de que en esta fase se conjuntan dos tareas diferentes; una de naturaleza administrativa (planeación) y la otra de naturaleza analítica que no va más allá de la detección de requerimientos.

En 1986 la IEEE (*Institute of Electical and Electronic Engineers*) aprobó el estándar 1012, relativo al *Estándar para la Verificación y Validación de Software*, en el cual se definía una fase de concepto[3] que tenía como objetivo organizar de una manera más eficiente el modelo de ciclo de vida de un sistema, a la vez que liberaba de actividades a la fase de análisis.

II.2. Definición de la Fase de Concepto.

La fase de Concepto es la "fase inicial en el desarrollo de un proyecto de software en el cual las necesidades del usuario son descritas y evaluadas a través de documentación (por ejemplo, listas de necesidades, estudios de factibilidad, definición del sistema, regulaciones, procedimientos o políticas relevantes al proyecto)"[4]

A juzgar por Booch, un sistema OO comienza con una fase de concepto (o conceptualización como él la llama), cuyo propósito fundamental es el establecimiento de los requerimientos principales para el sistema. El análisis OO deja de ser un análisis de requerimientos, gracias a la fase de concepto, para convertirse en una fase que intenta describir el sistema en cuestión a través de modelos de su estructura y su comportamiento.

[3] Hago uso indistinto de la Fase de Concepto y la Fase de Concepto de Operaciones.
[4] IEEE Std 1012-1986, *Standard for Software Verification and Validation.*

Las actividades propias de la fase de concepto son de naturaleza intensamente creativa e informal; describen los ambientes operacionales del sistema actual en funcionamiento y mantienen un registro de ideas y soluciones nuevas para el sistema a desarrollar. Por tal motivo es necesario una comunicación efectiva que involucre todas las partes interesadas en la construcción del software.

El producto primordial de la fase de concepto es el Documento de Concepto de Operaciones (ConOps).

A pesar de que la IEEE introdujo la Fase de Concepto desde 1986 y 1987[5], hasta la fecha no ha emitido estándar alguno sobre este punto. Los estándares disponibles pertenecen a oficinas gubernamentales de los Estados Unidos de América, y por ende, no han tenido una gran aceptación fuera de ese país.

En Junio de 1985 el Departamento de la Defensa publicó el documento DI-MCCR-80023 el cual describe un estándar para la elaboración de un Documento de Concepto de Operaciones (OCD, por las siglas de Operational Concept Document) también conocido como ConOps. En Febrero de 1989 la Administracion Nacional para la Aeronáutica y el Espacio (NASA por las siglas en ingés de National Aeronautics and Space Administration) incluyó un estándar más completo para el producto de la fase de concepto[6].

Actualmente no hay un estándar reconocido para la elaboración del ConOps. Sin embargo, hay un grupo de planeación y desarrollo del Comité de Estándares para Ingeniería de Software que trabaja sobre este punto.[7]

---

[5] IEEE Std 1002-1987, *Standard Taxonomy for Software Engineering Standards.*

[6] *Product Specification Documentation Standard and Data Item Descripcion* (DID) SMAP-DID-P100. Publicado en Thayer, Richard H & Dorfman, Merlin. *Standards, Guidelines and examples on System Software Requirement Engineering*. IEEE., Los Alamitos, California, Computer Society Press.

[7] P1362 Standard for Information Technology - System Definition - Concept of Operations. Richard H. Thayer, chair. Tomado de Overview of the software Engineering Standards Committee of the IEEE Computer Society. Version 3.2, 15 September 1997.

Aunque no hay estándares para la elaboración del ConOps, existen lineamientos y guías para su realización. Las más notables pertenecen a Richard Thayer, catedrático de la Universidad Estatal de California en Sacramento, miembro de la Sociedad de Computación de la IEEE y presidente del grupo de trabajo del estándar para el Concepto de Operaciones. Según él, un índice de cualquier ConOps debería tener los sigs. puntos:

1. Alcance.
1.1. Identificación.
1.2. Visión General del documento.
1.3. Visión General del sistema.

.2. Documentos referenciados.

3. Sistema y/o situación actual
3.1. Antecedentes y objetivos.
3.2. Descripción de la situación actual.
3.3. Políticas operacionales y restricciones del sistema.
3.4. Clases de usuarios.
3.4.1. Perfiles de usuarios.
3.4.2. Interacciones entre las clases de usuarios.
3.4.3. Otro personal involucrado.
3.5. Ambiente de soporte para el sistema actual.

4. Justificación de los cambios propuestos y/o nuevas características.

5. Concepto de operaciones para el sistema propuesto

6. Escenarios operacionales para el sistema propuesto.

7. Resumen de impactos.

8. Notas.
Apéndices.
Glosario.

Thayer señala que el propósito de la Fase de Concepto de Operaciones es el proveer un puente entre las necesidades del usuario y los documentos de los requerimientos técnicos de los desarrolladores[8]. La Fase de Concepto establece la razón y la naturaleza del sistema que será desarrollado, enumera los propósitos y los riesgos del desarrollo propuesto, esto es realiza un análisis conceptual del sistema.

Thayer retoma los estándares que existen sobre el ConOps (SMAP-DID-P100 y DI-MCCR-80023 el primero perteneciente a la NASA y el segundo al Departamento de la Defensa de los Estado Unidos) y recomienda que éste debe estar escrito empleando el lenguaje cotidiano del usuario y su formato. Un documento ConOps debe estar escrito en prosa y debería hacer uso de las formas visuales (diagramas, ilustraciones, gráficas, etc.) cual sea posible.

Puesto que Booch acepta que en la "Fase de Concepto de un sistema OO no hay nada inherentemente orientado a objeto"[9], pretendo elaborar un ConOps tomando en cuenta los lineamientos para la elaboración del ConOps recomendado por Thayer.

Lo que resta del presente capítulo estará dedicado a la fase de Concepto para el SICPRH.

II.3 CONOPS para el SICPRH

Debido a limitantes de espacio no me es posible incluir todo el Documento del Concepto de Operaciones en el presente capítulo, en todo caso se incluiría como un bonito apéndice de más de 100 hojas. Sin embargo, presento sólo un capítulo del ConOps; el correspondiente a la descripción del Sistema y/o situación actual.

[8] Thayer, Richard H. and Dorfman, Merlin. *Software Engineering*. Computer Society, IEEE, 1997
[9] Booch, Grady, op.cit. p.p. 251

II.3.1 Sistema y/o situación actual

Lo que se presentará a continuación es de crucial importancia, nos permitirá identificar el funcionamiento operacional actual y delimitar todo lo que se encuentre dentro del dominio de la aplicación por implementar. El producto de esta actividad facilitará el AOO posterior.

II.3.1.1 Antecedentes, objetivos y alcance

La Secretaria de Hacienda y Crédito Público es una entidad dependiente del Ejecutivo Federal del más alto nivel. Entre algunas de sus atribuciones principales podemos mencionar:

- Proponer, dirigir y controlar la política del Gobierno Federal en materia financiera, fiscal, de gasto público, crediticia, bancaria, monetaria, de divisas, de precios y tarifas de bienes y servicios del sector público, de estadística, geografía e informática.
- Controlar, vigilar y asegurar el cumplimiento de las disposiciones fiscales, en el cobro de impuestos, contribuciones, derechos, productos y aprovechamientos federales en los términos de las leyes aplicables.
- Contratar créditos internos y externos a cargo del Gobierno Federal, nombrar representantes para la negociación de los mismos y para la suscripción de los documentos relativos, cuando así proceda.
- Establecer relaciones y mecanismos de coordinación que permitan obtener la congruencia global de la Administración Pública Paraestatal con el Sistema Nacional de Planeación y con los lineamientos generales en materia de financiamiento.
- Dirigir y coordinar la elaboración e integración del Plan Nacional de Desarrollo y los Programas Regionales y Especiales que le encomiende el Ejecutivo Federal.

- Planear, coordinar y evaluar el Sistema Bancario Mexicano, respecto de la Banca de Desarrollo y las Instituciones de Banca Múltiple en las que el Gobierno Federal tenga el control por su participación accionaria.
- Ejercer las atribuciones que le señalen las leyes en lo referente a banca múltiple, seguros y fianzas, valores, organizaciones auxiliares de crédito, sociedades mutualistas de seguros y casas de cambio.

Para el cabal cumplimiento de sus actividades, la Secretaría de Hacienda y Crédito Público posee la estructura que se muestra en la figura 2.2

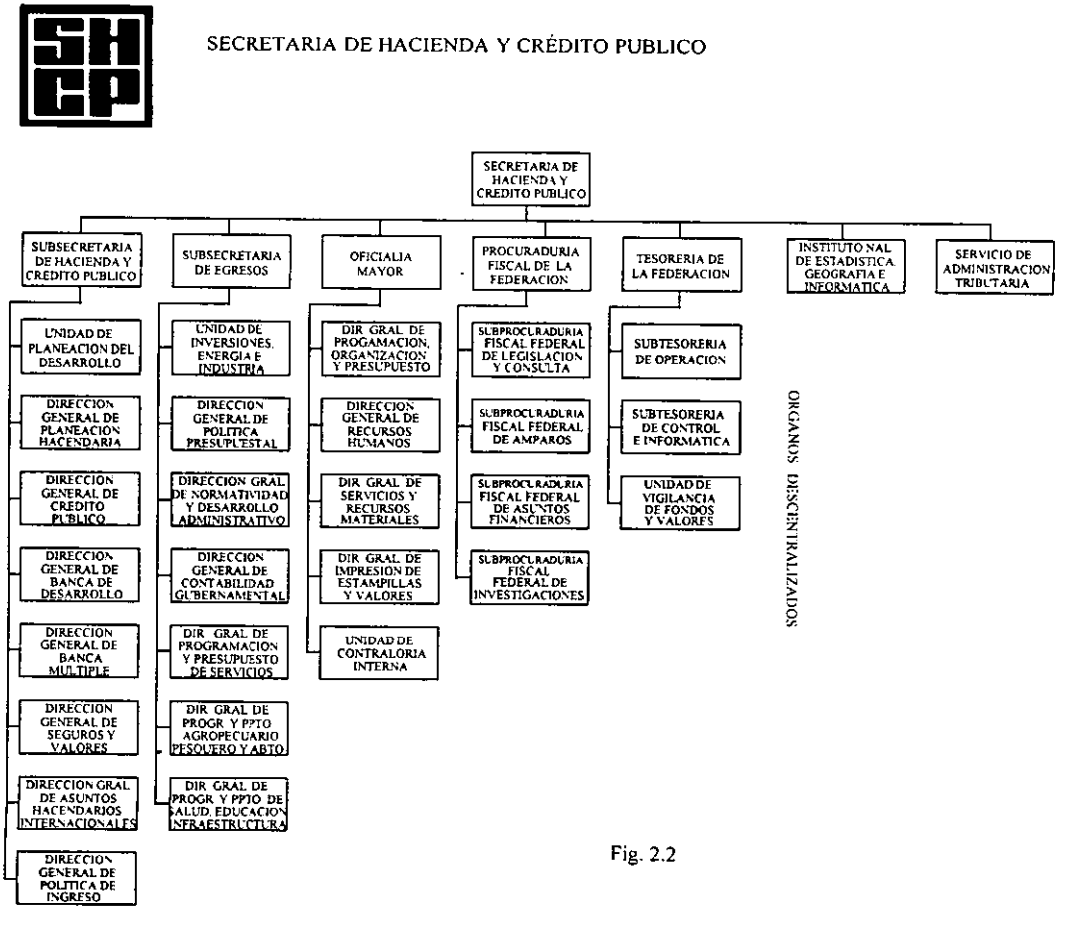


Fig. 2.2

La Subsecretaría de Hacienda y Crédito Público está encargada de:

- Proponer estrategias en materia financiera, económica, social, fiscal, de divisas y de precios y tarifas de bienes y servicios del sector público.
- Proponer políticas de desarrollo y la elaboración de estudios para conocer la situación económica y social, nacional e internacional, con el propósito de enriquecer y aportar elementos que le correspondan a la Secretaría y a sus entidades coordinadas, en la formulación del Plan Nacional de Desarrollo y de los programas sectoriales, regionales y especiales que se determinen, así como controlar y evaluar el desarrollo y actualización de los mismos.
- Manejar la deuda pública del Gobierno Federal y del Distrito Federal, asi como vigilar la aplicación y manejo de los fondos provenientes del crédito público.
- Coordinar la relación del gobierno Federal y evaluar la contratación de financiamiento con el Banco Internacional de Reconstrucción y Fomento, el Banco Interamericano de Desarrollo y los organismos de fondos similares y, conjuntamente con el Banco de México coordinar la relación con el Fondo Monetario Internacional.
- Proponer la política hacendaria de la Secretaría con el exterior, incluida la relativa a relaciones bilaterales.
- Dirigir la preparación del informe presidencial en las materias de competencia de la Secretaría y la preparación del Informe de Labores de la misma.
- Administrar los recursos humanos, financieros y materiales asignados a la Subsecretaría de Hacienda y Crédito Público, de acuerdo con los lineamientos fijados y de conformidad con las disposiciones emitidas por la Oficialía Mayor.

El 26 de Agosto de 1983 se publicó el Reglamento Interior de la Secretaría de Hacienda y Crédito Público, en el que se establece la creación de las Coordinaciones de Administración, así como la especificación de las áreas que contarían con el apoyo de esta unidad administrativa, entre las que se incluye a la Subsecretaría de Hacienda y Crédito Público.

El 4 de Enero de 1990 se reforma dicho Reglamento, modificándose el nombre de Coordinaciones Administrativas por el de Direcciones de Técnica Operativa. además, se ratifica la Dirección de Técnica Operativa de la Subsecretaría de Hacienda y Crédito Público.

En cuanto a las atribuciones y funciones de las Direcciones de Técnica Operativa, el artículo 92 del actual Reglamento Interno señala que "Las Subsecretarías de Hacienda y Crédito Público, la de Ingresos (ahora Sistema de Administración Tributaria), la de Egresos, la Oficialía Mayor, la Procuraduría Fiscal de la Federación y la Tesorería de la Federación contarán cada una de ellas, con una Dirección de Técnica Operativa, a las que compete:

i. Integrar y consolidar los programas y presupuestos de la unidad administrativa de que dependa, de acuerdo a los lineamientos que fije la Oficialía Mayor;
ii. Atender las necesidades en materia de personal, recursos materiales y presupuestales, de acuerdo a los lineamientos que fije la Oficialía Mayor;
iii. Auxiliar a las autoridades competentes de la Secretaría, en la ejecución del presupuesto autorizado a la unidad administrativa de que dependa, así como llevar el control de disponibilidades de los recursos financieros asignados a la misma;
iv. Controlar la operación administrativa y los procedimientos de trabajo de las unidades administrativas de su área y, en su caso, servir de enlace entre las unidades administrativas centrales y las regionales correspondientes y auxiliarlas en el eficaz desempeño de sus funciones;
v. Coordinar las actividades relativas a la selección de aspirantes, a la capacitación y desarrollo de personal que presta sus servicios en la unidad administrativa;
vi. Adscribir al personal de la unidad administrativa de la que dependa la Dirección Técnica Administrativa, y cambiarlo de adscripción cuando el cambio se realice a cualquiera de las unidades de aquella;

vii. Integrar y consolidar la evaluación de la unidad administrativa de que dependa, en la ejecución de los programas a su cargo;

viii. Las demás que le encomiende el titular de la unidad administrativa de que dependa y las que sean necesarias para el adecuado funcionamiento de la misma. "[10]

Para el cumplimiento de las funciones que la ley señala, la Dirección de Técnica Operativa de la Subsecretaría de Hacienda y Crédito Público posee la sig. estructura orgánica:

Fig. 2.3

El fundamento esencial de la Subdirección de Recursos Humanos consiste en coordinar la administración de los recursos humanos de la Subsecretaría del Ramo y de las unidades administrativas dependientes para el cumplimiento de sus programas asignados, conforme a los lineamientos emitidos por la Oficialía Mayor, a fin de atender las necesidades de selección de personal, de planeación, desarrollo organizacional, remuneraciones al personal y capacitación.

[10] Citado en el *Manual de organización específico de la Dirección de Técnica Operativa de la Subsecretaria de Hacienda y Crédito Público*. Septiembre de 1996. p.p. 8

Entre sus funciones podemos mencionar las sigs.:

- Supervisar el cumplimiento de las normas, políticas y procedimientos en materia de administración de personal, emitidos por la Oficialía Mayor.
- Coordinar las actividades de reclutamiento, selección, contratación, inducción, desarrollo y capacitación al personal adscrito a la Subsecretaría.
- Llevar a cabo el control de plazas, pagos, prestaciones, movimientos e incidencias de personal.
- Atender y resolver las reclamaciones por parte del personal, en relación con el pago de sueldos.
- Promover la realización de los eventos de capacitación en las unidades administrativas, propuestos por la Secretaría e instituciones especializadas.
- Difundir y tramitar las prestaciones a las que tiene derecho el personal de la Subsecretaría y sus familiares.
- Todas aquellas que en el ámbito de su competencia le encomiende la superioridad.

Arriba se mencionaron las funciones generales de la Subdirección de Recursos Humanos, en apoyo a tales funciones ésta se ha organizado en dos departamentos y ocho oficinas.

SUBSECRETARIA DE HACIENDA Y CREDITO PUBLICO
DIRECCION DE TECNICA OPERATIVA
SUBDIRECCION DE RECURSOS HUMANOS

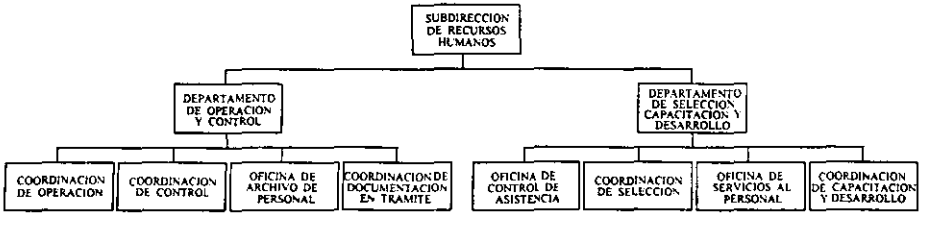


Fig. 2.4

El Departamento de Operación y Control fue creado para atender los requerimientos de personal de las diferentes unidades administrativas dependientes de la Subsecretaría del ramo y controlar los cambios de adscripción, licencias y bajas; asimismo, atender el pago de remuneraciones, manejar altas, bajas, promociones, conversiones y demás movimientos, a fin de contar con un registro actualizado de la plantilla de personal.

Entre sus funciones están:

- Control de suspensiones de efectos de nombramiento.
- Controlar licencias sindicales.
- Hacer conversiones de plazas-puesto
- Realizar movimientos de personal
- Dar aviso de cambios de situación de personal federal.
- Control de licencias.
- Notificar la recuperación de salarios no devengados cuando se amerite.
- Inscripción al programa de retiro voluntario
- Inscripción al personal al Fondo de Ahorro Capitalizable (FONAC).
- Manejar movimientos de personal al Sistema de Ahorro para el retiro (SAR).
- Gestionar la regularización de sueldos.
- Regularizar sueldos.
- Controlar pagos ordinarios y adicionales.
- Distribuir comprobantes de pago.
- Realizar regularizaciones del pago de prima quinquenal.
- Gestionar la declaración de situación patrimonial de mandos medios.

El Departamento de Selección, Capacitación y Desarrollo realiza las actividades mencionadas de acuerdo a la normatividad establecida por la Oficialía Mayor, a fin de reclutar candidatos para cubrir los puestos vacantes, llevar el control de incidencias de inasistencias, apoyar al personal en los trámites de los servicios a los que tiene derecho, así como difundir los eventos de capacitación que proporciona la Secretaría.

Sus funciones incluyen:

- Reclutamiento de personal.
- Selección de personal.
- Filiación.
- En colaboración con el Departamento de Operación y control se realizan las contrataciones de personal.
- Integración de la bolsa de trabajo.
- Detección de necesidades de capacitación.
- Coordinación y programación de cursos, diplomados, conferencias y eventos para capacitación y desarrollo.
- Promoción e inscripción al sistema de educación abierta.
- Gestión de becas.
- Control del programa para prestadores de servicio social.
- Elaboración de actas administrativas.
- Revisión y concentración de las cédulas referentes a premios, estímulos y recompensas.
- Control de asistencia.
- Otorgamiento de estímulos a empleados puntuales.
- Tramitación de credenciales de identificación.
- Tramitación del nuevo seguro institucional
- Tramitación de seguros colectivos de gastos médicos mayores.
- Adscripción del trabajador hacendario ante el I.S.S.S.T.E.

Para un manejo adecuado y oportuno de las actividades de la Subdirección de Recursos Humanos, se solicitó un sistema de información. El objetivo básico de tal sistema sería coadyuvar al control y toma de decisiones en lo relativo a los movimientos de personal y de plazas-puesto; ingresos, reingresos, bajas, cambios de adscripción, promociones, vacantes, conversiones de puestos, etc. Algunas actividades menos técnicas, las que realiza el departamento de selección, capacitación y desarrollo, como el control de asistencia, el manejo de credenciales y el programa de educación abierta quedan fuera del alcance del sistema deseado.

II.3.1.2. Descripción de la situación actual

Como puede notarse, el Departamento de Operación y Control administra las plazas-puesto de la Subsecretaría del Ramo, así como los movimientos de personal de acuerdo a su plaza asignada, mientras que el Departamento de Selección, Capacitación y Desarrollo, por su parte, provee de recursos humanos a la Subsecretaría y gestiona los servicios a los que el personal tiene derecho

Con el fin de apoyar las actividades del Departamento de Operación y Control, en 1994 se desarrolló el *Sistema de Plantillas*.

El sistema de plantillas permite crear puestos y asignarlos a empleados, lleva un registro de vacantes, licencias, sueldos, además de una lista de datos generales de los trabajadores. Sin embargo, el sistema no contempla las promociones de personal, las reubicaciones se hacen directamente sobre la base de datos, no se lleva un control sobre fondos de ahorro (SAR y FONAC), no se contabilizan incidencias para efecto de recuperación de sueldos a devengar, ni se contempla el control de quinquenios.

Lo que es peor, el Sistema de Plantillas se ha estado parchando desde entonces para manejar las modificaciones que se han presentado; se ha incrementado el número de campos en una tabla de la base de datos para representar ciertas situaciones, como lo es distinguir tipo de personal, empleados de confianza que cotizan al Sindicato Nacional de Trabajadores de Hacienda, pagos a la Cruz Roja Mexicana, personal comisionado a otras áreas, manejo de las plazas presupuestal y organizacionalmente, etc.

Lo anterior desembocó en una estructura de tablas mal organizada, poco clara e inmantenible.

Como el Sistema de Plantillas no contempla incidencias, y puesto que realizar modificaciones a la estructura de las tablas se volvió difícil, hace unos años se optó por desarrollar un *Sistema de Control de Incidencias*. La solución no fue muy adecuada, puesto que este sistema está aislado de la base de datos de plantillas y duplica las tablas de personal con su propia plantilla de empleados, ocasionando, a veces, inconsistencia de los datos (ambas bases de datos no poseen la misma plantilla de personal, lo cual no debería ser posible).

El Sistema de Control de Incidencias contabiliza tipos de inasistencias (licencias médicas, cuidados maternos, faltas no justificadas) con el fin de reflejar sanciones en las percepciones de los empleados por concepto de incentivos (productividad).

Todo lo anterior, aunado a la falta de cultura informática de los mandos medios de la Subdirección de Recursos Humanos y a problemas administrativos, ha generado desconfianza en los sistemas de información computarizados. El Departamento de Operación y Control ejerce sus actividades en forma manual; los nombramientos de ingresos, reingresos, cambios de adscripción, promociones, etc. se registran en formatos de papel, se contabilizan las plazas, se archivan. Después se actualizan estos movimientos en los Sistemas de Plantillas y Sistemas de Control de incidencias, de tal forma que al cierre de quincena se cotejan ambos resultados (contabilización manual y por programa), se realizan los ajustes necesarios en los sistemas o en los controles manuales, y se emiten reportes.

Por otra parte, el Departamento de Selección, Capacitación y Desarrollo cuenta con el Sistema de Reloj y el Sistema de Madres Trabajadoras.

El *Sistema de Reloj* registra las firmas de entrada y salida automatizadas exclusivamente del personal de la Dirección Técnica Operativa. Contabiliza los registros de asistencia del personal operativo de la Oficina del C. Subsecretario del ramo, la Asesoría y la Unidad de Sistemas Estadísticos, las cuales son recibidas a través de listas de asistencia. A finales de quincena son enviadas a la Dirección General de Recursos Humanos las listas de personal

puntual, personal con licencias, personal con faltas injustificadas, etc., con el propósito de aplicar los respectivos descuentos al salario.

El *Sistema de Madres Trabajadoras* registra datos generales de los hijos de las empleadas hacendarias para entrega de regalos (juguetes en día de reyes, día del niño, etc.).

Es necesario mencionar que los cuatro sistemas descritos someramente están aislados entre sí (Sistema de Plantillas, Sistema de Control de Incidencias, Sistema de Reloj y Sistema de Madres Trabajadoras). Tienen una plantilla de personal conceptualmente común, pero físicamente cada uno tiene su propia base de datos. En ocasiones encontramos en el sistema de madres trabajadoras personal que ya no labora en esta subsecretaría, lo mismo ocurre en el catálogo de personal del sistema de reloj.

Para los servicios a empleados, tales como cursos, diplomados, solicitudes de préstamos, inscripción al ISSTE, FONAC, SAR, gestión de seguros médicos, tramitación de credenciales, etc., se utilizan formatos. La información se obtiene del Sistema de Plantillas, razón por la cual la gente encargada de tramitar estas formas solicita a la jefa del Departamento de Control y Operación la relación deseada, ella turna el trabajo al Departamento de Informática. Este ultimo genera la relación, la cual se regresa al Departamento de Operación y Control y después al usuario final. El usuario llena el formato con la información obtenida, ya sea a máquina de escribir, o a través de un procesador de palabras y/o hoja de cálculo.

Tal ejemplo de burocratismo clásico genera duplicidad (triplicidad y tetraplicidad) de trabajo y mal uso de recursos humanos, materiales (uso de equipo de cómputo, impresoras, hojas, etc.) y tiempo. Además, cuando por alguna razón, el reporte no es entregado a tiempo, a menudo se culpabiliza al Departamento de Informática, cuando la mayoría de las veces la solicitud del listado es la que está atrasada.

A veces nos solicitan reportes que parecen no tener algún criterio de búsqueda en la base de datos, ya sea porque el elemento común para tal reporte no está contemplado en tablas o porque aunque si está contemplado, el estado actual de dichas tablas hace difícil su elaboración.

Hace unos meses el jefe del Departamento de Selección, Capacitación y Desarrollo planteó la necesidad de un Sistema que controle el número de cursos de capacitación a los que el personal ha asistido, así como la creación de una relación de cursos e instituciones educativas que los imparte.

El estado de aislamiento de los sistemas ha provocado información redundante e inconsistente, lo cual nos ha llevado a proponer un Sistema Integral que elimine estos problemas. Sin embargo, la inclusión de características nuevas en los sistemas, el cambio de algunas que ya existen, además de el hecho de compartir la misma información no parecen ser soportados por las bases de datos existentes. Por lo tanto hemos propuesto la creación de un sistema totalmente nuevo; el Sistema Integral de Control de Plazas y Recursos Humanos (SICPRH).

El SICPRH estaría basado en el proceso técnico de creación, conversión (cambio) y eliminación de puestos así como su asignación al personal. Se registrarán los datos generales de los empleados y sus familias (para el caso de madres trabajadoras). También se contemplarán sus derechos, tales como SAR, FONAC, ISSSTE, seguros de vida, etc. Se desea que los empleados tengan un kárdex en donde se registre el último puesto del empleado, su último salario, su antigüedad (a efectos de quinquenios), escolaridad, cursos y diplomados que ha tomado, becas a las que ha sido acreedor, etc.

El SICPRH involucrará las funciones de los Sistemas de Plantillas, de Control de Incidencias y de Madres trabajadoras. Por política de la Secretaría, el SICPRH no abarcará las funciones del Sistema de Control de Asistencia, y sólo el personal autorizado podrá tener acceso a las tablas y asignaciones de sueldos e incentivos.

II.3.1.3. Políticas operacionales y restricciones

Ahora se describirá cada una de las actividades que serán contempladas en el SICPRH, así como sus políticas de operación y algunas restricciones.

CONTRATACIONES DE PERSONAL

Es el convenio celebrado entre la Secretaría y el personal a efectos de que éste preste sus servicios a favor de la institución, recibiendo una remuneración a cambio.

Políticas de operación.

Ninguna persona podrá iniciar sus servicios en la Secretaría, sin haber cumplido previamente con los trámites y requerimientos de admisión establecidos en los artículos 9 y 10 de las Condiciones Generales de Trabajo, vigentes en esta Secretaría.

Deberá integrarse el expediente con la siguiente documentación:

- Constancia de nombramiento.
- Solicitud de empleo, debidamente requisitada.
- Compatibilidad de empleos en caso de laborar en otra dependencia del Gobierno Federal.
- Acta de nacimiento.
- Filiación certificada por el Sistema Nacional de Filiaciones.
- Cartilla del Servicio Militar Nacional, en su caso.
- Constancia oficial de estudios, de acuerdo al perfil del puesto.
- Certificado médico, emitido por Institución Oficial.
- Consentimiento para ser asegurado y designación de beneficiario de Aseg. Hidalgo.
- Contrato del S.A.R.
- Curriculum vitae (únicamente mandos medios y superiores).

CAPACITACIÓN Y DESARROLLO.

Constituye un programa integral de instrucción a los empleados con el fin de adquirir conocimientos y habilidades que permitan el desarrollo personal haciendo coincidir éstos con los objetivos institucionales.

Políticas de operación.

- Es responsabilidad del trabajador participar en los programas de capacitación.
- Las acciones de capacitación se orientarán al personal operativo (vertiente operativa), mandos mèdios y superiores (vertiente de responsabilidad) y a la actualización de conocimientos (vertiente institucional).
- Serán consideradas como acciones de capacitación los cursos, talleres, conferencias y otros eventos que cumplan con los requisitos del proceso técnico de capacitación.
- Las acciones de capacitación para su realización se ordenarán en cuatro etapas:
  - Detección de necesidades.
  - Programación.
  - Ejecución
  - Seguimiento y evaluación.
- Las Unidades Administrativas deberán:
  - Aplicar una detección de necesidades de capacitación al personal de su área de adscripción.
  - Elaborar la calendarización anual y trimestral de los cursos, programados en base a los datos obtenidos en la detección de necesidades de capacitación.
  - Evaluar el desempeño laboral del personal capacitado.
- El personal que participe en cursos de capacitación tendrá derecho a obtener Constancia de Participación siempre y cuando cumpla con un mínimo de 80% de asistencia y calificación aprobatoria mayor a 7.4

GESTIÓN DE BECAS.

Promueven el desarrollo del empleado hacendario, permitiéndole ampliar sus conocimientos en las distintas materias vinculadas con su puesto y con el campo de actuación de la Subsecretaría de Hacienda y Crédito Público.

Políticas de operación.

- Para la obtención de una beca el solicitante deberá ser empleado de la Secretaría de Hacienda y Crédito Público y tener una antigüedad mínima de dos años.
- No se otorgará beca a estudios de preparatoria o equivalente.
- El empleado becario se comprometerá a pagar un porcentaje del costo de inscripción y colegiatura de acuerdo a su nivel de puesto.
- El becario deberá obtener calificaciones aprobatorias y guardar la conducta adecuada durante el tiempo que dure el estudio.
- En caso de abandonar los estudios o reprobarlos, el becario se compromete a reintegrar el costo total de la beca.
- Las becas internas, a diferencia de las becas normales que son saldadas por la Secretaría a través de la Dirección General de Recursos Humanos, son financiadas por la Dirección General a la que pertenece el becario. Este tipo de becas deberá ser autorizada por el titular de la Unidad Administrativa y se deberá contar con disponibilidad presupuestal en la partida correspondiente.

Restricciones

Las becas internas no son manejadas por la Subdirección de Recursos Humanos, sino por la Subdirección de Programación, Organización y Presupuesto.

PREMIOS, ESTÍMULOS Y RECOMPENSAS.

Consiste en un sistema que estimula y premia a los servidores públicos por la actitud y aptitud manifestada en el desempeño de las actividades del puesto que ocupa en la Subsecretaría de Hacienda y Crédito Público, a fin de proporcionar la satisfacción en el desempeño laboral y desarrollar el potencial de los trabajadores que conlleva al incremento de productividad y calidad de los servicios ofrecidos por la Secretaría.

Políticas de operación

- Debe realizarse el sistema de evaluación de desempeño a todo el personal operativo de base o confianza hasta el nivel salarial 27 C.
- El personal debe ser evaluado semestralmente obteniendo un promedio anual.
- Cada empleado público deberá ser evaluado por su jefe inmediato.
- El evaluador firmara las cédulas de evaluación pero permanecerá anónimo con relación al empleado.
- Los estímulos y recompensas serán repartidas entre empleados con una año de antigüedad como mínimo.
- Los estímulos y recompensas consistirán en las que determine la Dirección General de Recursos Humanos, podrán ser periodos de vacaciones extraordinarias, pagos en efectivo, preseas, etc.

Los estímulos anteriores son independientes de los estímulos que puedan resultar de concursos extraordinarios, tales como Olimpiadas de Calidad, Productividad, Premio Nacional de Administración Pública, etc.

Restricciones.

Los estímulos económicos se dan al personal, no a la plaza. Por esta razón ningún estímulo, premio o recompensa se considera en las partidas presupuestales correspondientes a pago de empleados. Lo anterior para efectos de conversiones de puestos-plazas.

El manejo de esta información es confidencial, por tal motivo no será considerada en el SICPRH.

SEGURO INSTITUCIONAL

La Secretaría permite brindar una mejor protección y coadyuva a fortalecer la estabilidad económica familiar en caso de invalidez total y permanente o fallecimiento del empleado hacendario.

Políticas de operación

- Todos los servidores públicos deben ser incluidos en este programa con el fin de protegerlos en caso de invalidez total y permanente o fallecimiento, con excepción de los trabajadores que presten sus servicios por honorarios.
- Los servidores públicos, cualquiera que sea su sexo, edad u ocupación, sin necesidad de examen médico serán integrados al programa de seguro de vida institucional.
- Las aportaciones del seguro institucional para el servidor público son de 1.8% del salario.
- En caso de incapacidad total y permanente o de fallecimiento, será igual a la cantidad equivalente a 40 meses del último salario devengado al momento de ocurrir el siniestro.
- Las Unidades Administrativas serán las encargadas de todas las gestiones necesarias que resulten de los puntos anteriores, ya sea con el servidor público, sus deudos y/o con Aseguradora Hidalgo.

SEGURO COLECTIVO DE GASTOS MÉDICOS MAYORES.

Es un complemento a la asistencia médica que proporciona el sistema actual de seguridad social con que cuentan los trabajadores (ISSSTE). El seguro de gastos médicos mayores cubrirá los gastos resultantes de la atención médica y hospitalaria por una enfermedad o accidente amparado por una póliza.

Políticas de operación

- Se incorporan de manera automática al Seguro Colectivo de Gastos Médicos Mayores todos los servidores públicos que ocupen del nivel MC06 hasta el nivel 37.

- Se podrá incorporar voluntariamente a dicho seguro al cónyuge, hijos dependientes económicos y padres del titular.
- La suma asegurada básica, otorgada por el Gobierno Federal, es de 111 veces el Salario Mínimo General Mensual del DF (SMGMDF) hasta 333 veces, dependiendo del nivel sel servidor público.
- De manera voluntaria, con prima a su cargo, el servidor público puede incrementar su suma asegurada hasta alcanzar 740 veces el SMGMDF.
- Las Unidades Administrativas serán las encargadas de todas las gestiones necesarias que resulten de los puntos anteriores, ya sea con el servidor público, sus deudos y/o con Aseguradora Hidalgo.

ADSCRIPCIÓN ANTE EL I.S.S.S.T.E.

Inscripción del servidor público al fondo de seguridad social y médica que proporciona el I.S.S.S.T.E.

Políticas de operación

- La Unidad Administrativa deberá recibir la documentación del candidato a ingresar a la Subsecretaría de Hacienda y Crédito Público de conformidad con los requisitos establecidos.
- La Unidad Administrativa deberá elaborar constancia de nombramiento y oficio de envío dirigido a la Dirección General de Recursos Humanos, para operar en el módulo de "Movimientos afiliatorios al ISSSTE" el alta respectiva.

SUSPENSIONES DE EFECTOS DE NOMBRAMIENTO.

La suspensión de efectos de nombramiento es de naturaleza temporal, podrá ser total, si opera tanto en funciones como en salario, y parcial, si tan solo se dicta en las funciones. En ambos casos implica:

Para el trabajador es la abstención del ejercicio de los derechos y el cumplimiento de las obligaciones que la Ley y las Condiciones Generales de Trabajo le otorgan y le imponen.

Para la Secretaría significa dejar de utilizar los servicios del trabajador y la interrupción paralela de las obligaciones que respecto a éste, le imponen la Ley y las Condiciones Generales de Trabajo.

Políticas de operación.

- La suspensión operará:
  - Cuando se compruebe con certificado médico expedido por el ISSSTE, que el trabajador contrajo alguna enfermedad transmisible que signifique un peligro de contagio.
  - Cuando se descubra alguna irregularidad en la gestión de los trabajadores que tengan encomendado el manejo de fondos, valores o bienes. Podrán ser suspendidos hasta por 60 días mientras se practica la investigación y se resuelve su cese.
  - La prisión preventiva del trabajador seguida de sentencia absolutoria.
  - Cuando la ordene el titular o servidor público legalmente autorizado para ello en casos en que los trabajadores se encuentren demandados por incurrir en alguna de las causales de cese.
- Se mantendrá actualizada la información relativa a la situación que guarda el personal en la materia por parte de las Unidades Administrativas.

TERMINACION DE LOS EFECTOS DE NOMBRAMIENTO

Consiste en la conclusión de las relaciones entre el servidor público y la Secretaría.

Políticas de operación.

Son motivos de cese de los efectos de nombramiento lo dispuesto en los artículos 138, 139 y 140 de las condiciones Generales de Trabajo. A saber por renuncia, abandono de empleo, abandono de labores técnicas, falta de probidad u honradez y renuencia a los castigos descritos en el artículo 46 fracción V de esa ley.

AVISO DE CAMBIO DE SITUACIÓN DE PERSONAL FEDERAL.

Se consideran cambios de situación de personal los sigs. movimientos: bajas, licencias, reanudación de labores y cambios de radicación de sueldos. Estos movimientos tan comunes se gestionan a través del formato de Aviso de Cambio de Situación de Personal Federal.

Las bajas de empleados denotan terminación de efectos de nombramiento.

Las licencias consisten en suspensiones de efectos de nombramiento, asimismo, en el formato de Aviso de Cambio de Situación de Personal Federal se indica la fecha de reanudación de labores.

Los cambios de radicación de sueldos ocurren cuando el personal es comisionado a desempeñar sus servicios en el interior de la República sin menoscabo de su plaza.

Políticas de Operación

- Cualquier cambio de situación de personal federal deberá ser notificada a la Dirección General de Personal para su gestión administrativa dentro de los plazos establecidos, quedando obligadas las jefaturas de servicios administrativos a presentar la documentación original dentro de los quince días siguientes al cierre de la quincena.
- Si el movimiento es de baja, se deberá especificar el tipo de baja en que se esté incurriendo para efecto de presentarse alguna situación específica en trámites especiales.
  - **Baja por renuncia.** Las bajas que se generen por este concepto deberán enviarse a la Dirección General de Personal conforme al calendario establecido.
  - **Baja por defunción**. La Dirección Técnica Operativa deberá elaborar el aviso de cambio de situación de personal federal, con la fecha que indique el acta de defunción así como original de la misma.
  - **Baja por jubilación.** La fecha anotada en el aviso de renuncia será la indicada por el empleado, la cual deberá coincidir con la del día inmediato siguiente al término de la licencia por jubilación. En caso de que el empleado renuncie por este motivo y no haya disfrutado de la licencia prejubilatoria, deberá anotarse esta situación en el campo de observaciones.

– Los avisos que promuevan **bajas por abandono de empleo, pérdida de la confianza, acuerdo superior, resolución del Tribunal de Conciliación y Arbitraje**, serán elaborados por esta Dirección; las bajas por incapacidad ISSSTE serán elaboradas por la Dirección General de Personal.

- En caso de licencias, se deberá atenerse a lo siguiente:

  – **Licencias sin goce de sueldo**. La solicitud de licencia por asuntos particulares sin goce de sueldo, deberá realizarlo esta Dirección, antes del inicio de la misma y adjuntando el Aviso de Cambio de Situación Federal respectivo y la Constancia de Nombramiento y/o asignación de remuneraciones, en caso de ser una licencia sin goce de sueldo de más de seis meses.

  – **Licencias por enfermedad**. Las licencias médicas que al computarizarse requieran de ajuste en las percepciones del empleado (las que sobrepasen doce días de licencias anuales empezando y terminando en Mayo), invariablemente deberán reportarse en el Aviso de Cambio de Situación de Personal Federal.

  – **Licencia Prejubilatoria** Se orientará al empleado a efecto de que se culminen éstas en días 15 o a finales de mes.

- Cuando ocurre un cambio en la radicación de sueldos:

  – **Cambio de Radicación de Sueldos.** La vigencia de los cambios de radicación, cuyo trámite se solicite, deberá ser a partir de los días primero o 16 del mes, anexando a éstos fotocopia del oficio, mediante el cual se notifica al empleado su cambio, ajustándose al calendario para la recepción de constancias de nombramiento y/o asignación de remuneraciones.

- Será obligación de la Dirección mantener actualizado un registro kardex con los datos de incidencias de los trabajadores de la región adscritos.

• En caso de que no se envíen los documentos originales y completos a la Dirección General de Personal, ésta procederá a la suspensión del pago de remuneraciones de los empleados involucrados en los movimientos, y para los casos de bajas tramitadas se congelará la vacante que no haya sido notificada.

CONTROL DE LICENCIAS.

A través del Aviso de Cambio de Situación de Personal Federal se notifican a la Dirección General de Recursos Humanos los movimientos en cuanto a licencias se refieren a excepción de las licencias médicas que no constituyen enfermedades permanentes y/o profesionales, éstas se notifican en formatos de control de asistencia. El control de licencencias determina las acciones a realizar para el desarrollo de los trámites involucrados.

Políticas de operación.

• Las Unidades Administrativas elaborarán quincenalmente el reporte de licencias otorgadas a los trabajadores debidamente signado por el titular de la Coordinación Administrativa.

• El trabajador podrá pedir una licencia sin goce de sueldo con una anticipación de un mes en el supuesto de seis meses de licencia, tres semanas si es para cuatro mesos, dos semanas en el caso de dos meses y una semana en el caso de un mes. Estas licencias se podrán conceder:

  – A los trabajadores con antigüedad mayor a seis meses para asuntos particulares.

  – A los trabajadores electos para cargos de elección popular, durante el tiempo que dure su cargo.

  – A los trabajadores que ocupen puesto de confianza tanto en la dependencia de adscripción como en otras.

• Las licencias con goce de sueldo se podrán conceder:

- Hasta por tres días en un mes sin exceder de tres veces en un año.
- Hasta por el tiempo que dure el encargo de un trabajador, como miembro de los Comités Nacionales del Sindicato o como representante del mismo ante la Federación de Sindicatos de Trabajadores al Servicio del Estado (licencias sindicales).
- Hasta por cinco días para contraer matrimonio.

Hasta por cinco días naturales para sustentar examen profesional a nivel licenciatura o postgrado.

CONVERSIONES DE PLAZAS-PUESTO

Las conversiones de plaza-puesto significan la reducción de la plantilla de las Unidades Administrativas a cambio de puestos de mejor nivel.

Políticas de Operación

- Para realizar una conversión de plaza-puesto, será indispensable que exista la plaza a convertir y que el puesto requerido se encuentre contemplado en el catálogo institucional de puestos en vigor.
- Toda solicitud de conversión deberá venir acompañada con la información correspondiente; constancias de nombramiento y/o asignación de remuneraciones (sin vigencia y sin percepciones); formato de solicitud de conversión de puestos-plazas asegurándose que la plaza a convertir de reducción exista en las plantillas de puestos-plazas autorizadas por esta Dirección General de Personal.
- Las conversiones propuestas deberán ser autofinanciables, es decir que el área solicitante deberá aportar los recursos necesarios para efectuar el movimiento requerido mediante la acumulación de plazas vacantes, adicionando el equivalente de reducción al 30% del costo total de las conversiones, que requiera la Dirección General de Normatividad y Desarrollo Administrativo, como medidas de racionalización para apoyo de futuros incrementos salariales

MOVIMIENTOS DE PERSONAL

Los movimientos de personal se registran en el formato de Constancia de Nombramiento y/o Asignación de Remuneraciones, éstos son:

Un movimiento de **nuevo ingreso** significa la contratación de un empleado que nunca ha laborado en esta Secretaría.

Un **reingreso** es la recontratación de personal.

Una **promoción** consiste en cualquier ascenso de nivel en las percepciones de un empleado. En la Secretaría de Hacienda y Crédito Público no se manejan despromociones.

Por **cambio de adscripción** se entiende al cambio de plaza-puesto asignado a un empleado, de tal forma que este último pueda percibir un mayor o menor sueldo (dependiendo del tipo de plaza que tenía y de la que se le asigna). Sería un cambio de adscripción interno si la plaza por recibir pertenece presupuestalmente a la Subsecretaría de Hacienda y Crédito Público; es externa cuando la plaza que recibe el empleado pertenece presupuestalmente a la Subsecretaría de Egresos, Oficialía Mayor, Procuraduría Fiscal de la Federación o Tesorería de la Federación.

Una reubicación es el movimiento de personal y de su plaza-puesto a una área diferente a la de su adscripción.

Políticas de Operación

- Cualquier tipo de movimiento de personal deberá cumplir con la documentación necesaria, conforme al perfil del puesto.
- Todo movimiento se deberá sujetar a las vacantes existentes, tomando en cuenta el presupuesto asignado.
- En caso de **nuevo ingreso**, la documentación requerida es:
  - Copia de Constancia de Nombramiento
  - Copia Certificada del Acta de Nacimiento.
  - Filiación.

- En su caso, copia fotostática de la cartilla de liberación del S.M.N. y el resello de vigencia.
- Original y copia de la solicitud de empleo de la S.H.C.P. con fotografía tamaño infantil.
- Certificado médico, en original y copia.
- Formato de designación de beneficiarios del nuevo seguro institucional
- Original del contrato para la inscripción al S.A.R. o si ya se tiene cuenta abierta, anexar el último estado de cuenta

- En caso de **reingreso** solicitar la siguiente documentación
  - Las enunciadas en el caso anterior
  - La Dirección General de Personal tramitará las reubicaciones internas de plazas que presenten la Dirección Técnica Operativa, siempre que se traten de aquellos de naturaleza operativa dando prioridad a las acciones de desconcentratización.
- En caso de **conversiones** deberá presentarse Constancia de Nombramiento sin vigencia ni percepciones, de igual forma la solicitud de conversión puesto-plaza; asimismo, deberán presentar dichos documentos con oficio dirigido al Director de Presupuesto y Pagos de la Dirección General de Personal.
- En caso de **retiro voluntario** se deberá presentar las Constancias de Nombramiento que integra la cadena promocional que se genera por el retiro voluntario, así como el mismo volante de cancelación de plaza debidamente autorizado por el Departamento de Certificaciones de la Dirección General de Personal.

SALARIOS NO DEVENGADOS

La recuperación de salarios no devengados consiste en la aplicación de ajustes y descuentos en percepciones que hace la Secretaría de Hacienda y Crédito Público. Estas son originadas por incidencias (faltas, acumulaciones de cuidados maternos, acumulaciones de licencias médicas, etc.) en que incurren los servidores públicos.

Políticas de operación

- Las Unidades administrativas llevarán a cabo el registro y actualización del control de incidencias del personal, de acuerdo a la normatividad vigente establecida.
- Las Unidades Administrativas deberán vigilar que el personal que incurra en alguna incidencia proporcione los comprobantes y los justificantes como máximo dos días hábiles después de ocurrir la incidencia.
- Las Unidades Administrativas serán las responsables de reportar quincenalmente las incidencias ocurridas, verificando que se gestionen los ajustes y los descuentos que procedan con la oportunidad necesaria.
- Para el caso de reporte de incidencias se contemplan dos criterios, el primero consiste en lo establecido en las Condiciones Generales de Trabajo; el segundo se basa en la normatividad de descuentos a incentivos mensuales.

PROGRAMA DE RETIRO VOLUNTARIO

Es un programa que apoya a los servidores públicos que desean voluntariamente separarse del servicio de la Secretaría, obteniendo una compensación económica que redituе en su beneficio personal sin menoscabo de sus derechos adquiridos.

Políticas de operación

- Quedan incluidos todos los servidores públicos considerados como personal operativo, siempre y cuando al momento de su incorporación a dicho programa no tengan derecho a pensión a cargo del I.S.S.S.T.E o puedan ser considerados en el Programa Pensionario de Trato Especial.
- Quedan excluidos de este programa los servidores públicos superiores, mandos medios y homólogos, puestos de enlace.

- Será responsabilidad de la Unidad Administrativa el verificar, antes de opinar favorablemente la solicitud, que la ausencia del trabajador no afectará prioritarios o la adecuada operación de las funciones sustantivas.
- El personal que se incorpore al Programa tendrá derecho a una compensación económica, en función de los años de servicios y el nivel de remuneraciones asignado.
- La muerte del trabajador no cancelará estos derechos a favor de sus beneficiarios.
- Una vez autorizada la incorporación al programa no se aceptará desistimientos.

Restricciones.

Para los movimientos de retiro voluntario sólo se tiene interés al momento de la liberación de una plaza-puesto. Los beneficios del personal retirado son manejados en la Dirección General de Recursos Humanos.

FONDO DE AHORRO CAPITALIZABLE (FONAC)

El FONAC es un fondo de ahorro con aportaciones quincenales que suscribe la Secretaría de Hacienda y Crédito Público y los empleados con puestos operativos del nivel 15 al 27 C.

Un porcentaje menor al 6% del sueldo mensual es aportado por los empleados y el restante lo cubre la Secretaría. Los empleados de base reciben un porcentaje adicional por parte del Sindicato Nacional de Trabajadores de la Secretaria de Hacienda.

El FONAC proporciona, además, un seguro de vida por fallecimiento o invalidez total y permanente.

Políticas de operación

- Las Coordinaciones Administrativas deberán:
- Proporcionar la asesoría y el apoyo técnico necesario para dilucidar posibles dudas relativas al funcionamiento del FONAC.

• Concentrar las solicitudes tanto de inscripción. como de renuncia al FONAC, y liquidación anticipada del mismo; así como de enviarlas en el tiempo requerido a la Dirección General de Recursos Humanos.

• Tramitar en su caso, ante la Dirección General de Recursos Humanos la reclamación por escrito de parte del beneficiario del pago del seguro de vida por fallecimiento o invalidez total y permanente otorgado por formar parte del FONAC y liquidación de aportaciones quincenales.

SISTEMA DE AHORRO PARA EL RETIRO (SAR).

Es una cuenta de ahorro depositada en instituciones bancarias con carácter obligatorio. Los servidores públicos realizan sus propias aportaciones en dos subcuentas; la cuenta de vivienda y la de pensión.

Políticas de operación.

• Las aportaciones se calcularán en forma mensual, por el importe equivalente al 2% del sueldo del tabulador conforme al puesto y nivel del trabajador.

• La Dirección General de Recursos Humanos cubrirá las aportaciones mediante la entrega de los recursos correspondientes en instituciones de crédito, para su abono en las cuentas individuales del S.A.R. abiertas a nombre de los trabajadores.

• La cuenta individual será única y contendrá para su identificación el Registro Federal de Contribuyentes por lo que no deberá tenerse otra cuenta aun teniendo otra relación de trabajo.

• El entero de las aportaciones se hará por bimestre vencido.

• El trabajador deberá solicitar por escrito a la institución de crédito la entrega de los fondos de la cuenta del S.A.R., acompañando los documentos respectivos (formularios).

• El trabajador titular de la cuenta del S.A.R., deberá a la apertura de la misma, designar beneficiarios. lo anterior sin perjuicio de que en cualquier tiempo el trabajador pueda

sustituir a las personas que hubiera designado, así como modificar en su caso, la proporción correspondiente a cada una de ellas.

- El trabajador, en cualquier momento podrá solicitar directamente a la institución de crédito el traspaso de sus fondos del S.A.R a otra institución.

REGULARIZACIÓN DE SUELDOS.

La regularización de sueldos permite corregir las irregularidades presentadas en sueldos, ya sea por percepciones o por descuentos indebidos, ante la Dirección General de Recursos Humanos y establecer los mecanismos de seguimiento y control, a fin de evitar en lo posible incurrir en errores posteriores.

Políticas de operación.

- Será obligación de las Unidades Administrativas analizar y verificar en kardex y controles el origen de la irregularidad. El reporte para regularización de sueldos deberá enviarse con los siguientes anexos:
- Parte proporcional de aguinaldo cuando exista una baja de personal.
  - Fotocopia del Aviso de Cambio de Situación de Personal Federal por baja.
  - Fotocopia del último comprobante de pago.
  - En caso de defunción, original del acta correspondiente y original del acta de nacimiento o matrimonial certificada, según sea el caso.
- Regularización de sueldos.
  - Copia fotostática de comprobante de pago que comprenda el período que solicita la regularización.
  - Copia fotostática del último comprobante de pago en caso de omisión de depósito.
- Pago de prima vacacional.
  - Fotocopia de último comprobante de pago
  - Comprobante de pago de 6 meses previo a la fecha de pago de partida 32.

- Diferencia de aguinaldo.
  - Copia de constancia de Nombramiento y/o Asignación de Remuneraciones en el caso de promoción y reingreso, o Aviso de Cambio de Situación de Personal Federal por licencia y reanudación de labores.
  - Copia fotostática de comprobante de pago del empleado en donde se realizaron los pagos ordinarios de aguinaldo.
  - Anexar fotocopia del último comprobante de pago y los comprobantes de pago de las quincenas donde le fue suspendido el concepto.
- Omisión de pago de quinquenio.
  - Elaborar oficio para la regularización del pago del concepto, anexando copia del comprobante de pago en el cual figuraba el quinquenio y en el que ya no figura.

PAGOS ORDINARIOS Y ADICIONALES.

Es la entrega del sueldo que le corresponde al personal por la prestación de sus servicios a la Secretaria.

Políticas de operación.

- El sueldo será para cada uno de los puestos consignados en el Catálogo Institucional de Puestos del Gobierno Federal y se fijará con los tabuladores, quedando comprendido en los presupuestos de egresos respectivos.
- El sueldo que se asigna en los tabuladores para cada puesto constituye el salario total que debe pagarse al trabajador a cambio de los servicios prestados, sin perjuicio de otras prestaciones ya establecidas.
- Para suspender el pago al personal civil federal, se deberá tomar como base el Aviso de Cambio de Situación de Personal Federal correspondiente debidamente autorizado.
- Los comprobantes de pago y/o ficha de depósito solo se entregarán al interesado, previa presentación de una identificación personal oficial.

• En caso de que el empleado esté incapacitado para recoger su comprobante de pago y/o ficha de depósito, podrá realizarlo a través de una tercera persona, con la presentación de una carta poder, una identificación y la justificación correspondiente.

• La Dirección General de Recursos Humanos a través de las Unidades Administrativas será la encargada del envío de los comprobantes de pago y/o fichas de depósito de nómina ordinaria en las fechas señaladas en el calendario correspondiente.

REGULARIZACIONES DEL PAGO DE PRIMA QUINQUENAL.

Consiste en la inscripción y actualización de pago de quinquenios a que tienen derecho los servidores públicos.

Políticas de operación.

• Los trabajadores que presten sus servicios en las Dependencias de la Administración Publica Federal, exceptuando el personal de honorarios, tendrán derecho al pago de una prima como complemento del salario, por cada cinco años de servicio efectivo de acuerdo a la siguiente tabla:

De 5 años a menos de 10 años:	Un quinquenio
De 10 años a menos de 15 años:	Dos quinquenios
De 15 años a menos de 20 años:	Tres quinquenios
De 20 años a menos de 25 años:	Cuatro quinquenios
De 25 años o más:	Cinco quinquenios

• Para el cómputo de la antigüedad en el servicio público, no se tomarán en cuenta las faltas injustificadas así como los períodos de licencia por cualquier motivo, exceptuando por ocupar puesto de confianza.

• Las Coordinaciones Administrativas tendrán la responsabilidad de tramitar ante la Dirección General de Recursos Humanos la regularización de los quinquenios del personal adscrito a sus áreas.

- Los trabajadores que teniendo derecho al pago de uno o más quinquenios, no fueren solicitados en su oportunidad, sólo podrán considerar en su liquidación un mínimo de un año de retroactividad, contando a partir de la fecha de su reclamación.
- Los trabajadores que tengan autorizada la compatibilidad de empleos, solo recibirán el pago de quinquenio en el empleo de mayor antigüedad.
- La documentación requerida para la regularización del pago de prima quinquenal consiste en:
  - Solicitud de pagos de quinquenio en original y dos copias.
  - Copia del último comprobante de pago.
  - Original o copia al carbón y fotocopia de hoja(s) de servicio, en caso de haber laborado en otras dependencias.
  - Para pagos retroactivos anexar copias de los comprobantes de pago del período solicitado.

II.3.1.4. Clases de usuarios del sistema actual

En el ejercicio de sus atribuciones y tareas, la Subdirección de Recursos Humanos está conformada por 30 personas trabajando en turnos matutino, vespertino y ambos.

II.3.1.4.1. Estructura organizacional

A continuación se detalla el personal adscrito a la Subdirección mencionada, desglosado en departamentos y oficinas. Para una visión más completa, remítase al organigrama de la Subdirección de Recursos Humanos (figura 2.4).

1.1.0.1.1.0.0. Subdirección de Recursos Humanos.
1 Subdirector.
1 Secretaria personal.
1 Pagador.
2 Analistas

El número que se encuentra en la esquina superior izquierda del cuadro de arriba es la clave de la ubicación, es decir, constituye la nomenclatura que se utiliza actualmente para localizar cada instancia dentro de la Subsecretaría del Ramo. Está formada por siete dígitos cuyo significado, de izquierda a derecha es el siguiente:

1er. dígito:	Secretaría de Hacienda y Crédito Público.
2o.. dígito:	Subsecretaría del Ramo.
3er. dígito:	Dirección General.
4o.. dígito:	Dirección de Area.
5o. dígito:	Subdirección.
6o. dígito:	Departamento.
7o. dígito:	Oficina.

Como el trabajo en la Dirección Técnica Operativa sólo implica Direcciones Generales adscritas a la Subsecretaría del Ramo, los dos primeros dígitos de la ubicación no se utilizan, pues siempre son 1.1.

0.1.1.1.0. Departamento de Operación y Control.
1 Jefe de Departamento.
2 Secretarias personales.
2 Analistas

0.1.1.1.1. Coordinación de Operación.
1 Jefe de Oficina.

0.1.1.1.1. Coordinación de Operación. (Continuación.)
1 Secretaria.
4 Analistas

0.1.1.1.3. Coordinación de Control.
1 Analista

0.1.1.1.4. Oficina de archivo de personal.
1 Secretaria.

0.1.1.1.5. Coordinación de documentación en trámite.
2 Gestores.

0.1.1.2.0. Departamento. de Selección, Capacitación y Desarrollo.
1Jefe de Departamento.
1 Secretaria personal.
1 Secretaria.

0.1.1.2.1. Oficina de control de asistencia.
1 Analista.
1 Secretaria.

0.1.1.2.2. Coordinación de Selección..
1 Jefe de Oficina.
2 Analistas

0.1.1.2.3. Oficina de Servicios al personal.
1 Analista

0.1.1.2.4. Coordinación de Capacitación y Desarrollo.
2 Analistas

II.3.1.4.2. Perfiles de usuarios

Mandos medios u homólogos: Son los responsables administrativos de una subdirección o jefatura de departamento. Su trabajo reside en la delegación de responsabilidades y la toma de decisiones.

- Jefes de oficina: Son responsables del trabajo operativo de su oficina. Generalmente coordinan actividades de su personal subalterno.
- Secretaría personal: Archivan y dan seguimiento a los oficios turnados a su adscripción. Elaboran oficios y contestan llamadas.
- Analistas: Comprenden al personal más heterogéneo dentro de la Secretaría, sus funciones y perfiles son amplios, aunque tienen algo en común; realizan el trabajo operativo. Los analistas aplican controles de información, elaboran listas, dan seguimiento a trámites administrativos, crean y llenan formatos computarizados, realizan capturas a los sistemas de software, emiten reportes, etc.

- Pagadores: Son los encargados de entregar al personal sus cheques y/o depósitos de sueldos, vales de despensas, estímulos, etc.
- Secretarias: Sus labores son características de la oficina a la que corresponden. Mecanografían oficios, reciben documentos, archivan, dan atención al público, etc.
- Gestores: Entregan y reciben documentos y formatos dentro y fuerá de la Secretaría.

II.3.1.4.3 Interacciones entre las clases de usuarios

Los principales usuarios de los Sistemas de Software son los analistas, generalmente ellos llevan un control de la información de su departamento u oficina.

Los analistas llegan a conocer con mayor exactitud el dominio y limitaciones de sus actividades, así como los trámites a seguir. Sin embargo, los mandos medios casi siempre imponen su visión del problema a los Sistemas de Información. Esto es un gran obstáculo, pues en ocasiones los mandos medios carecen de cultura informática, son prepotentes u ocultan información, o lo que es peor, son removidos de sus cargos ininterrumpidamente, razón por lo cual no hay continuidad en el desarrollo de Sistemas de Software.

Las secretarias raras veces son usuarias de los Sistemas de Software.

II.3.1.4.4. Otro personal involucrado

Por el control de puestos y personal, el SICPRH parece influenciar a buena parte de los sistemas de software de la Dirección Técnica Operativa, por lo menos a los sigs.:

- Sistemas de Correspondencia.
- Sistema de solicitudes de gastos a comprobar y recuperar.
- Sistema de parque vehicular.
- Sistema de solicitudes de servicio.
- Sistema de control de asistencia.
- Hojas de viáticos y pasajes en Excel.

La interacción de estos sistemas con el SICPRH se limita a sólo unos datos, a saber, el rfc, nombre del empleado y ubicación en donde labora. En los casos de las hojas en Excel se utiliza, además, el nivel del empleado y sus percepciones.

II.3.1.5. Ambiente de soporte para el sistema actual

Para la puesta en operación del SICPRH se dispone de una red de área local (LAN) Ethernet de 48 nodos, corriendo con WINDOWS NT 4.0 y NOVELL NETWARE 3.12.

La red cuenta con dos servidores de archivos, un equipo AQUANTA UNYSIS y un HP SERVER 316. Las estaciones de trabajo son microcomputadoras DELL 386, AT&T 486 y COMPAQ DESKPRO PENTIUM.

Además existe una línea telefónica para accesar servidores de otras áreas dentro de la Subsecretaría del Ramo, y una cuenta a INTERNET.

Para la implementación del SICPRH contaremos con un equipo PENTIUM y para la puesta en operación con uno o dos equipos PENTIUM, dependiendo de la demanda de microcomputadoras.

II.3.2. Notas

Algunas de las fuentes impresas de donde se recopiló lo anterior son:

- Manual de Procedimientos Administrativos de Recursos Humanos. Tomo II. Dirección General de Personal de la Oficialía Mayor de Hacienda.. 1995.
- Manual de Organización Específico de la Dirección de Técnica Operativa de la Subsecretaría del Ramo, 1997.
- Diario Oficial de la Federación.
- Condiciones Generales de Trabajo de la Secretaría de Hacienda y Crédito Público.

Capítulo III

Análisis Orientado a Objetos del SICPRH

III.1 Macroproceso

Al igual que el Análisis Estructurado (AE), el propósito del Análisis Orientado a Objetos (AOO) es describir el funcionamiento del sistema (problema) utilizando modelos. La diferencia estriba en la forma de lograrlo; mientras que el AOO busca modelar el sistema a través del reconocimiento de las clases y objetos y lo que les ocurre a éstos, el AE examina el sistema mediante el modelo clásico de entrada-proceso-salida, es decir, escudriñando el flujo de información.

Según Booch, en el AOO debemos desarrollar dos actividades básicas: el análisis de dominio y la planeación de escenarios. En el **análisis de dominio** debemos identificar las clases y objetos que conforman el sistema a modelar.

Para describir la planeación de escenarios es necesario hacer unas precisiones:

Un **punto funcional** es una descripción de algún comportamiento del sistema considerando a este último como una caja negra. Desde el punto de vista del usuario final,

un punto funcional representa una actividad primaria del sistema en respuesta a algún evento.

En el AOO indicamos la semántica de los puntos funcionales del sistema por medio de escenarios. Los **escenarios** resultan aún más importantes que los puntos funcionales puesto que describen esquemas de eventos que resaltan algún comportamiento del sistema.

Mientras que los puntos funcionales intentan describir al sistema considerando las funciones que realiza, los escenarios agrupan estas funciones para identificar los comportamientos del sistema.

Un **escenario primario** muestra comportamientos considerados clave, es decir, comportamientos esenciales del sistema; un **escenario secundario** señala cómo se comporta el sistema bajo condiciones anormales de operación.

Los escenarios como esquemas de eventos están formados por las abstracciones y mecanismos que a través de sus responsabilidades individuales y su interrelación proporcionan el comportamiento que dicho escenario intenta resaltar.

Un escenario primario está formado, al menos, por una **abstracción clave**, ésta puede ser una clase u objeto que forma parte del dominio del problema y que lo define. Una abstracción clave resalta las cosas que son parte del sistema y que son relevantes para el análisis y/o diseño, además, permite eliminar los objetos que caen fuera del sistema o que no son de interés.

Un **mecanismo** es cualquier estructura mediante la cual los objetos colaboran entre sí para obtener algún comportamiento que satisface los requerimientos del sistema, por ejemplo, piénsese en un mecanismo de freno de un auto como si fuera la manera en que tienen que trabajar los elementos de un sistema de frenado mecánico para poder detener un automóvil.

A nivel análisis se identifican los mecanismos clave por medio de la identificación de las abstracciones clave.

Hechas las consideraciones anteriores podemos decir que la **planeación de escenarios** consiste en dividir el sistema a desarrollar en esquemas de eventos (escenarios) bien elaborados, para posteriormente identificar las clases y objetos que juegan un papel importante en cada escenario. Los escenarios en su conjunto describirán al sistema total.

La planeación de escenarios incluye las siguientes tareas:

- Identificación de los puntos funcionales del sistema, y de ser posible, su agrupación.
- Desarrollo de un escenario para cada grupo de puntos funcionales. La elaboración de diagramas de objetos ilustra más claramente el escenario.
- Generación de escenarios secundarios.
- Siempre que el ciclo de vida de un objeto sea especial para un escenario, se desarrolla una máquina de estados finitos (o un diagrama de estados).

Los productos del AOO son dos; los modelos del sistema y un documento de control de riesgos que identifique las áreas que puedan afectar negativamente el proceso de diseño.

III.2 Modelo de un sistema orientado a objeto

Hemos mencionado que el producto principal de la fase de análisis OO radica en la construcción de modelos. Sin embargo, resulta importante tener una notación expresiva y bien definida, una notación estándar que haga posible para un analista o desarrollador describir un escenario o formular una arquitectura y comunicar esas decisiones de la manera menos ambigua a otros desarrolladores.

Los modelos de un sistema son complejos; la construcción de un Sistema de Software Orientado a Objeto implica comprender la estructura y funcionamiento de los objetos que componen el sistema, debemos además, entender la estructura taxonómica de las clases, los mecanismos de herencia que usan, los comportamientos del sistema total. De esta manera, podemos decir que no es posible capturar todos los detalles subyacentes a un sistema de software complejo en una sola vista.

Booch comenta que las decisiones de análisis y diseño se pueden realizar en dos dimensiones: una dimensión lógica-física y otra estática-dinámica.

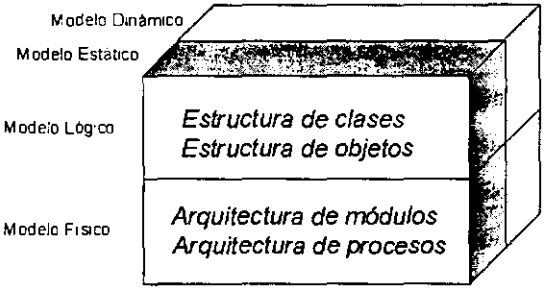


Fig. 3.1

La *vista lógica* de un sistema describe la existencia y significado de las abstracciones clave y mecanismos que forman el dominio del problema. La *vista física* describe el software concreto y los elementos de hardware del sistema a implementar.

En la segunda dimensión se toma en cuenta el comportamiento del sistema a través del tiempo. A este respecto, la *vista estática* resalta la invariación de los elementos del sistema, mientras que la *vista dinámica* introduce la noción de cambio, por ejemplo, la creación y destrucción de objetos, el cambio de estados de éstos, el paso de mensajes, etc.

La notación de Booch incluye seis diagramas diferentes para modelar estas vistas:

Diagrama	Da respuesta a:	Vista del diagrama
Diagrama de clases.	¿Qué clases existen? ¿Cómo están relacionadas las clases?	Vista lógica. Vista estática.
Diagrama de objetos.	¿Qué mecanismos son usados para la colaboración entre objetos?	Vista lógica. Vista estática.
Diagrama de módulos.	¿Dónde debe ser declarada cada clase y objeto?	Vista física. Vista estática.
Diagrama de procesos.	¿A qué procesador debe asignársele un proceso	Vista física. Vista estática.
Diagrama de transición de estados.	¿Cómo cambia el sistema con el paso del tiempo? ¿Qué estados tiene el sistema?	Vista lógica. Vista dinámica.
Diagrama de interacciones.	¿Cuál es la secuencia de eventos a través de todos los objetos del mismo escenario?	Vista lógica. Vista dinámica

A partir de este momento y conforme se vaya desarrollando el SICPRH se explicará la notación que sugiere Booch para la construcción de diagramas.

III.3. El Microproceso en el Análisis OO.

En el primer capítulo mencionamos la necesidad de conciliar el micro y macroelementos del proceso de desarrollo del software. Se dijo que el microproceso de un Sistema OO es un conjunto de actividades sencillas y diarias que realizan los desarrolladores del sistema, éstas son controladas y refinadas por actividades globales denominadas macroproceso (como puede observarse, los capítulos del presente trabajo no son más que los elementos del macroproceso de desarrollo de un sistema). El microproceso está formado por las cuatro tareas que se detallan a continuación.

III.3.1 Identificación de clases y objetos.

Esta labor constituye el primer paso del microproceso y es una de las más difíciles, la identificación de clases y objetos involucra dos actividades básicas: descubrimiento e invención. Como parte del análisis, aplicamos este paso para descubrir aquellas abstracciones que forman el vocabulario del dominio del problema, a través de esto podemos empezar a restringir nuestro sistema decidiendo lo qué es y lo qué no es de interés.

Esta actividad no es fácil, los analistas deben ser descubridores de abstracciones, deben ser capaces de observar los detalles sin perder de vista el sistema total (hay quien comenta que un buen analista no debe dejar que los árboles le tapen el bosque). De manera similar, los arquitectos del sistema y/o diseñadores deben tener habilidades para crear nuevas clases y objetos que contribuyan a la solución del sistema.

Los críticos del enfoque OO señalan que este paradigma no es robusto puesto que no hay métodos o reglas generales para la identificación de clases y objetos.

Lo anterior es cierto, en parte. Se ha señalado que no hay "recetas de cocina" en la construcción de sistemas OO. De igual manera, el seguimiento riguroso de un método estructurado por sí mismo no garantiza el éxito del proyecto. Aún más, la naturaleza creativa de cualquier tipo de diseño implica riesgos.

A pesar de que no existen reglas para identificar clases y objetos, la realización de esta actividad se basa en heurísticas y enfoques:

ENFOQUE CLASICO

La tarea de identificar clases y objetos pudiera basarse en la forma en que clasificamos las cosas. El enfoque clásico se basa en los principios de la categorización clásica, la cual supone que todas las entidades que tienen una propiedad o conjunto de propiedades en común forman una categoría (podemos pensar que todos los animales que nacen del vientre de su madre forman la categoría mamíferos, por ejemplo).

Así entonces, el enfoque clásico considera que las clases y objetos se pueden identificar por los requerimïentos del sistema. Por ejemplo, éstos pueden provenir de las sigs. fuentes:

- Cosas tangibles (carros, sensores, personas)
- Roles (maestro, político, operador)
- Eventos (una requisición de artículos de almacén)
- Interacciones (un préstamo, una factura, etc.)

Para el modelado de Bases de Datos, podríamos tener objetos como:

- Gente.
- Lugares.
- Organizaciones.
- Empresas.

ANALISIS DE COMPORTAMIENTO

Mientras que el enfoque clásico enfatiza las cosas tangibles, existen otros enfoques de Análisis OO que consideran que el comportamiento dinámico del sistema es la principal fuente de clases y objetos. De esta manera se forman clases basadas en grupos de objetos que exhiben un comportamiento similar, tienen responsabilidades en común y forman jerarquías de clases.

En el enfoque de análisis de comportamiento intentamos entender lo que ocurre en el sistema a modelar. Para lograr esto identificamos los principales comportamientos del sistema. Después se asignan estos comportamientos a las partes del sistema que son encargadas de realizar la acción y se identifican los elementos que inician y participan en estos comportamientos. Así, iniciadores y participantes que juegan roles significativos son reconocidos como objetos y se les asigna una responsabilidad bien definida.

ANALISIS DE DOMINIO

Los enfoques anteriores son atribuibles a aplicaciones específicas. El análisis de dominio sugerido por Neighbors[1] intenta identificar clases y objetos que son *comunes a todas las funciones que forman parte de un mismo dominio.*

El análisis del dominio es útil para señalarnos las abstracciones clave que han resultado útiles en otros sistemas relacionados, dando al analista ideas de las abstracciones pertinentes en el sistema que se está diseñando.

TECNICA *USE CASE*

Si se ven de manera aislada, las prácticas del análisis clásico, análisis del comportamiento y análisis del dominio dependen en gran medida de la experiencia personal por parte de los analistas. Esto es inaceptable para la mayoría de los proyectos porque no hay forma de determinar o predecir el éxito.

Sin embargo, podemos complementar los enfoques anteriores para obtener un proceso de análisis más confiable. Tal práctica es conocida como la técnica USE CASE (*Use Case Analysis*).

[1] Mencionado por Booch, op. cit., p.p.157

Se puede aplicar esta técnica tan pronto como se tengan identificados los escenarios fundamentales del sistema. Este análisis estudia el sistema por medio de guiones (*storyboarding*) para cada escenario, similares a los que se usan en la televisión o en el cine.

Conforme el equipo de análisis avanza en la asimilación de un escenario, se deben identificar los objetos que participan en él y sus responsabilidades. En una etapa más avanzada se expanden los escenarios iniciales para considerar las condiciones menos importantes y, por último, las situaciones anormales de operación (escenarios secundarios). Los resultados de estos escenarios secundarios pueden introducir nuevas abstracciones y/o modificar o reasignar responsabilidades de abstracciones ya existentes.

TECNICA CON TARJETAS CRC[2]

El uso de tarjetas CRC es una forma efectiva de analizar escenarios. La tarjeta no es más que una carta (como las fichas bibliográficas) en donde el equipo de analistas escribe con lápiz el nombre de la clase candidata (en la parte superior), sus responsabilidades (en el centro) y los colaboradores de esa clase (en la parte inferior de la tarjeta). Se usa una tarjeta para cada clase marcada como relevante para el escenario.

Conforme los analistas avancen en el estudio del escenario en cuestión, pueden:

- Agrupar nuevas responsabilidades a una clase existente.
- Agrupar ciertas responsabilidades para formar una nueva clase.
- Dividir las responsabilidades de una clase en grupos más pequeños, y quizás, distribuirlas a clases diferentes.

[2] Tarjetas CRC ó tarjetas de clases/responsabilidades/colaboradores.

DESCRIPCION INFORMAL EN ESPAÑOL

Esta técnica para la identificación de clases y objetos sugiere que se realice una descripción, en Español, del problema a manera de prosa o de enunciados. Enseguida se subrayan los sustantivos y verbos empleados.

De esta manera se puede decir que los sustantivos son candidatos a objetos y los verbos son candidatos a operaciones de esos objetos.

Desdichadamente, el lenguaje humano es terriblemente impreciso, de tal forma que la calidad de los objetos y operaciones así producidos dependen de las habilidades de escritura y redacción de los analistas.

GENERALIDADES DE LOS ENFOQUES ANTERIORES

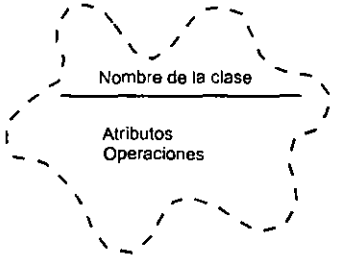
Los enfoques y técnicas que se han descrito pueden usarse en cualquier combinación o de manera aislada para la identificación de clases y objetos. El producto central en este primer paso del microproceso de desarrollo es la creación de un diccionario de datos que contenga un amplio conjunto de abstracciones, con nombres consistentes y una sencilla separación de responsabilidades.

La notación de Booch[3] utiliza los diagramas de clases para mostrar la existencia de clases y sus relaciones. Así, cada clase es expresada como una nube con bordes discontinuos teniendo en la parte superior el nombre de la clase e indicando, en la parte inferior del símbolo, sus atributos y operaciones. Ambas partes de la clase son separados por una línea continua (véase la fig. 3.2).

A un nivel más alto, es posible expresar las clases del sistema como conjuntos de clases, es decir, como categorías de clases. Una categoría de clases se expresa por medio de un

[3] Toda la simbología que se explica en este trabajo corresponde a la notación de Booch para la construcción de diagramas para AOO y DOO.

rectángulo. En la parte superior de la figura se indica el nombre de la categoría de clases y, separados por una línea, se enlistan abajo las clases miembro de la categoría. Booch menciona que no es necesario expresar todas las clases miembro, aún más, es válido no expresar ninguna; en este caso sólo se escribe el nombre de la categoría de clases.[4]



Nombre de la categoría de clases

Clases miembro

Fig. 3.2

Los diagramas de objetos se usan para mostrar la existencia de objetos y sus relaciones en la vista lógica del sistema. Cada diagrama de objetos representa las interacciones o relaciones estructurales que se podrían dar entre un conjunto de instancias de clases. El símbolo utilizado para representar a un objeto es el de una nube con bordes continuos. El nombre del objeto se coloca en la parte superior de la nube y, separados por una línea, se enlistan los atributos del objeto.

Los nombres de objetos se pueden expresar en cualquiera de las sigs. formas:

- Indicando sólo el nombre del objeto, en este caso la clase a la que pertenece el objeto es anónima (por ejemplo, **A**)

[4] Véase Booch Object Oriented Analysis and Design op.cit. cap. V, para una descripción más detallada de la notación y construcción de diagramas.

- Indicando sólo el nombre de la clase a la que pertenece el objeto, de esta manera la instancia se considera anónima (por ejemplo: **C**).
- Indicando el nombre del objeto y su clase (por ejemplo **A:C**)

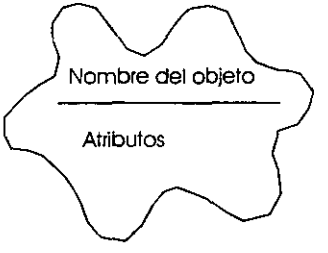


Fig. 3.3

III.3.2 Identificación de la semántica de clases y objetos

Una vez identificadas las principales clases y objetos en la microactividad anterior, el siguiente paso consiste en establecer el comportamiento, atributos y responsabilidades para cada abstracción. El propósito de identificar la semántica[5] de clases y objetos consiste en establecer el comportamiento y atributos de cada abstracción identificada en la fase previa.

Esta microactividad tiene varios productos. El primero es el refinamiento del diccionario de datos, creando las especificaciones para cada abstracción. Estas especificaciones constituyen el **protocolo** para cada clase puesto que describen las operaciones que la abstracción puede realizar.

[5] En Gramática se define a la Semántica como el estudio del significado de las palabras. En nuestro caso podemos decir que la semántica de nuestras clases y objetos describe el significado de éstos por medio del conocimiento de su estructura, atributos, comportamiento y responsabilidades en interrelación con otras clases y objetos.

Se pueden elaborar diagramas de objetos y de interacciones que expresen semánticas a través de libretos o "guiones" de comportamiento para cada escenario (*storyboarding*). Tal actividad podría incluir los sigs. eventos:

- Seleccionar un escenario e identificar las abstracciones relevantes.
- Observar la actividad del escenario y asignar responsabilidades a cada abstracción de forma que sea suficiente para producir el comportamiento deseado.
- Asignar los atributos a las abstracciones de manera que les permitan cumplir las responsabilidades asignadas.
- Reubicar responsabilidades de tal forma que exista una distribución de comportamiento balanceado pero sin alterar la integridad de las abstracciones.

En el análisis debemos ubicar responsabilidades para los diferentes comportamientos del sistema utilizando los enfoques que se detallaron en el paso anterior.

El producto de esta etapa es el refinamiento del diccionario de datos creando las especificaciones para cada abstracción. Además de lo anterior podemos obtener diagramas de objetos y diagramas de interacciones que capturen la semántica de los escenarios principales del problema.

Esta fase se considera concluida satisfactoriamente cuando tengamos un conjunto de responsabilidades y operaciones suficientes y completas para cada abstracción identificada en la etapa previa.

III.3.3 Identificación de las relaciones entre clases y objetos.

En esta etapa se consolidan los límites del sistema y se reconocen los colaboradores de cada abstracción identificada antes.

Booch identifica dos tipos de relaciones entre objetos:

- Las *ligas* (*links*) son conexiones físicas o conceptuales entre objetos que les permite colaborar entre sí; una liga denota una asociación a través de la cual un objeto (cliente) solicita los servicios de otro objeto (servidor).
- La *agregación* denota una jerarquía de "parte-totalidad" (*whole/part*) en donde un objeto contiene física o conceptualmente a otro u otros. Al objeto "total" (el más grande) se le conoce como el objeto agregado, mientras que los objetos contenidos son los atributos del primero.

Ambos tipos de relaciones son ventajosos, la agregación es capaz de encapsularse escondiendo los secretos de su implementación, las ligas permiten un acoplamiento débil entre objetos.

Un enlace (*link*) entre dos objetos puede existir si y sólo si hay una asociación entre sus respectivas clases. En los diagramas de objetos tales enlaces se expresan por medio de líneas que unen objetos. Arriba de cada enlace se coloca una flecha que apunta a la dirección de invocación, además se coloca el nombre de la operación o evento que representa el enlace y opcionalmente se indica un número de secuencia, el cual denota el orden en el que ocurren los eventos.

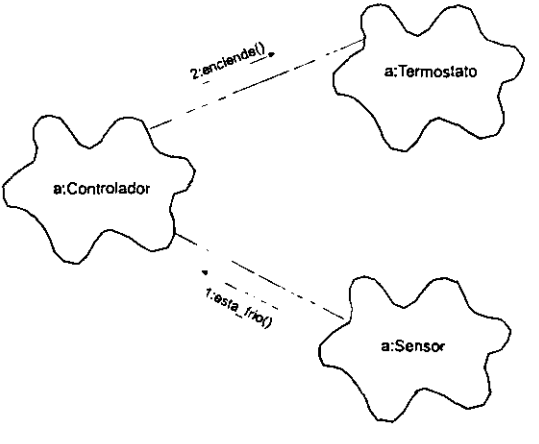


Fig. 3.4

El diagrama anterior comienza cuando el objeto **a** de la clase **sensor** envía el mensaje *esta_frio()* al objeto **a** de la clase **controlador**, éste, a su vez, envía la operación *enciende()* al objeto **a** de la clase **termostato**.

Como se puede observar, los diagramas de objetos son muy útiles puesto que indican la semántica de escenarios (primarios y secundarios). Además, como parte del diseño, ilustran la semántica de los mecanismos del sistema.

Por lo que respecta a relaciones entre clases, Booch considera tres tipos básicos:

- *Herencia (Generalización-Especialización).* Significa una relación del tipo "es un". Por ejemplo, una naranja "es un" tipo de cítrico, lo cual significa que una naranja es una subclase especializada de la clase general cítrico.
- *Agregación (Parte-Totalidad).* Consiste en una relación del tipo "tiene". De esta manera, por ejemplo, podemos decir que la naranja no es una pulpa sino que una naranja "tiene" pulpa.
- *Asociación.* Es la relación más débil, significa dependencia semántica entre clases que aparentemente no tienen relación alguna. Por ejemplo, pensemos en nuestra naranja y un pan tostado, ambas clases parecen ser independientes, pero podrían representar los alimentos del desayuno.

Además de las anteriores, se pueden identificar otros tipos de relaciones entre clases:

- *Cliente-Servidor (Relación por uso)* Es un tipo de relación de asociación, pero más refinada, en donde una clase "usa" los recursos de otra sin ser parte de ella. Podemos decir, como ejemplo, que la clase María "usa" a las naranjas para poder alimentarse.
- *Instanciación.* Es la relación que existe entre una clase que genera a otra (de la misma forma en que una clase instancia a un objeto). Pensemos en contenedores de clases, clases plantillas para la generación de otras clases, etc.

- *Metaclases*. Denota la relación de una "clase de clases" con sus clases miembro.

En los diagramas de clases utilizamos los sigs. símbolos para expresar relaciones:

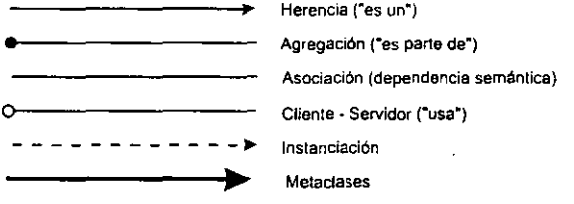


Fig. 3.5

El diagrama de la figura 3.6 constituye un ejemplo sencillo que muestra las principales relaciones entre clases.

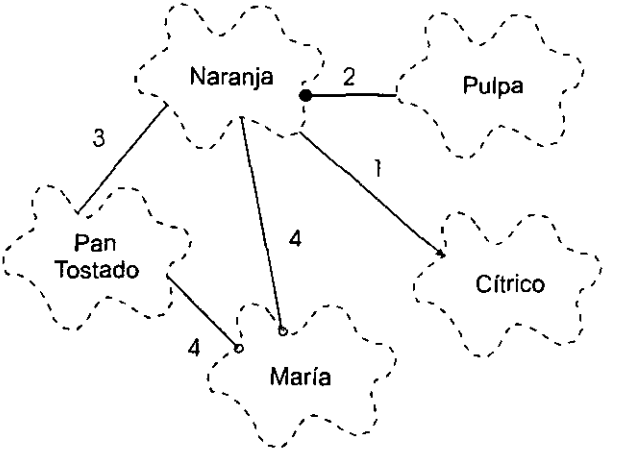


Fig. 3.6

El diagrama de clases[6] de la página anterior identifica las siguientes relaciones:

1. La clase naranja "es un" cítrico, o lo que es lo mismo, naranja es una subclase de cítrico.
2. La clase naranja "tiene" a la clase "pulpa", o naranja es un agregado que contiene a pulpa.
3. La clase naranja y la clase pan tostado complementan el desayuno (dependencia semántica).
4. La clase María "usa" a la clase naranja y a la clase pan tostado para alimentarse

Hemos hablado de relaciones entre objetos y entre clases, pero ¿Cuál es la relación entre clases y objetos? Las clases y objetos son conceptos íntimamente unidos. Hablando específicamente podemos decir que un objeto tiene una relación con alguna clase puesto que cada objeto es instancia de alguna clase.

Por otra parte, la etapa de identificación de las relaciones entre clases y objetos tiene como productos algunos diagramas, tales como los diagramas de clases, de objetos y de módulos.

III.3.4. Implementación de clases y objetos.

En este nivel de análisis OO, el propósito de implementar clases y objetos consiste en obtener un refinamiento de las abstracciones existentes, de tal forma que sea suficiente para descubrir nuevas clases y objetos en el siguiente nivel de abstracción.

[6] En el ejemplo anterior se han puesto etiquetas (números) sobre las líneas que unen a las clases exclusivamente para poder explicar el significado de cada relación. Normalmente esto no se hace.

Los productos de esta etapa representan decisiones acerca de la representación de cada abstracción y el mapeo de esas representaciones hacia un modelo físico. Para mostrar estas relaciones hacemos uso de los diagramas de módulos. Además debemos actualizar nuestro diccionario de datos.

Consideramos terminada esta etapa cuando hayamos identificado todas las abstracciones necesarias para satisfacer las responsabilidades de alto nivel de las abstracciones identificadas durante el primer ciclo del microproceso.

III.4. Análisis Orientado a Objeto del SICPRH

Como hemos mencionado, el análisis de un sistema OO es una fase del macroproceso de desarrollo que tiene dos tareas fundamentales: el análisis de dominio y la planeación de escenarios. La construcción de ambas tareas es soportada por las actividades diarias conocidas como microproceso.

Utilizando la teoría que se ha comentado, empezaremos a analizar el Sistema Integral de Control de Puestos y Recursos Humanos

III.4.1. Análisis de Dominio

En el análisis de dominio del SICPRH hemos identificado las abstracciones que definen el conjunto del sistema a implementar. Tras la aplicación de más de una vez de las actividades del microproceso podemos hacer un sumario a manera de diccionario de datos de las abstracciones del sistema[7].

[7] Se ha dicho que el AOO es un proceso iterativo e incremental. Por esta razón, las clases que se van a detallar no han sido resultado de la aplicación de la primera actividad del microproceso, ni siquiera de la aplicación de las cuatro tareas del microproceso coordinadas por el análisis del dominio; al contrario, han sido enriquecidas por las actividades del microproceso en función del análisis del dominio y, posteriormente, refinadas en la planeación de escenarios.

NOMBRE:
Subsecretaría de Hacienda y Crédito Público[8]
DEFINICIÓN:
Clase abstracta.
RESPONSABILIDADES:
Las que le encomienda el Poder Ejecutivo.
ATRIBUTOS:
Areas

NOMBRE:
Areas
DEFINICIÓN
Areas presupuestales
RESPONSABILIDADES:
Soportar las actividades de la Subsecretaría del Ramo.
ATRIBUTOS:
Cpl Identificador (cve. de área presupuestal).
Descripción
Ubicación
Puesto

NOMBRE:
Ubicación
DEFINICIÓN:
Lugar (oficina) donde el personal desempeña sus labores.
RESPONSABILIDADES:
Administrar plazas y recursos humanos.
ATRIBUTOS:
Ubica Identificador de ubicación (cve. de ubicación).
Descripción
Plaza

Iniciamos este análisis de dominio estableciendo las clases de nivel más alto; la clase llamada Subsecretaría de Hacienda y Crédito Público es precisamente la de mayor nivel, es la que da origen al SICPRH. Esta clase es una clase abstracta, es decir, una clase que carece de instancias.

[8] Aunque no se especifique explícitamente, todas las clases que nombraremos tienen al menos dos operaciones, generar() y borrar(). Éstas son el equivalente al constructor y destructor de la clase.

La clase que está más próxima en nivel a la Subsecretaría de Hacienda y Crédito Público es la clase Areas. En el mundo real la Subsecretaría del Ramo, para el cumplimiento de sus funciones delega sus tareas a ciertas áreas presupuestales (también conocidas como Direcciones Generales), cada una de ellas tiene sus actividades específicas, sus propias plazas y sus recursos humanos.

En la especificación de las clases se puede observar que la clase áreas posee cuatro atributos, los dos primeros son simples (identificador y descripción), los dos últimos son clases (ubicación y puesto).

Areas tiene 9 instancias, a saber:

- 200 Oficina del C. Subsecretario del Ramo.
- 210 Unidad de Planeación del Desarrollo.
- 211 Dir. Gral. de Planeación Hacendaria.
- 212 Dir. Gral. de Crédito Público.
- 213 Dir. Gral. de Banca de Desarrollo.
- 214 Dir. Gral. de Banca Múltiple.
- 215 Dir. Gral. de Seguros y Valores.
- 216 Dir. Gral. de Asuntos Hacendarios. Internacionales.
- 217 Dir. Gral. de Política de Ingreso.

En un nivel más bajo encontramos a la clase Ubicación. Cada área presupuestal está dividida en Direcciones de Area, subdirecciones, oficinas, etc., éstas se pueden localizar en un mismo inmueble o en diferentes puntos de la ciudad; a estas entidades organizacionales se les llama ubicaciones. La Dirección Técnica Operativa, que es una Dirección de Area perteneciente al área presupuestal 200, es una instancia de la clase ubicación.

Aunque las ubicaciones sólo tienen tres atributos (un identificador, su descripción y la clase plaza) su importancia va más allá de esto. Las ubicaciones mantienen una relación laboral con el personal de la Subsecretaría del Ramo. Las ubicaciones concentran el trabajo

de los empleados. Algunas de ellas (como la Dirección Técnica Operativa) administran plazas y recursos humanos. Puesto que la clase ubicación define lo que es de interés para el SICPRH (plazas, personal, relación de trabajo) podemos decir que ésta es una abstracción clave.

Cada área presupuestal tiene una cierta cantidad y tipo de puestos. Los cuales dan origen a las plazas que son asignadas al personal trabajando para cada ubicación. En un nivel de abstracción más bajo se han descubierto las siguientes clases:

```
NOMBRE:
 Puesto
DEFINICIÓN:
 Cada puesto o cargo tiene un nivel de responsabilidad dentro de la
 organización.
RESPONSABILIDADES:
 Normalizar las plazas.
ATRIBUTOS:
 CP2                  Primer subidentificador presupuestal del puesto.
 CP3                  Segundo subidentificador presupuestal del puesto.
 Descripción          Descripción del puesto.
 Nivel                Nivel del puesto de acuerdo a un tabulador.
 Subnivel             Subnivel de puesto para los niveles 27
 Tipo_plaza           Tipo de puesto: Confianza, Base u Honorarios.
 Grupo                Gpo. al que pertenece: operativo, enlace, etc.
 FIEF                 Control utilizado por la DGRH.
 Sueldo_Neto          Sueldo neto para ese puesto.
 Productividad        Bono de productividad para ese puesto.
OPERACIONES
 Obt_datos            Obtener datos generales.


NOMBRE:
 Plaza
DEFINICIÓN:
 Cada plaza es única y corresponde a un tipo de puesto. Los gastos de las
 plazas por efectos de salario son presupuestados anualmente.
RESPONSABILIDADES:
 Crear un contrato de trabajo entre el personal y la Subsecretaría.
ATRIBUTOS:
 Los que hereda de la clase puesto y los sigs.:
 Folio                Identificador de la plaza.
 CP1                  Area presupuestal de la plaza.
 CP4                  Número de la plaza.
```

```
 Situacion              Situación (p.e. Congelada por licencia)
 Kardex                 Historial de movimientos de la plaza.
 Observaciones
OPERACIONES:
 Las que hereda de la clase puesto y las sigs.:
 generar()              Crear una instancia de la clase plaza.
 borrar()               Eliminar la instancia.
 asignar()              Asigna personal a una plaza vacante.
 descargar()            Desasigna al personal de su plaza.
 en_licencia()          Desactiva la plaza al ponerla en licencia.
 fin_licencia()         Activa la plaza al terminar la licencia.
 cambiar()              Cambia la plaza a otra área presupuestal.
 verificar()            Verifica si la plaza está vacante.
```

La clase Puesto proviene de un catálogo de puestos para empleados federales; los niveles de cada puesto son los mismos para cualquier Secretaría de Estado, no asi los salarios y montos por productividad, los cuales son homólogos sólo dentro de la S.H.C.P.

La clase plaza es una subclase de la clase puesto. Piénsese en la instancia de puesto denominada 'Director de área'; en la Subsecretaria podrían haber nueve plazas de la misma clase base puesto con el mismo puesto (nueve plazas de directores de área).

Una plaza es asignada a una persona. Sin embargo, en el dominio del SICPRH sólo nos interesan aquellas personas que mantienen un contrato de trabajo con la Secretaría, es decir, del ámbito de personas que son empleadas y los hijos de los empleados. Existe pues, una relación de herencia entre las clases persona, empleado e hijos (se sobreentiende que esta última es hijos de empleados hacendarios).

Por la cantidad de variantes y por su interrelación podemos decir que las clases empleado y plaza son relevantes para esta etapa de AOO, por este motivo las hemos identificado como abstracciones clave del SICPRH.

Iniciaremos el estudio de la clase persona definiendo sus propiedades:

NOMBRE:

Persona

DEFINICIÓN:

Cualquier persona.

RESPONSABILIDADES:

Clase abstracta.

ATRIBUTOS:

Rfc	Registro Federal de Contribuyentes.
Nombre	Nombre de pila.
Paterno	Apellido paterno.
Materno	Apellido materno.
Domicilio	Calle, número, colonia, delegación, C.P.
Telefono	Teléfono personal.
Sexo	Sexo.
Edo_civil	Estado civil.
F_nac	Fecha de nacimiento.

NOMBRE:

Empleado

DEFINICIÓN:

Cualquier persona que tenga un vínculo laboral con la Subsecretaría del ramo.

RESPONSABILIDADES:

Las que le asigna las Condiciones Generales de Trabajo de la Secretaría de Hacienda y Crédito Público.

ATRIBUTOS:

Los que hereda de la clase persona y los sigs:

Situacion	Ultimo movimiento del empleado por ejemplo: promoción, nuevo. ingreso.
F_efectiv	Fecha de efectividad del último movimiento del empleado.
Quinquenio	Número de quinquenios.
Funcion	Actividades que realiza el personal.
Importe	Bonificación extra al empleado.
N_cuenta	Número de cuenta de cheques (depósito bancario para pagar sueldos).
Banco	Banco que maneja la cuenta de cheques.
Homoclave	Adición al rfc para manejo de SAR.
Banco_SAR	Banco que maneja el SAR del empleado.
S_social	Número de seguridad social (Filiación al ISSSTE).
Sindicato	Estado de cotización al S.N.T.H.
Turno	Turno de labores.
F_gobierno	Fecha de ingreso al Gobierno Federal.
F_shcp	Fecha de ingreso a S.H.C.P.
Folio_hgo	Folio del contrato de seguro de vida con Aseguradora Hidalgo.
Fonac	Estado de inscripción al FONAC
Escolaridad	Nivel último de escolaridad
Idiomas	Idiomas.

Cursos	Cursos aprobados.
Hijo	Hijos.
Incidencias	Incidencias (faltas, justificantes, etc.).
Kardex	Historial de movimientos del empleado.

OPERACIONES:

generar()	Creación de una instancia de la clase empleado.
borrar()	Destrucción del objeto.
obt_datos()	Obtener datos generales personales.
en_licencia()	Suspensión del objeto del tipo empleado.
fin_licencia()	Reanudación de labores de la instancia de empleado.
cambiar()	Cambiar al objeto de ubicación.
Act_fecha()	Actualizar fecha para los casos de reingresos.

NOMBRE:
Hijo
DEFINICIÓN:
Hijos del personal hacendario.
RESPONSABILIDADES:
Ninguna. (sólo beneficios)
ATRIBUTOS:
Los que hereda de la clase persona.
OPERACIONES:

generar()	Crear una instancia de hijo.
borrar()	Eliminar la instancia de hijo.
obt_datos()	Obtener datos generales.

Se ha comentado la necesidad de llevar un histórico de todos los movimientos de plazas y personal. La abstracción que se encargará de esta tarea es la clase kardex.

NOMBRE:
Kardex
DEFINICIÓN:
Historial de movimientos de personal y/o plaza.
RESPONSABILIDADES:
Mantener un histórico de movimientos.
ATRIBUTOS:

Folio	Identificador de plaza.
Rfc	Registro Federal de Contribuyentes del personal.
Nombre	Nombre del personal.
Paterno	Apellido paterno.
Materno	Apellido materno.
Ubicación	Ubicación del personal.

CP1	Clave del área presupuestal de la plaza
CP2	primer subidentificador presupuestal del puesto.
CP3	segundo subidentificador presupuestal del puesto.
CP4	Clave presupuestal de la plaza (identificador de plaza)
Nivel	Nivel del puesto de acuerdo a un tabulador.
Subnivel	Subnivel de puesto para los niveles 27.
Tipo_mov	Situación de la plaza o del personal.
F_efectiv	Fecha de efectividad del último movimiento de personal o de la plaza.
Observaciones	

OPERACIONES:

actualizar()	Actualizar movimiento en kardex.

Nuestro análisis no puede estar completo si no se toma en cuenta el papel que juegan los bancos en cuanto al manejo de cuentas de cheques para cobro de salarios, SAR y FONAC.

NOMBRE:

Banco

DEFINICIÓN:

Institución de Banca Múltiple

RESPONSABILIDADES:

Manejo de cuentas.

ATRIBUTOS:

Cve_banco	Clave de la institución bancaria
Descripcion	Razón social del banco.

La relación entre las clases citadas arriba puede expresarse en un diagrama de clases.

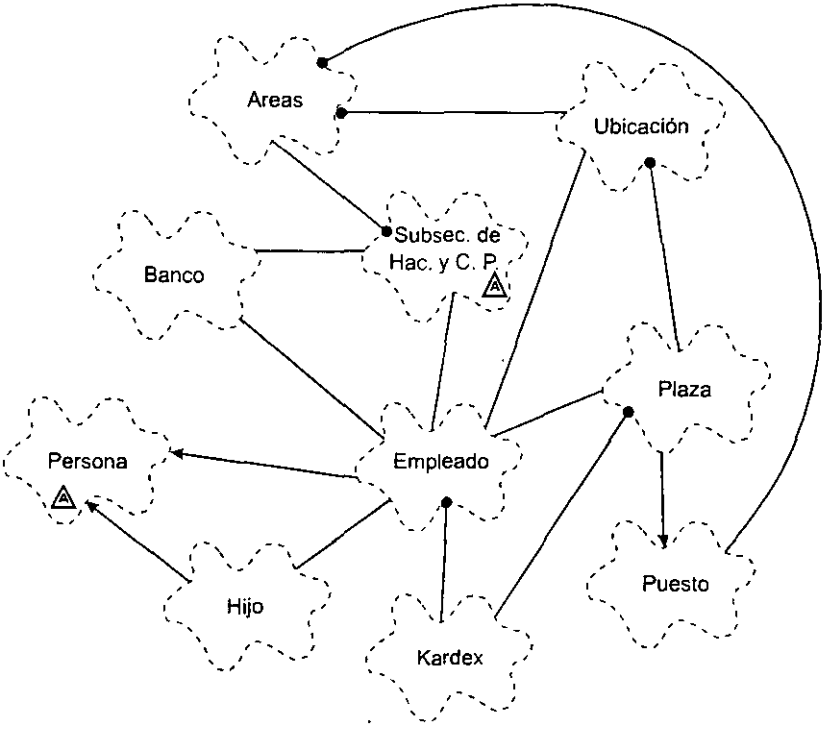


Fig. 3.7

En el diagrama anterior podemos observar la relación entre las abstracciones previamente identificadas. Se puede notar que todo está íntimamente relacionado. Sin embargo, la clase Subsecretaría de Hacienda y Crédito Público da forma a toda la semántica del dominio del SICPRH; esta clase es un agregado que contiene directamente a la clase áreas y como la clase áreas contiene directamente a ubicación y puesto e indirectamente a plaza, podemos decir que la clase abstracta Subsecretaría del ramo contiene también a las clases ubicación, puesto y plaza.

La clase abstracta Subsecretaría de Hacienda y Crédito Público tiene, además, una asociación (o dependencia semántica) con banco y empleado.

La clase puesto es una superclase de plaza (puesto hereda a plaza).

Kardex es una abstracción que es parte de empleado y al mismo tiempo, es parte de la plaza.

La clase empleado y la clase hijo son subclases de la clase persona. La clase empleado como abstracción clave es muy importante, el diagrama anterior muestra que ésta tiene una dependencia semántica con ubicación, con banco y con plaza, y por lo mismo, tiene una asociación con la Subsecretaría del ramo. La clase hijo es importante por su relación semántica con empleado, en el mundo real no nos interesan los hijos cuyos padres no son empleados hacendarios.

No obstante el descubrimiento de estas relaciones, es necesario un estudio más profundo del significado completo que tiene la abstracción empleado en el contexto del SICPRH. Este estudio muestra el descubrimiento de las siguientes clases:

```
NOMBRE:
 Escolaridad
DEFINICIÓN:
 Nivel último de escolaridad del empleado
RESPONSABILIDADES:
 Registrar información de niveles de escolaridad y profesiones para uso
 en la definición de funciones de personal y perfiles de empleados.
 Posible uso de la información en movimientos de promociones (ascensos).
ATRIBUTOS:
 Cve_nivel              Nivel de escolaridad.
 Descripción            Descripción del nivel de escolaridad.
 Porcentaje             Porcentaje de créditos cubiertos en el nivel de
                        escolaridad
 Instituto              Institución educativa que respalda el nivel de
                        escolaridad.
OPERACIONES:
 obt_datos()            Obtener datos generales.


NOMBRE:
 Idioma
DEFINICIÓN:
 Idiomas que domina el empleado
RESPONSABILIDADES:
 Registrar información concerniente a idiomas.
```

```
ATRIBUTOS:
 Cve_idioma              Clave del idioma.
 Descripción             Descripción del idioma.
 Porcentaje              Porcentaje de dominio del idioma
 Instituto               Institución educativa donde se aprendió el idioma.
OPERACIONES:
 Obt_datos()             Obtener datos generales.


NOMBRE:
 Cursos
DEFINICIÓN:
 Registro participación del empleado en cursos y diplomados
RESPONSABILIDADES:
 Llevar una contabilidad de cursos de capacitación
ATRIBUTOS:
 Tipo                    Curso o diplomado.
 F_inicio                Fecha de inicio.
 F_termino               Fecha de fin de curso.
 Instituto               Institución educativa
 Estado                  Aprobado o reprobado.
 Observaciones
OPERACIONES:
 generar()               Crear una instancia de la clase cursos.
 obt_datos()             Obtener datos generales.


NOMBRE:
 Incidencias
DEFINICIÓN:
 Registro de todo tipo de faltas de asistencia (Falta a la entrada, a la
 salida y faltas totales), así como el uso de los justificantes del tipo
 días económicos, licencias médicas y cuidados maternos.
RESPONSABILIDADES:
 Control de incidencias para efecto de recuperación de salarios no
 devengados.
ATRIBUTOS:
 Tipo                    Tipo de incidencia.
 F_inicio                Fecha de inicio.
 F_termino               Fecha de fin de la incidencia.
 Estado                  ¿Aplicación de descuento?.
 Observaciones
OPERACIONES:
 generar()               Crear una instancia de incidencias.
 obt_datos()             Obtener datos generales.
```

La definición de estas últimas clases permite describir la semántica de la clase empleado dentro del dominio del SICPRH.

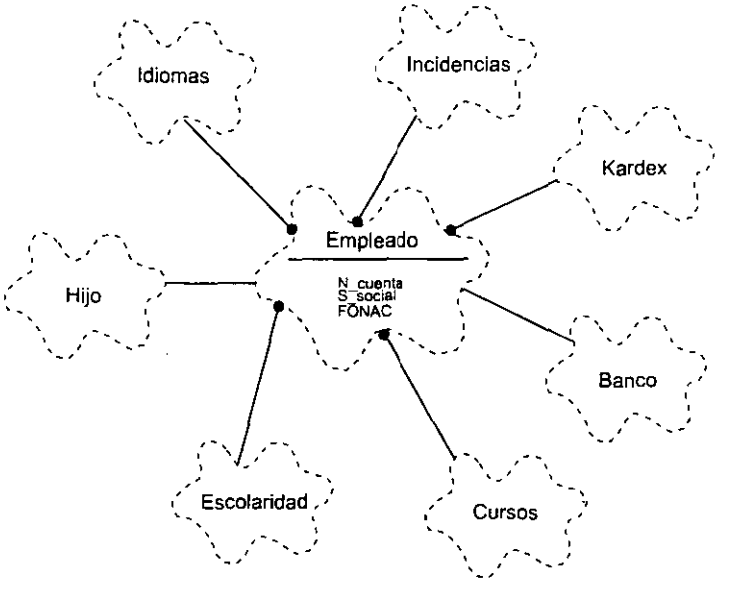


Fig. 3.8

La figura 3.8 es el diagrama de clases correspondiente a la estructura de la clase empleado. Por cuestiones de espacio sólo se indicaron tres atributos s_social (seguridad social, filiación al ISSSTE), n_cuenta (número de cuenta de cheques) y FONAC (estado de inscripción al Fondo de Ahorro Capitalizable).

Como puede observarse, las clases escolaridad, idiomas, cursos, incidencias y kardex forman parte de la clase empleado, la clase banco mantiene una asociación con empleado y la clase hijo mantiene una relación semántica con esta última.

III.4.2 Planeación de escenarios

Ahora que hemos establecido los limites del sistema definiendo las abstracciones que forman el dominio del SICPRH, continuaremos nuestro análisis estudiando los comportamientos del sistema a través de la asimilación de los diferentes escenarios de interés.

III.4.2.1 Escenarios primarios

Empezaremos enumerando los escenarios primordiales del dominio del SICPRH.

1. Suspensión de efectos de nombramiento (licencias).
2. Terminación de efectos de nombramiento (bajas).
3. Cambio de situación de personal federal: reanudación de labores.
4. Movimientos de personal: nuevo ingreso.
5. Movimientos de personal: reingreso.
6. Movimientos de personal: cambio de adscripción.
7. Movimientos de personal: promoción.
8. Movimientos de personal: reubicación.
9. Conversión de plazas-puesto.
10. Registro de incidencias.
11. Capacitación y desarrollo.

Un escenario integra objetos, eventos y puntos funcionales (los puntos funcionales son las respuestas del sistema hacia algún evento). La manera más cómoda de expresar la semántica (el significado) de un escenario es por medio de diagramas de interacciones.

Los diagramas de interacciones son usados para conocer el ambiente de un escenario en el mismo contexto de un diagrama de objetos. La ventaja de usar diagramas de interacciones consiste en que es más fácil leer el paso de mensajes en relativo orden.

Escenario 1. Suspensión de efectos de nombramiento.

Podemos definir gráficamente el escenario correspondiente a licencias de la sig. forma:

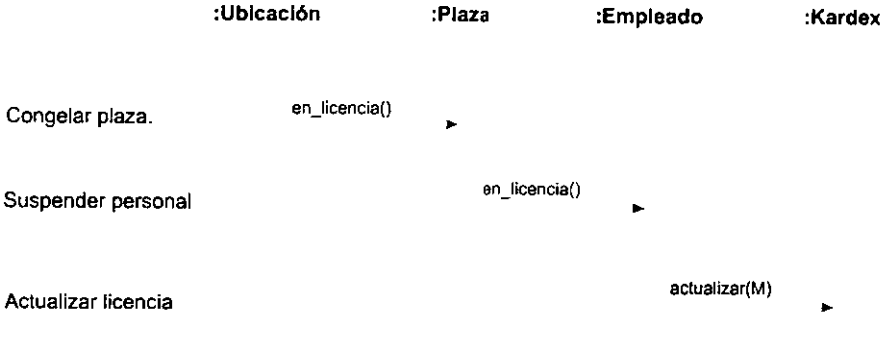


Fig. 3.9

Como se puede notar, los diagramas de interacción aparecen en forma tabular. Las entidades de interés (objetos) son escritas horizontalmente en el encabezado del diagrama. Una línea punteada es dibujada bajo cada objeto.

Los mensajes (eventos u operaciones) se muestran horizontalmente representados por líneas que conectan el objeto cliente con el objeto servidor. El orden de los mensajes dentro del escenario se lee desde la parte superior del diagrama hasta la inferior. Los diagramas pueden contener *scripts* en la parte izquierda, su misión consiste en facilitar la interpretación del diagrama.

La interpretación que se puede concluir del diagrama de arriba es la siguiente: el escenario Suspensión de efectos de nombramiento comienza cuando una instancia anónima de la clase ubicación invoca la operación en_licencia() sobre un objeto anónimo de plaza. Una vez que la instancia de plaza se ha puesto en licencia, ésta envía una petición a una instancia de empleado para que haga lo mismo. Cuando la instancia de empleado se ha marcado en licencia, ésta solicita al kardex que se actualice el movimiento mediante la operación actualizar(M). El parámetro que se proporciona a la instancia de kardex es el tipo de movimiento, en este caso, licencia sin goce de sueldo, licencia a medio sueldo o licencia con goce de sueldo.

Escenario 2. Terminación de efectos de nombramiento (bajas)

Como se puede observar en el diagrama de la figura 3.10, el escenario comienza cuando un objeto de la clase ubicación le pide a alguna plaza que termine la relación que tiene con su empleado asignado mediante la operación descargar(). Cuando la plaza destruye su asignación con el empleado, ésta envía una respuesta a la instancia de ubicación a través del mensaje estado, indicándole que esta plaza se ha convertido en vacante.

La misma instancia de plaza le pide a kardex, mediante actualizar(M), que registre el movimiento con el tipo de baja como parámetro. Este último podría ser una baja por

renuncia, defunción, jubilación, retiro, baja por falta de probidad u honradez, por acuerdo superior o por resolución del Tribunal de conciliación y Arbitraje.

Después de actualizado el objeto de tipo kardex se procede a eliminar, si hubieren, instancias de la clase hijo que tengan alguna relación con el personal a dar de baja. Por último se hace lo mismo con el objeto anónimo de empleado.

Fig. 3.10

Como puede observarse, la interpretación de los diagramas de interacciones es sencilla, además, la inclusión de texto a manera de *scripts* en la parte izquierda clarifica el significado del diagrama. Por esta razón omitiremos, salvo algunas excepciones, los detalles de interpretación de los diagramas de interacciones correspondientes a los escenarios faltantes.

Escenario 3. Cambio de Situación de Personal Federal (reanudación de labores)

Como puede notarse, este escenario es el inverso al escenario correspondiente a suspensión de efectos de nombramiento.

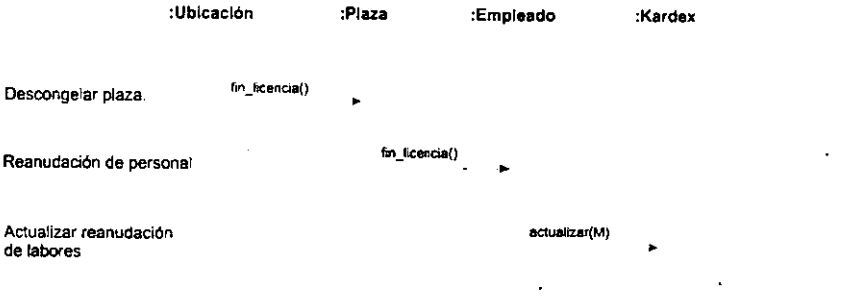


Fig. 3.11

Escenario 4. Movimientos de Personal. (nuevos ingresos)

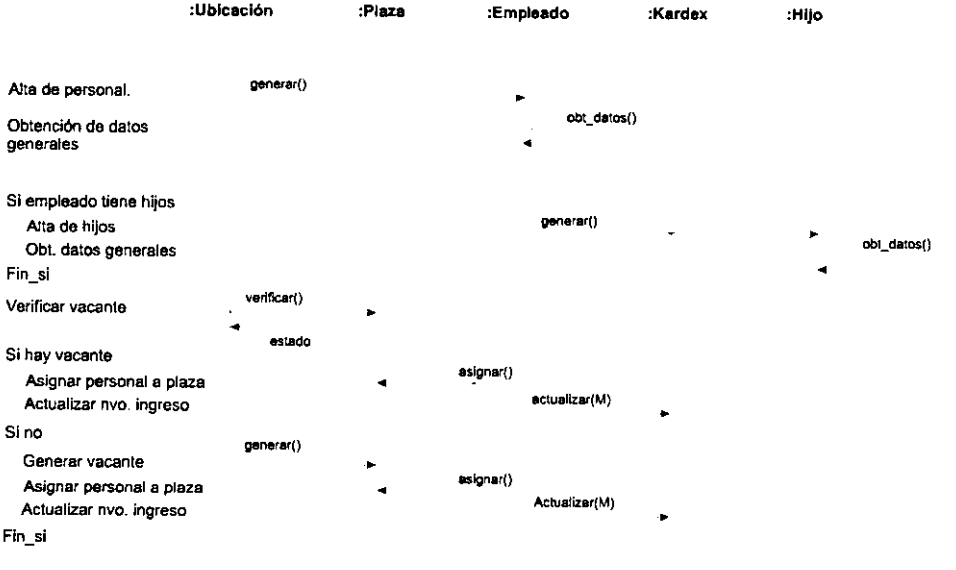


Fig. 3.12

Escenario 5. Movimientos de Personal (Reingreso)

Como puede notarse, el escenario anterior y éste (véase la figura 3.13), tienen casi el mismo comportamiento, salvo la invocación de la instancia de ubicación sobre la instancia anónima de empleado para que se actualice la fecha de reingreso para efectos de quinquenios, antigüedad, baja por de jubilación, etc.

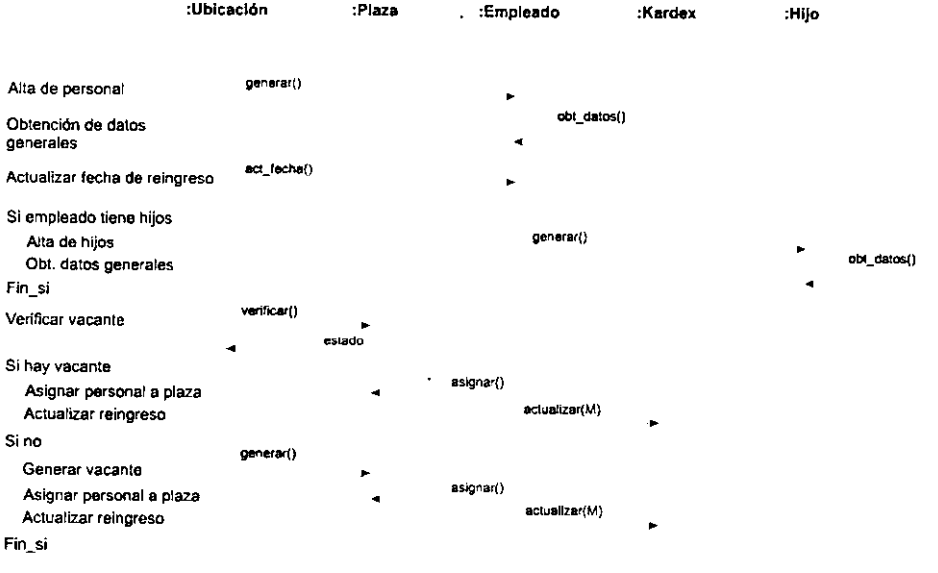


Fig. 3.13

Escenario 6. Movimientos de Personal (Cambio de Adscripción)

Un cambio de adscripción implica para el trabajador, renunciar a su plaza y obtener otra plaza no necesariamente del mismo nivel en otra área presupuestal.

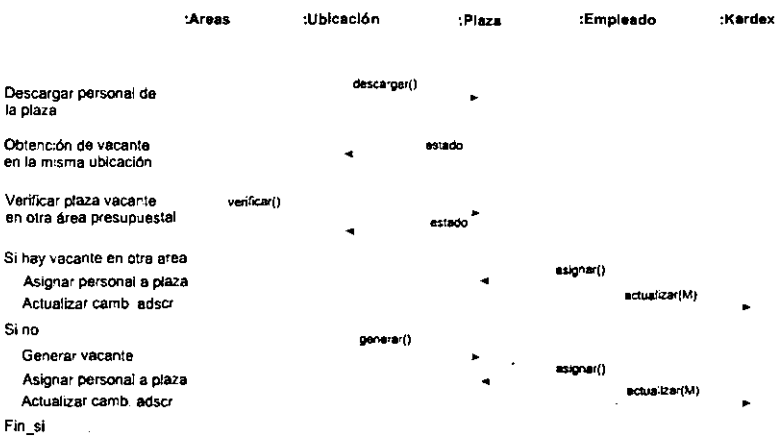


Fig. 3.14

Escenario 7. Movimientos de Personal (Promoción)

Una promoción es el cambio de asignación de plaza al personal. De esta forma, el personal deja vacante su antigua plaza para obtener una plaza de mejor nivel. Ambas plazas, vacante y receptora deben ser de la misma área presupuestal.

La mayoría de las promociones tienen como entrada una plaza vacante de mayor nivel y generan como salida otra plaza vacante de menor nivel. Debido a esta naturaleza de las promociones, observamos que estos movimientos generalmente no vienen solos.

En términos de los usuarios, decimos que son más comunes las cadenas de promociones, es decir, movimientos consecutivos de promociones que comienzan con la asignación de una plaza de buen nivel a un empleado, el cual deja vacante su plaza, esta plaza es asignada a otro empleado que a su vez deja vacante su plaza, y así sucesivamente.

Lamentablemente para el SICPRH, las cadenas de promociones no serán contempladas en el sistema, esto se debe a las políticas de operación que exigen la mayor discreción en cuanto a promociones. No obstante lo anterior, el escenario de promociones puede representarse en su nivel más simple a través del siguiente diagrama de interacciones.

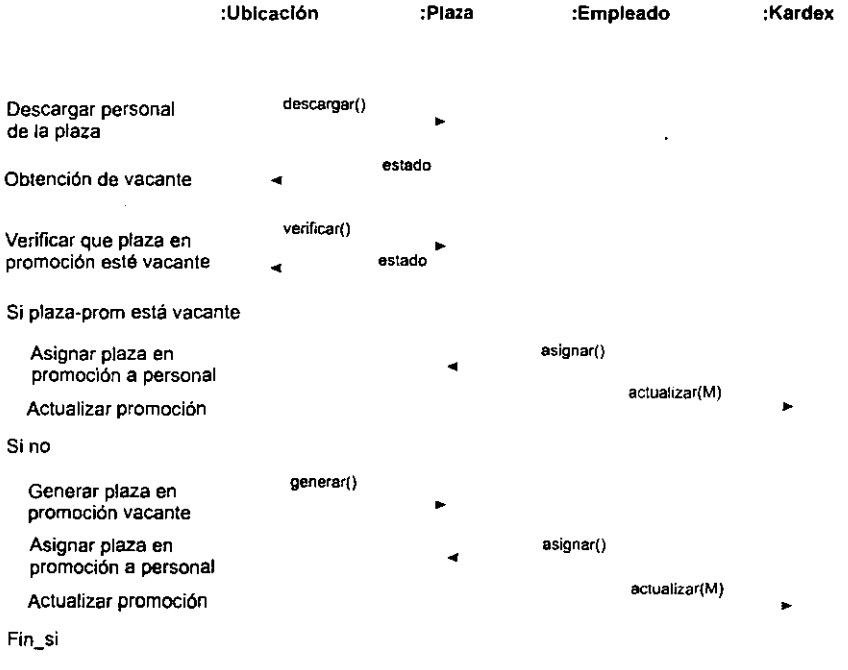


Fig. 3.15

Escenario 8. Movimientos de Personal (Reubicación)

Una reubicación es el movimiento del personal (con todo y plaza) a otra oficina dentro de la S.H.C.P.

En el siguiente diagrama puede observarse que hay varios tipos de reubicaciones:

1. Las reubicaciones internas se dan entre áreas presupuestales que pertenecen a la Subsecretaría del Ramo.

2. Si la reubicación es externa, el empleado puede ingresar a esta Subsecretaría, por tal motivo, se da de alta su plaza.

3. Si la reubicación es externa, el empleado puede salir de la Subsecretaría del Ramo, esto implica para esta entidad la pérdida presupuestal de la plaza y por consiguiente, la plaza se da de baja.

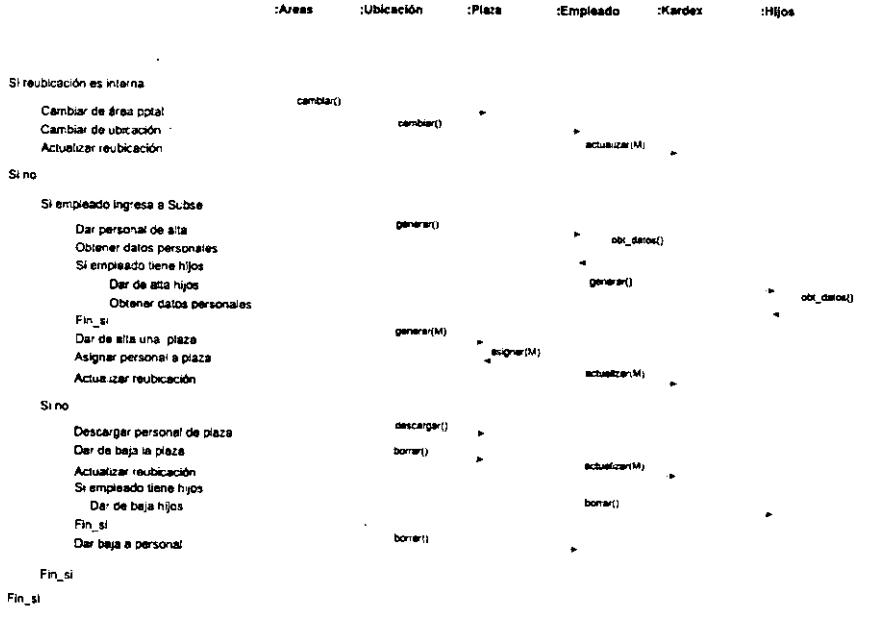


Fig. 3.16

Escenario 9. Conversión de plazas-puesto.

La conversión de plazas-puesto consiste en la fusión de una o más plazas de bajo nivel para obtener una plaza de mayor nivel, tal como se muestra en el diagrama 3.17.

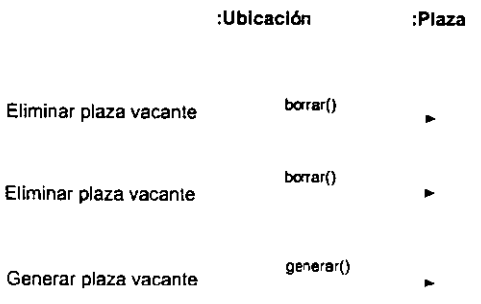


Fig. 3.17

Como se habrá notado, el diagrama de interacciones de arriba no es muy claro, una observación rápida solamente encuentra una instancia de la clase plaza que "parece" eliminarse dos veces para después volverse a crear. En estos casos utilizamos los diagramas de objetos.

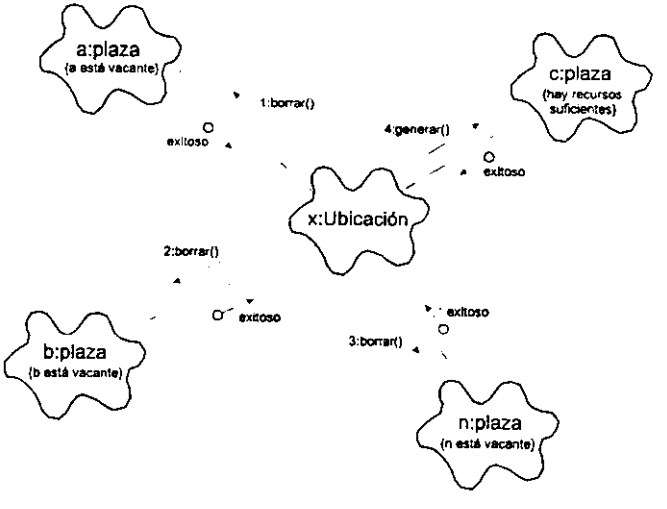


Fig. 3.18

Los diagramas de objetos se usan para mostrar la existencia de objetos y sus relaciones en el diseño lógico del sistema. En el AOO, indican la semántica de los escenarios primarios y secundarios que proporcionan indicios del comportamiento del sistema. Cada diagrama de objetos representa interacciones o relaciones estructurales que se podrían dar entre un conjunto de instancias de clases.

Un enlace entre dos objetos puede existir si y sólo si hay una asociación entre sus respectivas clases. El objeto que manda un mensaje conoce al objeto receptor, sin embargo, no necesariamente el receptor conoce al objeto que envió el mensaje.

Debe existir consistencia entre la estructura de clases y la estructura de objetos del sistema, de tal forma que si una operación (borrar(), por ejemplo) está siendo invocada al objeto a:plaza, la especificación de a:plaza (o de alguna de sus superclases) debe contener la declaración de la operación borrar().

Los enlaces entre objetos se denotan por una línea que une a ambos. La relación que sostienen los objetos consiste en:

a) Un símbolo de dirección de invocación (una línea con flecha) que especifica qué objeto invoca y qué objeto recibe el mensaje.

b) Una operación u evento escrito sobre la línea con flecha.

c) Opcionalmente, un número de secuencia a la izquierda de la operación invocada.

d) El flujo de datos entre objetos se especifica a través de un circulo (desde donde sale el dato) y una punta de flecha apuntando en la dirección a donde éste se dirige. El nombre del dato o variable se coloca debajo de este símbolo.

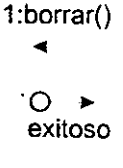


Fig. 3.19

Como en los diagramas de clase, en estos diagramas de objetos es útil reconocer los roles de algunos objetos, las condiciones necesarias para enviar un mensaje o realizar una acción y la identificación de campos llave.

La simbología de Booch contempla todos estos casos: los roles de los objetos se denotan como etiquetas o rótulos, las condiciones se encierran entre llaves "{}" y los campos llaves se identifican porque están encerrados entre corchetes cuadrados "[]".

Estamos ahora en posibilidad de poder interpretar el diagrama de objetos que se mostró previamente (Fig. 3.18); el escenario de conversión de plazas-puesto empieza cuando el objeto x (de la clase ubicación) envía una operación de borrar() al objeto a (de la clase plaza), sin embargo, para que éste pueda destruirse necesita cumplirse la condición de que este objeto se encuentre vacante, dependiendo si la condición sea cierta o no, el objeto a envía un valor en la variable exitoso que indica si el objeto se destruirá o si no lo hará. Si el objeto a se destruye, el objeto x hará la misma petición al objeto b y al objeto n.

La destrucción de los objetos del tipo plaza acrecentarán los recursos presupuestales para la creación de plazas de mejor nivel, de esta forma, el objeto x invocará la creación del objeto c. El éxito de este último movimiento será indicado por el valor de la variable exitoso.

Escenario 10. Registro de Incidencias.

El dominio del SICPRH sólo contempla incidencias para efectos de descuentos en bonos mensuales de productividad para los empleados. De esta manera, se registran ocurrencias tales como faltas, o uso de justificantes (días económicos, cuidados maternos y licencias médicas).

El uso de cuidados maternos y licencias médicas no representa descuento alguno si no se utilizan más de 9 justificantes (para cada tipo) al año. De cualquier forma, toda incidencia debe registrarse.

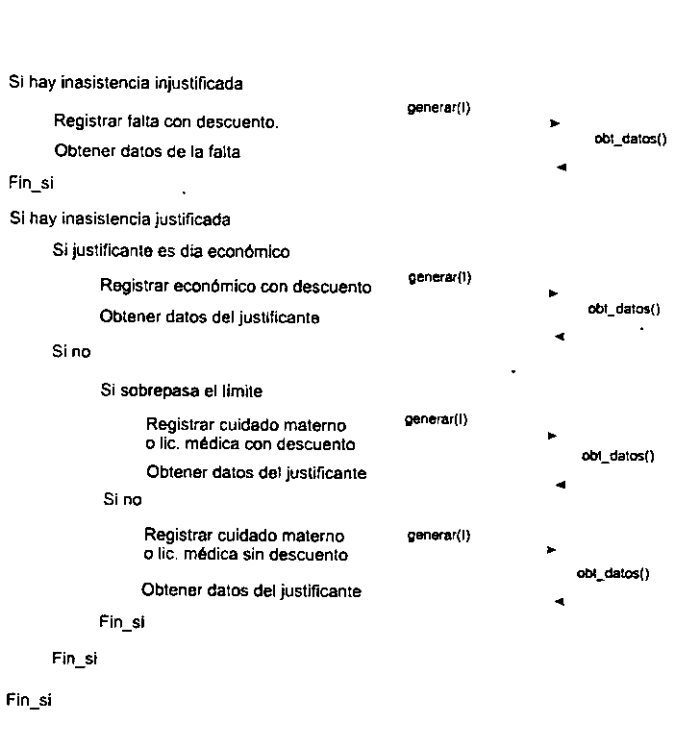


Fig. 3.20

Escenario 11. Capacitación y Desarrollo

Cada año se aplica un cuestionario de detección de capacitación a todo el personal de la Subsecretaría del Ramo. Las deficiencias son cubiertas por medio de cursos que imparte la propia Secretaría o alguna institución pública o privada.

Asimismo, los empleados pueden solicitar cursos, diplomados, talleres, etc. en cualquier época del año, la inscripción del personal a dichos cursos solicitados está en función de la existencia de recursos presupuestales. Algunas veces la Dirección General de Recursos Humanos invita a cursos extemporáneos a personal que cubra cierto perfil.

Cualquier tipo de asistencia a cursos, talleres o diplomados debe registrarse.

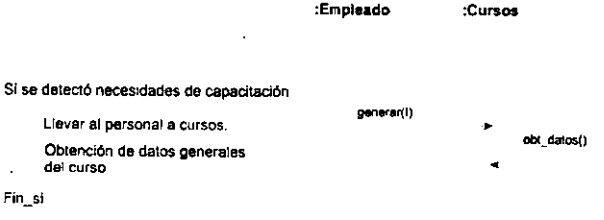


Fig. 3.21

III.4.2.2. Escenarios secundarios

Se ha mencionado que los escenarios secundarios señalan el comportamiento del sistema bajo condiciones anormales de operación. De esta forma, en este AOO del SICPRH hemos identificado los sigs. escenarios:

1. Generación de objetos antes del arranque del sistema.
2. Creación, modificación y destrucción de áreas presupuestales.
3. Creación, modificación y destrucción de ubicaciones.
4. Actualización de los datos de empleados federales.

Describiremos ahora la semántica de estos escenarios.

Escenario 1. Generación de objetos antes del arranque del sistema.

Este escenario secundario es típico en todos los sistemas OO. El SICPRH necesita, antes de la puesta en operación, contener las áreas presupuestales de la Subsecretaría del ramo, las ubicaciones para cada área presupuestal y todos los puestos (no las plazas) necesarios.

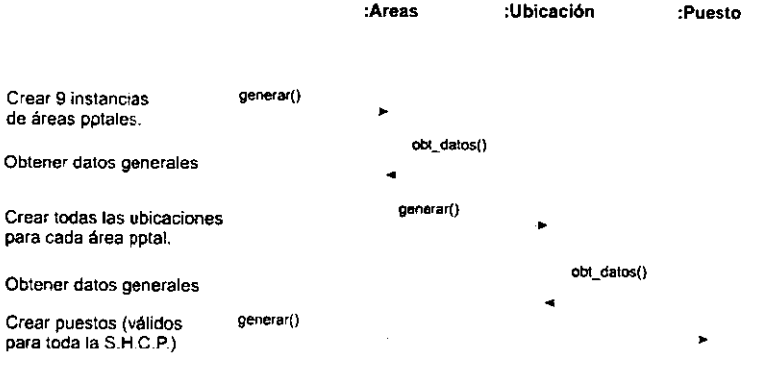


Fig. 3.22

Escenario 2. Creación, modificación y destrucción de áreas presupuestales.

Los movimientos en cuanto a áreas presupuestales no son normales, sin embargo llegan a presentarse de acuerdo a las reestructuraciones en el gobierno federal o por mandato presidencial. Por ejemplo, a mediados de 1997 no existía el área 310 Dirección General de Política de Ingreso, posteriormente se le cambió su clave a 217.

Los movimientos de áreas pueden ser diversos, en la actualidad existe el rumor de que el área 210 Unidad de Planeación del Desarrollo se va a fusionar con la 214 Dirección General de Banca Múltiple. Aunque la fusión de áreas parece no estar contemplado, en el AOO se ha descubierto que, todos los movimientos habidos y por haber se pueden realizar a través de los movimientos primitivos de creación, modificación y eliminación de áreas.

Estos movimientos primitivos están contemplados en el diagrama de la fig. 3.23.

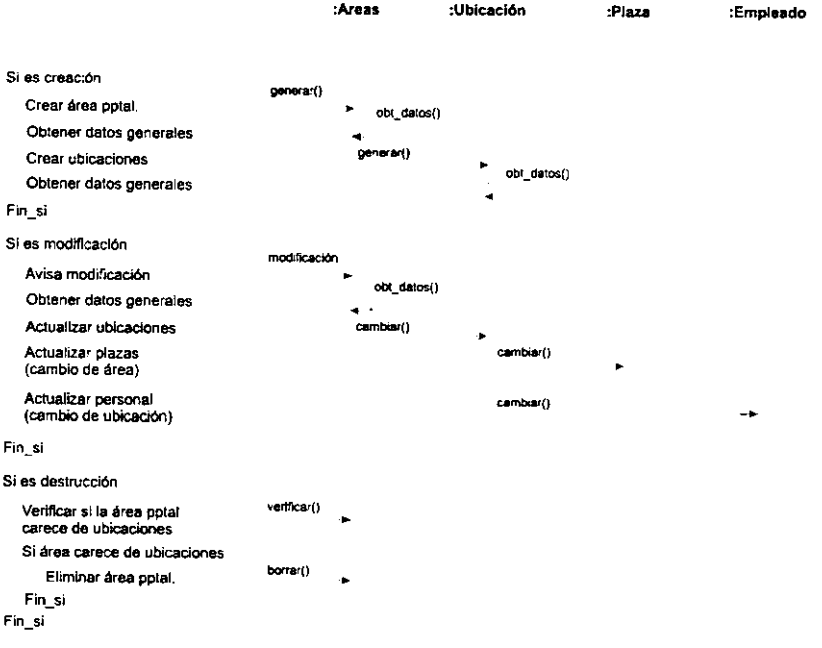


Fig. 3.23

Nótese que ciertas operaciones son invocadas desde afuera del sistema (la operación generar() instancía un objeto de la clase áreas), además el mensaje modificación también es enviado desde el medio ambiente del SICPRH hasta la instancia de áreas.

Escenario 3. Creación, modificación y destrucción de ubicaciones.

Los movimientos de ubicaciones son más comunes que los movimientos de áreas porque no requieren la aprobación del Ejecutivo Federal. Normalmente ocurren cuando hay cambios de mandos medios (directores o subdirectores de área) o cuando la necesidad así lo exige.

Estos tipos de movimientos contemplan la creación y/o destrucción de subdirecciones, departamentos y oficinas, así como la modificación en la clave o la descripción de las mismas. Todos estos movimientos, con excepción de la eliminación, deben actualizarse sobre el personal que trabaja en estas ubicaciones.

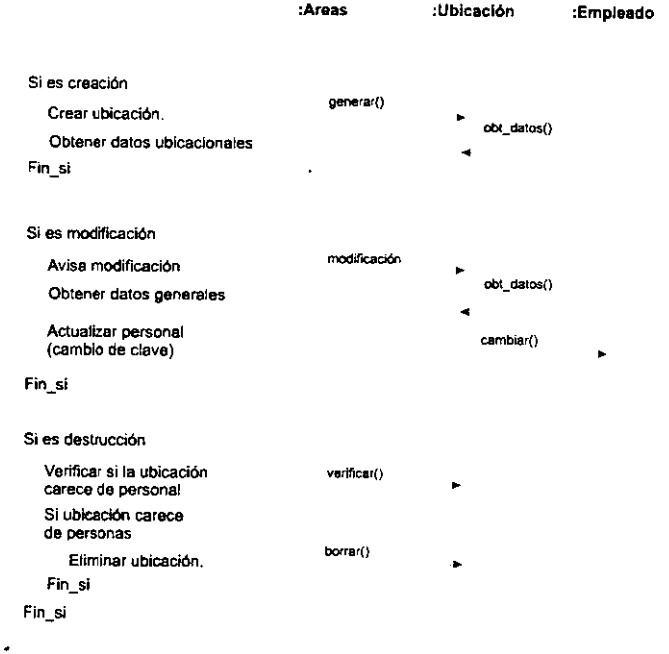


Fig. 3.24

Escenario 4. Actualización de los datos de empleados federales.

En este escenario estamos manejando datos personales de los empleados, tales como nivel de escolaridad, posesión de idiomas extranjeros o registro de hijos. Aunque pudiera parecer que todo lo anterior es un escenario primario, no es así; son muy pocos los movimientos debidos a actualización de datos personales.

Esto se debe a que los empleados federales carecen de interés en incrementar su nivel escolar o en aprender una lengua extranjera. La mayor parte de estos movimientos se limitan a registros de nacimiento de hijos para obtener licencias de tiempo por lactancia, regalos el día del niño o el día de las madres. Además, las licencias de tiempo no repercuten en el control de plazas-puesto y de recursos humanos.

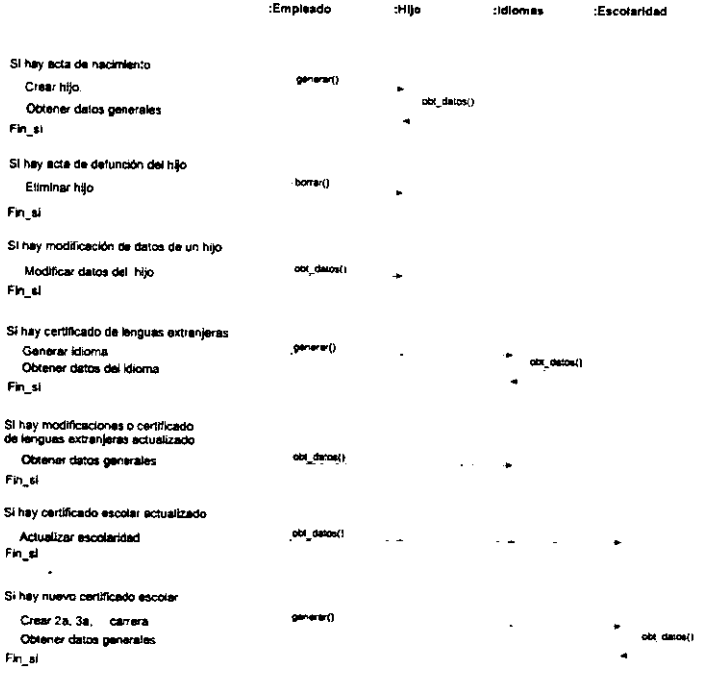


Fig. 3.25

Algunas otras características del sistema parecen ser parte de un comportamiento de operación "extraño" por parte del SICPRH y se podría pensar incluirlos como escenarios secundarios. Podemos decir que los aumentos de salario anuales o extraordinarios, y la modificación de descuentos o percepciones son ejemplos de ello. Sin embargo, estos hechos no son escenarios sino responsabilidades de los objetos (de la clase puesto en este caso) y por este motivo no son considerados en esta etapa.

III.4.3 Vista dinámica del SICPRH

La última tarea de la planeación de escenarios consiste en el desarrollo de máquinas de estados finitos o diagramas de estados para las abstracciones que tengan un ciclo de vida importante.

Aunque los diagramas de interacciones y los diagramas de objetos permiten conocer en gran medida la semántica de los escenarios y por ende, el comportamiento del sistema, nuestro análisis no estaría completo si no identificamos los estados de las abstracciones clave.

El **estado de un objeto** es la suma acumulada de su comportamiento, es decir, de sus cambios de estados pasados. En un determinado momento, el estado de un objeto está dado por todas sus propiedades (usualmente estáticas) y por los valores actuales de dichas propiedades (usualmente dinámicos).

Como ejemplo, pensemos en el objeto clima de la Cd. de México, el estado actual del clima está formado por las propiedades humedad, temperatura, viento, etc. y por los valores que éstos tienen, de tal forma que la conjunción de éstos pueden provocar que el estado de ese objeto sea lluvia. Además el estado actual de lluvia podría ocasionar, en el futuro, otro estado (el estado niebla, por ejemplo).

Un **evento** es cualquier ocurrencia que pueda causar que el estado de un sistema cambie, provocando una acción o no. "Una **acción** es una operación, que, para efectos prácticos, no lleva tiempo realizarla"[9]. Una acción denota la invocación de un método, la activación de otro evento o el comienzo o fin de otra actividad. Esta actividad es alguna operación que necesite algún tiempo para su realización.

Los diagramas de transición de estados son usados para mostrar los estados de una determinada clase, los eventos que causan las transiciones de un estado a otro, y las acciones resultantes de esos cambios.

La notación de Booch para los diagramas de estados tiene sus antecedentes en la notación utilizada por Harel[10]. En ésta los estados están representados por rectángulos con bordes redondos. Estos estados se identifican por un nombre ubicado en la parte superior del rectángulo. Una línea opcional separa el nombre del estado de las acciones que se realizan en este estado.

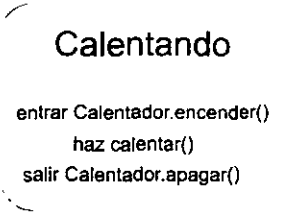


Fig. 3.26

[9] Booch, Grady. op.cit. p.p 201
[10] Op. Cit. p.p. 199

El icono que se muestra arriba corresponde al estado calentando del objeto termostato (éste es un agregado que contiene a un objeto calentador). Nótese que al ingresar a ese estado se invoca la operación calentador.encender(), mientras se mantiene el estado calentando de igual forma se mantiene la operación calentar(), al salir de este estado se ejecuta la operación calentador.apagar().

Una transición de estados es un cambio de un estado a otro, o a sí mismo, causada por la aparición de un evento. Esta situación se representa por una línea que une al estado que termina y el estado que comienza, identificando a este último por una punta de flecha. Sobre esta línea de transición de estados se coloca una etiqueta que nombra el evento y opcionalmente la acción que produce esa transición.

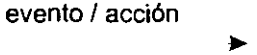


Fig. 3.27

Por último, en los diagramas de transición de estados existe un estado inicial por defecto, (representado por un círculo negro) y sólo en algunos es necesario indicar un estado final (representado por dos círculos concéntricos)

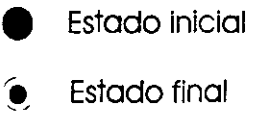


Fig. 3.28

Ahora estamos en posibilidad de estudiar el ciclo de vida de las dos abstracciones clave del SICPRH; las clases plaza y empleado. El reconocimiento de todos los estados para cada abstracción es importante porque añade a nuestra planeación de escenarios una nueva vista, una vista que no es observable en el estudio de cada escenario porque en éstos es notoria la interrelación entre objetos, pero no es observable el cambio que sufre un objeto a través de toda su vida.

Es hora de que examinemos cuidadosamente y de forma aislada el comportamiento interno de las abstracciones clave. Empecemos con la clase plaza.

Fig. 3.29

En el diagrama de la página anterior puede observarse que una plaza puede crearse por ingreso a la Subsecretaría del Ramo vía creación por reubicación externa, lo cual manda a la plaza al estado de En trámite de reubicación, si se presenta el evento reubicación aceptada, la plaza se convierte en una plaza normal o Activa. Por otra parte, si existen recursos, la Subsecretaría puede crear nuevas plazas, lo cual genera una plaza vacante.

La plaza vacante puede sufrir una asignación a personal y convertirse en una plaza activa o puede sufrir una baja lo cual invoca la acción de plaza.borrar()

No es posible eliminar una plaza activa (puesto que implicaría dar de baja al personal que ostenta esa plaza), para realizar esto es necesario un descargue de personal, lo cual convierte a la plaza en vacante. Por otra parte, una plaza activa puede convertirse en plaza en licencia si se presenta una suspensión de efectos de nombramiento y, de manera recíproca, una plaza en licencia puede regresar a activa si surge una reanudación de labores. Una plaza en licencia puede sufrir una baja e invocar a la operación plaza.borrar().

El diagrama de abajo corresponde al comportamiento de un objeto de la clase empleado.

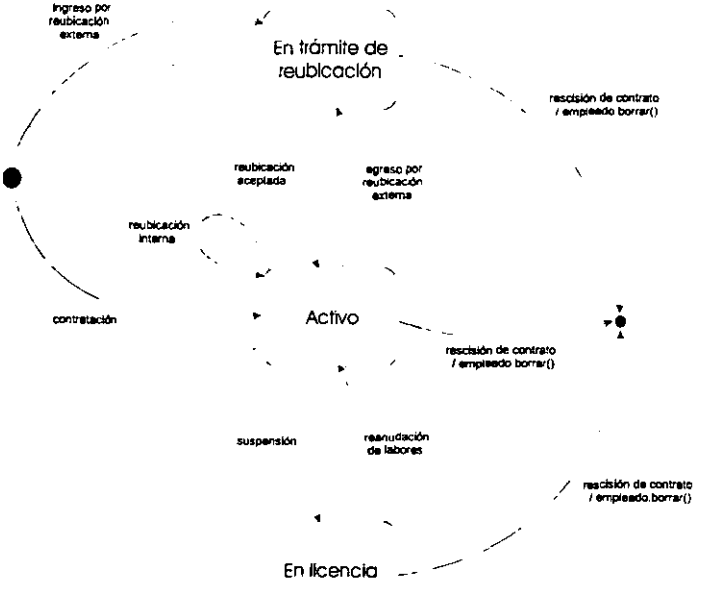


Fig. 3.30

Del diagrama podemos deducir que cualquier empleado sólo puede tener tres estados, en trámite de reubicación, activo y en licencia. Este empleado puede crearse si hay un ingreso por reubicación externa y de esta forma llegar al estado de en trámite de reubicación, si su ingreso es aceptado se convierte en un empleado en activo.

Un empleado en activo pudo haber sido creado por contratación. Este empleado puede sufrir una reubicación interna y no cambiar de estado, puede padecer una suspensión de labores, lo cual la convierte en un empleado en licencia (sin sueldo, con medio sueldo, con goce de sueldo íntegro), pero puede regresar a activo cuándo se presente una reanudación de labores.

Cualquier empleado puede esperar el evento de rescisión de contrato, no importando el estado en el que se encuentre, si esto ocurre, se activa la operación empleado.borrar().

Capítulo IV

Diseño Orientado a Objetos del SICPRH

IV. 1. Macroproceso

En las metodologías estructuradas para la generación de sistemas los modelos conceptuales utilizados para el análisis, el diseño y la implementación difieren entre sí. "Los analistas utilizan los modelos de relación entre entidades, la descomposición funcional y las matrices. Los diseñadores utilizan diagramas de flujo de datos, tablas de estructura y diagramas de acción. Los programadores usan las construcciones COBOL, FORTRAN, C, ó ADA"[1]

En cambio, en las metodologías orientadas o objetos todos utilizan el mismo modelo conceptual: analistas, diseñadores, programadores y hasta los usuarios finales, "todos piensan en los tipos de objetos (clases), los objetos y su comportamiento. Establecen jerarquías de clases donde los subtipos comparten las propiedades de su padre. Piensan en términos de objetos compuestos de otros objetos y su generalización y encapsulado. Piensan en eventos que cambian los estados de los objetos y activan ciertas funciones u operaciones"[2].

[1] Martin, James. op.cit.,.p.p.78
[2] Idem.

Lo anterior hace posible una transición entre análisis y diseño de manera natural, aunque en ocasiones sea confuso diferenciar ambas etapas.

Así entonces, se puede decir que los límites entre análisis y diseño son muy difusos, y en ocasiones imperceptibles. No obstante, los enfoques son diferentes; en el análisis buscamos modelar el sistema a través del descubrimiento de las clases y objetos que conforman el dominio del problema. **En diseño *inventamos* las abstracciones y mecanismos que proveen el comportamiento que el modelo requiere.**

Si las fronteras entre AOO y el DOO son oscuras, es igualmente complicado saber cuándo terminamos el análisis para iniciar con la etapa de diseño. Aquí entra en juego la experiencia del equipo de desarrollo del proyecto; generalmente el proceso de diseño comienza tan pronto como se tenga un modelo de la estructura y el comportamiento del sistema *razonablemente completo.*

Sobre este punto es necesario hacer una reflexión; no debemos empezar a diseñar si el AOO no ha llegado a su madurez, esto es, si todavía no se han identificado y modelado las abstracciones y escenarios primarios. Hacerlo significa construir un diseño prematuro que pondrá en riesgo los tiempos de trabajo, y lo que es peor, la arquitectura del sistema. Por otra parte, es igualmente importante impedir que esta fase de diseño se relegue hasta que el análisis haya producido un modelo completo (léase perfecto), a este error tan común se le ha denominado la "parálisis del análisis"[3].

El Diseño Orientado a Objeto (DOO) tiene un doble propósito, el primero consiste en crear una arquitectura del sistema para la fase de producción. El segundo es el establecimiento de políticas para aspectos comunes del sistema (*common tactical policies*).

[3] Booch, Grady. Op. cit. pp. 255

Hablemos del primer objetivo, la **elaboración de una arquitectura**. A la estructura interna (lógica y física) de un sistema se le denomina su **arquitectura**. La arquitectura de un sistema OO abarca la estructura de clases y objetos del dominio del problema, organizado en términos de capas o niveles de abstracción.

El DOO toma el modelo del sistema producido en el AOO y comienza a elaborar una arquitectura a través de éste. Así entonces podemos decir que la construcción de la arquitectura empieza desde el análisis, pero es en el diseño donde se formaliza.

Para la elaboración de la arquitectura del sistema debemos distinguir las **decisiones de diseño estratégicas y tácticas**. Las primeras afectan las estructuras arquitectónicas más altas (mecanismos para el manejo y detección de errores, interfaces de usuario, manejo de memoria, por ejemplo). Las decisiones tácticas tienen implicaciones arquitectónicas a un nivel local (el protocolo de una clase, la descripción de un método, por ejemplo).

Otro factor a considerar es el manejo de **mecanismos**. Mientras que la identificación de las abstracciones clave juegan un papel esencial en el análisis, los mecanismos son el alma del diseño; el equipo de desarrollo debe considerar no sólo el diseño de las clases en forma individual, sino cómo las instancias de esas clases trabajan juntas. Un mecanismo significa una decisión de diseño acerca de cómo los objetos cooperan entre sí. De esta forma, se puede decir que los mecanismos representan patrones de comportamiento.

La elección de mecanismos a usar para cada escenario a menudo es el resultado de factores como costos, confiabilidad, seguridad, portabilidad, etc. Sin embargo, una vez que los desarrolladores se han decidido sobre un particular patrón de comportamiento, el trabajo se distribuye sobre los objetos que conforman el escenario por medio de la definición de métodos para cada instancia.

La descripción de la arquitectura de un sistema se expresa por medio de diagramas de clases y objetos. A pesar de lo anterior, Booch recomienda tener versiones sucesivas de la arquitectura[4]. Estas versiones representan una parte de la arquitectura total del sistema que sólo considera ciertas semánticas importantes, aunque incompletas.

La razón de una versión parcial de la arquitectura se justifica si se piensa que ésta debe ser ejecutable, es decir, debe permitirse la implementación de la estructura del sistema para poder ser estudiada y evaluada. La realización de una nueva versión arquitectónica será más completa y robusta que la anterior, de forma que siguiendo las iteraciones obtendremos una arquitectura total.

El segundo producto del diseño, el **establecimiento de políticas para aspectos comunes del sistema**, no es menos importante que la creación de una arquitectura. De hecho, un pobre diseño táctico de políticas puede arruinar la arquitectura más robusta.

El establecimiento de políticas comunes implica la identificación de los mecanismos que se pueden presentar a lo largo de todo el sistema, además del diseño de los artefactos que permitan manejarlos (políticas para el manejo y detección de errores, manejo de memoria, manejo de almacenamiento en disco, etc.). Estas políticas se establecen cuando ya han sido definidas las decisiones estratégicas de diseño.

Es importante diseñar explícitamente estas políticas, pues de otra forma podríamos ver a cada programador inventar sus propias soluciones para aspectos comunes del sistema, arruinando de esta forma la arquitectura.

Las actividades propias para esta fase del macroproceso son tres:

[4] Recuérdese que para Booch el micro y macroproceso de desarrollo de un sistema OO es una actividad iterativa e incremental.

PLANEACIÓN DE LA ARQUITECTURA.

Esta tarea implica inventar las capas y particiones del sistema en su conjunto. Una **capa** es una colección de categorías de clases o subsistemas en un mismo nivel de abstracción. Una **partición** es una "rebanada" de diferentes capas, de tal forma que contenga subsistemas de varios niveles de abstracción que puedan constituirse en una versión arquitectónica.

La planeación arquitectónica involucra tanto una descomposición lógica (grupos de clases) como física (grupos de módulos, funciones y procesadores). Las siguientes son tareas típicas de esta actividad:

- Agrupar los puntos funcionales generados en el análisis y ubicarlos en capas o niveles de abstracción, de tal forma que las funciones que se construyan sobre otras funciones caerán en diferentes niveles, no así las funciones que colaboren entre sí, éstas conformarán un mismo nivel de abstracción.
- Con esa agrupación de puntos funcionales, crear una arquitectura "parcial" que contemple sólo algunos escenarios.
- Estimar las fortalezas y debilidades de la arquitectura así generada.

DISEÑO DE POLÍTICAS.

Se ha hablado sobre la importancia de la implementación de políticas para aspectos comunes a todo el sistema. La realización de este diseño puede basarse en los siguientes eventos:

- Identificar los mecanismos y/o aspectos **dependientes de la aplicación** y que son comunes a todo el sistema, tales como manejo de memoria, de errores, etc. Diseñar una política que sea congruente con las decisiones estratégicas del diseño de la arquitectura.
- De igual forma, identificar los aspectos **dependientes del dominio de la aplicación** y que son comunes a todo el sistema (idioma, respuesta en tiempo real, etc.). Diseñar una política para cada aspecto.

- Para cada política desarrollar un escenario que describa su semántica.
- Documentar cada política.

PLANEACIÓN DE VERSIONES ARQUITECTÓNICAS.

La generación de versiones arquitectónicas sucesivas requiere una planeación, tal actividad puede involucrar los eventos sigs. :

- Organizar los escenarios productos del análisis y ordenarlos de acuerdo a su impacto en el sistema total (hay escenarios fundamentales y otros que son "periféricos").
- Agrupar los escenarios o puntos funcionales en versiones arquitectónicas de tal forma que la última versión los contemple a todos.
- Integrar los escenarios fundamentales en una arquitectura y trabajar en un prototipo evolutivo (no es un prototipo de desecho).
- Ajustar los objetivos y calendarios de estas versiones de tal forma que exista suficiente tiempo para la elaboración de la siguiente versión arquitectónica.

Es deseable el empleo de versiones arquitectónicas sucesivas porque genera un producto intermedio que puede ser evaluado y validado en una revisión formal. Una buena arquitectura es aquella que oculta la complejidad del sistema de manera que parezca simple.

La validación de la arquitectura y el diseño de las políticas comunes del sistema nos permiten comenzar la siguiente fase del macroproceso, esto es, la Fase de Implementación (producción) del software.

IV.2 El Microproceso en el DOO

Como se vio en el capítulo I, las etapas del microproceso de desarrollo son las mismas para cualquier fase del macroproceso. Algunas veces la diferencia consiste en la profundidad o el grado de refinamiento que se requiere en una etapa específica del microproceso (identificación de clases y objetos en el DOO), con la que se obtiene en la misma etapa de otra fase del macroproceso (identificación de clases y objetos en el AOO, por ejemplo). En otras ocasiones es la fase del macroproceso la que controla los objetivos y/o altera de alguna manera la naturaleza del microproceso, de tal forma que una misma etapa de este último pueda ser diferente para cada fase de aquel.

Lo anterior no quiere decir que las etapas del microproceso son diferentes según sea la fase del macroproceso que se esté desarrollando. Todo lo contrario; existen tantas similitudes como diferencias en las etapas del microproceso a lo largo de todas las fases del macroproceso de desarrollo.

IV.2.1 Identificación de clases y objetos

Una vez descubiertas las abstracciones del dominio del problema, podremos notar que ciertos comportamientos de la aplicación no son realizados por las clases y objetos identificados en el AOO. Además, la generación de un nuevo sistema a menudo incluye características que no tienen semejanza en el mundo real. Por lo tanto ha llegado el momento de diseñar.

El sentido de esta microactividad, como parte del DOO, consiste en inventar nuevas abstracciones que formen parte de la solución al problema. Tal vez algunas de las clases y objetos que se identificaron antes podrían estar equivocados, pero esto no es necesariamente malo. Conforme se va aprendiendo y asimilando el problema, lo más probable es que vayan apareciendo cambios en ciertas abstracciones, ya sea por reubicación

de responsabilidades, por combinación de abstracciones similares o, lo más común, dividiendo abstracciones grandes en grupos de abstracciones pequeñas que forman parte de algún mecanismo de nuestra solución.

La integración de estas abstracciones en el diccionario de datos permitirá a los diseñadores tener una vista global del proyecto.

IV.2.2. Identificación de la semántica de clases y objetos

El propósito de esta etapa del microproceso es el de establecer el comportamiento y los atributos de cada abstracción identificada en la etapa previa. Esta tarea involucra tres actividades, la **elaboración de libretos de comportamiento para cada escenario** (se explicó para esta microactividad en el AOO), **el diseño de clases aisladas** y **la detección de patrones comunes de comportamiento**.

El **diseño de clases aisladas** permite refinar las abstracciones del sistema, al mismo tiempo que se consideran ciertas decisiones tácticas de diseño. Esta tarea incluye los eventos siguientes:

- Seleccionar una abstracción y enumerar sus roles y responsabilidades.
- Elaborar un conjunto de operaciones que puedan satisfacer esas responsabilidades. Hay que tomar en cuenta el reuso de operaciones.
- Considerar cada operación y asegurarse que es una operación primitiva (que no puede obtenerse por medio de otras operaciones). En caso de que no lo sea, aislar la operación y descomponerla en sus operaciones primitivas para que éstas puedan definirse en una superclase o categoría de clases.

- Considerar la completez de las operaciones de cada abstracción; las operaciones que no son exigidas ahora pueden serlo en un futuro inmediato.

La tercera actividad es la **detección de patrones comunes de comportamiento** (*pattern scavenging*), la cual tiene como objetivo encontrar elementos comunes que representen oportunidades de reuso. Las siguientes heurísticas van encaminadas a lo anterior.

- Buscar patrones de interacción entre las abstracciones. Tales colaboraciones deben ser examinadas para asegurarse de que la existencia de operaciones semejantes es justificada
- Buscar patrones de comportamiento (comportamientos semejantes) en el conjunto de abstracciones que se tiene. Roles y responsabilidades comunes deberán ser unificados en forma de una clase base, superclase o utilería de clases.

Esta microactividad concluye con la **elaboración de un conjunto de operaciones suficientes y completas para cada abstracción**, así como la **especificación total del protocolo de cada clase**.

IV.2.3. Identificación de las relaciones entre clases y objetos

Esta microactividad formaliza la separación entre los elementos conceptuales y físicos de nuestras abstracciones; aquí se definen los grupos de clases de mayor nivel que se convertirán en categorías de clases y los grupos de módulos que se convierten en subsistemas.

En el DOO aplicamos estas actividades para especificar las colaboraciones que conforman cada mecanismo de nuestra arquitectura. Diagramas de clases, objetos y módulos son considerados como productos de esta actividad puesto que estos diagramas ofrecen una vista amplia de la arquitectura del sistema.

Los diagramas que se elaboraron en el AOO para esta etapa del microproceso son refinados en el diseño, de tal forma que puedan expresar decisiones tácticas como relaciones de herencia, agregación, instanciación y/o uso, además de ciertas propiedades de las relaciones como cardinalidad, roles, etc.

Esta etapa consta de tres actividades:

La **especificación de asociaciones** es básicamente una actividad del AOO, consiste de inspeccionar cada escenario clave para encontrar dependencias entre abstracciones, las dependencias directas implican relaciones que posteriormente podrán ser catalogadas como herencia, agregación o uso; las dependencias indirectas podrían generar nuevas abstracciones que sirvan como agentes o intermediarios.

La **identificación de colaboraciones** es una tarea de diseño que permite descubrir relaciones entre abstracciones y proveer comportamientos a escenarios. Para reconocer tales colaboraciones podemos seguir las sigs. tareas:

- Producir un diagrama de objetos para cada mecanismo, ilustrando su semántica dinámica a través de diagramas de interacciones que muestren la manera de cómo los objetos proporcionan el comportamiento al mecanismo. Conforme se desarrolla esta actividad se puede descubrir la necesidad de introducir nuevas relaciones entre los objetos así como eliminar o consolidar las relaciones que ya existen.
- Toda vez que se tienen los mecanismos bien definidos, agrupar las clases en jerarquías de tal forma que se puedan identificar las capas o niveles de abstracción de la arquitectura del sistema.

- Ubicar las clases y objetos en módulos de acuerdo a la visibilidad que sus relaciones requieran.
- En la organización de los módulos es deseable que éstos sean altamente cohesivos y pobremente acoplados.

Por último, el **refinamiento de las asociaciones** entre abstracciones busca convertir las relaciones identificadas en el AOO en otras más precisas, obteniendo así una estructura lista para la implementación. Los eventos que se pueden desarrollar para este fin son los sigs.:

- Buscar patrones de comportamiento que expresen oportunidades de especialización/generalización (herencia) en las asociaciones entre clases. Ubicar estas clases en una jerarquía existente o inventar una si es posible.
- Si existen patrones en la estructura de las clases, considerar la creación de clases que capturen esta estructura común e introducirla en el escenario, en forma de herencia, agregación, etc.

Se ha concluido esta microactividad cuando se tienen bien especificadas la semántica y las relaciones entre las abstracciones, de tal forma que sea suficiente, estructuralmente, para la construcción de la arquitectura del sistema.

IV.2.4. Implementación de clases y objetos

El propósito de esta microactividad durante el DOO es crear representaciones tangibles de nuestra arquitectura, de preferencia deben ser ejecutables para poder obtener una versión sucesiva y parcial de la arquitectura que pueda evaluarse.

La tarea principal en esta etapa consiste en la selección de estructuras y algoritmos que representen la semántica de las abstracciones identificadas antes. Las heurísticas para esta actividad son:

- Se toman en cuenta los métodos y funciones propios del lenguaje de implementación para ubicar cuáles pueden ser utilizados en las operaciones de las abstracciones.
- Se selecciona el algoritmo adecuado para cada operación. De la misma forma, se introducen operaciones intermedias para dividir algoritmos complejos y convertirlos en algoritmos menos complicados y reusables.

Esta microactividad se considera finalizada cuando obtenemos un modelo cerca de ser ejecutable.

IV.3. DOO del SICPRH

Se ha mencionado que la transición entre el análisis y el diseño orientado a objetos es un proceso natural, lo mismo sucede con el paso del diseño hacia la implementación del software.

Las actividades del DOO buscan terminar de construir la arquitectura del sistema a través de la invención de nuevas abstracciones y mecanismos, así como el refinamiento de las relaciones que existen entre los objetos. Además, se requiere un nivel de detalle preciso, quiero decir que no es suficiente identificar el protocolo de los objetos, se debe llegar hasta las operaciones y los métodos (y definir estos últimos mediante algoritmos), de tal forma que se pueda iniciar la fase de producción.

Como se mencionó en el primer capítulo, en el SICPRH tenemos una decisión estratégica de sistema de gran importancia: la implementación hará uso de un SMBDR (sistema manejador de bases de datos relacionales) y de su lenguaje de programación. Por esta razón necesitamos un diseño que traslade nuestra arquitectura OO hacia una arquitectura tradicional, de tal manera que nuestra transición a la fase de implementación sea lo menos abrupta posible.

Así entonces, ¿de qué manera transformaremos nuestra arquitectura OO en una arquitectura de base de datos relacional? Existen varias maneras, dependientes del tipo de lenguaje de programación y del tipo de base de datos, a continuación describiremos algunas de ellas.

El uso de un **SMBDR en conjunción con un lenguaje de programación orientado a objetos** combina las ventajas del modelo a objetos con las de la tecnología de las bases de datos relacionales. Además, no requiere cambios significativos en la arquitectura del sistema y, mucho menos, en las actividades del DOO. Una posible solución para la integración de ambas tecnologías consiste en generar un mecanismo de persistencia de los objetos en una base de datos, este mecanismo sería soportado por una superclase (llamémosle ObjectToStore), la cual tendría las siguientes responsabilidades:

- Conocer la información (atributos del objeto) que serán almacenados.
- Distinguir el estado de persistencia del objeto (tal estado puede ser un valor lógico que indique falso si el objeto aún no es persistente o verdadero si ya lo es).
- Administrar el proceso de convertir al objeto de no persistente a persistente por medio de una petición a otro objeto.

Este objeto que recibe la petición sería una instancia de una clase adicional (podemos llamarle RDBPort) que fuera responsable de la interface con el SMBDR a través de comandos de SQL por ejemplo. De esta manera encapsularíamos la interface de nuestra arquitectura OO con el SMBDR, pues sólo esta clase conocería la manera en cómo trabajar con la base de datos relacional.

Esta solución implica la creación de las clases ObjectToStore y RDBPort, además del diseño e implementación de la tabla en donde se almacenarían los datos. Así, cada objeto que necesite persistencia debe ser instancia de alguna subclase de ObjectToStore y ninguna otra modificación importante sobre el DOO sería necesaria.[5]

El procedimiento anterior es una forma muy elegante para ligar un SMBDR y un DOO, sin embargo, las cosas no son tan fáciles cuando **no se hace uso de bases de datos relacionales y cuando el lenguaje de programación no soporta la orientación a objetos**.

En teoría, la creación de objetos y la construcción de software orientado a objetos se puede llevar a cabo utilizando cualquier lenguaje de programación convencional (por ejemplo, C o PASCAL). Para Martin y Odell[6] una implementación OO puede resumirse de la siguiente manera:

Implementación OO	=	Tipos de datos abstractos.	+	Herencia	+	Selección del método

El manejo de **tipos de datos abstractos** no es exclusivo de los lenguajes de programación orientados a objetos (LPOO), aún más, casi todos los lenguajes manejan los tipos de datos definidos por el usuario. Martin y Odell aconsejan definir procedimientos o subrutinas que simulen las operaciones comunes a todas las clases, por ejemplo, la invocación de constructores y destructores de objetos.

[5] Para mayor información, refiérase a Jacobson, Ivar. Et.al. *Object Oriented Software Engineering*. Addison_Wesley 4ª ed. P.p. 281

[6] Martin, James y Odell, James J., op.cit., p.p.461

La **herencia de clases** que proporciona la reutilización de código y datos a través de jerarquías de generalización no es soportada por los lenguajes no OO. Por lo tanto, todo lo que se construya en este ambiente debe utilizar un código explícito del programador. Para esto, James y Odell aconsejan soluciones no prácticas[7]:

- Copiado físico del código a partir de los módulos del supertipo
- Llamar a la rutina en módulos del supertipo. En vez de copiar el código, las operaciones podrían llamarse de cualquier módulo al módulo de su supertipo.
- Construir un sistema de soporte de la herencia.

En las implementaciones OO para la **selección del método**, el usuario sólo necesita especificar la operación que debe aplicarse a uno o más objetos y el sistema eligè el método adecuado para los parámetros especificados, buscándolo en la clase actual o en la jerarquía de clases. Sin embargo, los lenguajes no OO no reconocen las solicitudes de operaciones y la selección del método. No obstante, se podría simular de esta forma:

- Codificar la lógica de la selección dentro de la rutina que lo solicita.
- Construir un sistema de soporte de la selección del método.

Como se pudo ver, en términos prácticos, el soporte para métodos orientados a objetos debería estar incorporado directamente en el lenguaje de programación que se va a utilizar. De otra manera la implementación de mecanismos propios para el soporte de herencia o selección de métodos sería un problema mayor, en ocasiones equiparable al sistema por implementar.

Afortunadamente para nosotros, el panorama no es tan sombrío cuando hablamos de **sistemas de información relacionales y lenguajes de programación no OO**. Los SMBDR proveen una solución general a los problemas de almacenamiento de información,

[7] Martin, James y Odell, James J. *Análisis y Diseño Orientado a Objetos*, p.p.461

acceso concurrente a la base de datos, integridad de estos datos, seguridad y mecanismos de respaldo.

Sin embargo no todo es felicidad, el diseño de un sistema de información basado en SMBDR tiene un problema tradicional, éste consiste en la separación del diseño en dos partes; el diseño de los datos (lo que es una actividad para los expertos en bases de datos), y el diseño del software para el procesamiento de las transacciones sobre la base de datos (lo que se encarga a los desarrolladores de aplicaciones).

Estas técnicas de desarrollo tienen grandes ventajas, sin embargo, acarrean problemas muy reales. Los problemas provienen de las diferencias culturales entre los diseñadores de bases de datos y los programadores. "Los diseñadores de bases de datos intentan comprender el mundo en términos de tablas monolíticas y persistentes, mientras que los desarrolladores de aplicaciones modelan el mundo en términos de flujos de control"[8]

Así pues, resulta imposible alcanzar una integridad de diseño en un sistema complejo a menos que las diferencias entre ambos grupos sean atenuadas. Se ha comentado que una de las ventajas del modelo a objetos es el de romper las barreras entre analistas, diseñadores, programadores y usuarios finales al hablar todos en los mismos términos referentes a objetos. De esta manera, la arquitectura OO del SICPRH, como producto de un AOO, reconcilia notoriamente estas diferencias entre diseñadores de bases de datos y desarrolladores de aplicaciones.

Otra de las ventajas de las técnicas OO de las que haremos uso consiste en las similitudes entre el modelo a objetos y el modelado de base de datos. "La tecnología de las bases de datos relacionales... tiene contribuciones a la fundación del modelo a objetos, principalmente en el enfoque de los datos por medio del modelo de entidad-relación (E-R). En el enfoque E-R, el mundo es modelado en términos de sus entidades, los atributos de esas entidades y sus relaciones"[9]. Nótese la similitud entre una entidad y un objeto. Algunas

[8] Booch, op.cit. ,p.p. 377
[9] Booch, op. cit. p.p. 37

técnicas se valen de estas semejanzas para intentar pasar de un diagrama de objetos a un diagrama E-R y viceversa.

Lo que resta de este capítulo documentará la manera de obtener un diseño que permita una transición suave a la etapa de implementación. Para lograr esto, nuestros siguientes pasos irán encaminados hacia tres objetivos; el primero consistirá en separar nuestra arquitectura de clases y objetos para construir una arquitectura de datos y otra de programas. Nuestro segundo objetivo será definir las políticas para aspectos comunes del SICPRH (recuérdese que una mala definición de las políticas puede arruinar la arquitectura más robusta); el tercero consistirá en detallar los programas a través de algoritmos.

IV.3.1. Diseño Arquitectónico de los datos

En una base de datos relacional la información se concentra en tablas. En estas tablas sólo es posible almacenar tipos de datos primitivos, tales como cadenas de caracteres, valores numéricos, datos del tipo fecha, etc. Esta situación nos provoca problemas en nuestra ambición de trasladar nuestra arquitectura OO hacia una arquitectura relacional. Primero porque en las tablas sólo pueden almacenarse datos y no operaciones (las que provocan el comportamiento de nuestros objetos). Segundo porque las tablas no soportan la estructura de datos complejos de los objetos, y por último, surge la dificultad para expresar las relaciones de herencia de los objetos en una base de datos.

El primer problema nos remite crear una arquitectura para los datos (atributos de los objetos) y otra para los programas (operaciones).

Nuestro segundo dilema se resolverá si podemos convertir las estructuras de complejas de los objetos en sus correspondientes datos primitivos.

LAS CLASES SE CONVIERTEN EN TABLAS

La primera decisión consiste en qué clases y cuáles atributos de estas clases deben ser almacenados en la base de datos. Así entonces, cada una de esas clases serán representadas por al menos una tabla. El mapeo que convertirá a nuestras clases (o al menos sus atributos) en tablas ha sido descrito por Jacobson.[10]

1. Asignar una tabla para cada clase.
2. Cada atributo de una clase puede llegar a ser un campo de una relación (tabla). Si el atributo es complejo debe ser descompuesto en los tipos de datos primitivos que maneje el SMBD. Los campos que resulten de esta descomposición podrían situarse en la tabla que representa a la clase o pueden generar una tabla adicional.
3. El atributo que identifique a una instancia de cada clase es candidato para convertirse en llave primaria. Es deseable que este identificador sea invisible al usuario, de preferencia que pueda ser generado por el sistema. Bajo cualquier circunstancia este atributo candidato debe cumplir con las reglas de unicidad y las características que el modelo relacional le exige a una llave primaria.
4. Ahora, cada instancia de la clase será representada por un renglón (registro) en la tabla generada.
5. Cada asociación entre clases que posea una cardinalidad [N:N], es decir de muchos a muchos, podría dar origen a una tercera tabla que sirva de enlace entre las tablas que representan a las clases.

En este momento es necesario hacer una observación, al igual que el análisis y el diseño orientado a objetos, el proceso de transformar una estructura de objetos a una base de datos relacional no significar seguir una receta de cocina. Los puntos anteriores constituyen guías valiosas para esta actividad, pero no son verdades absolutas.

[10] Jacobson, Ivar *Object Oriented Software Engineering* op.cit., p.p.277

Hecha la aclaración pertinente podemos seguir con nuestra empresa. Una aplicación inteligente de los puntos anteriores nos arrojaría un primer diseño lógico de la base de datos, el siguiente paso consiste en refinar este diseño tomando en cuenta los postulados del modelo relacional.

El propósito más simple e importante en el diseño de una base de datos relacional consiste en la idea de almacenar cada cosa en un solo lugar. Esto elimina redundancia, simplifica el proceso de actualización de la información, facilita el mantenimiento de la integridad de la base de datos (esto es, autoconsistencia y correctez) y reduce los requerimientos de almacenamiento. Alcanzar este último objetivo no es tarea fácil (y no siempre es importante). No obstante, es la característica de diseño más deseable.

La teoría de la normalización de la información ha surgido como una técnica para conseguir ese objetivo. La normalización es una propiedad de una tabla en donde ésta puede formalizarse en diferentes niveles, dependiendo si satisface las siguientes situaciones.[11]

- Primera forma de normalización (1NF por las siglas en ingles de *First Normal Form*). Cada atributo representa un valor atómico, es decir, los atributos no pueden descomponerse en subatributos.
- Segunda forma de normalización (2NF). La tabla se encuentra en 1NF y cada atributo depende enteramente de la llave primaria. Los atributos son independientes funcionalmente.
- Tercera forma de normalización (3NF). La tabla está en el 2NF y ningún atributo depende de algún otro. Los atributos son independientes mutuamente.

[11] Booch, Grady. op.cit., p.p.395

"El diseño de una base de datos generado por un modelo a objetos nos conduce generalmente a la tercera forma de normalización (3NF). La razón es obvia, la tercera forma de normalización exige que cada renglón de una tabla consista de una llave primaria y un número de atributos mutuamente excluyentes".[12] De igual forma, los modelos OO son modelos de la realidad en donde intentamos identificar objetos únicos y los atributos que naturalmente poseen. Por esta razón podemos decir que un buen modelo OO está normalizado.

Actualmente se han desarrollado algunas técnicas para el diseño de bases de datos relacionales basadas en el modelo a objetos. La experiencia de la aplicación de éstas ha sido buena, e incluso superior a otras técnicas, hasta el punto de que algunos autores han concluido que "construir un modelo de datos desde un número pequeño de objetos es superior al enfoque tradicional de coleccionar todos los atributos, indicar las dependencias funcionales y sintetizar tablas."[13]

Una vez que se ha refinado el diseño lógico de la base de datos a través de la normalización, si es que este paso fue requerido, es necesario considerar el rendimiento en el acceso a los datos (en ocasiones una tabla en 3NF presenta este inconveniente). Este posible problema podría resolverse creando índices especiales, o en el último de los casos, denormalizando la tabla.

RELACIONES DE HERENCIA

Existen básicamente dos formas de representar las relaciones de herencia entre clases:

1) Solamente las clases descendientes (subclases) son representadas por tablas, de esta manera la clase base se queda sin representación. Los atributos heredados son copiados a todas las tablas que sustituirán a las subclases.

[12] Jacobson, Ivar, op.cit.., p.p. 278
[13] Jacobson, Ivar, op.cit., p.p. 279

2) La clase base se convierte en una tabla, los atributos de especialización de las subclases se almacenan en tablas diferentes (contemplando cada una de éstas una referencia a la tabla de la clase base).

Como ejemplo, considérese la relación de herencia que presenta la clase ficticia personas con las subclases ficticias empleados e hijos.

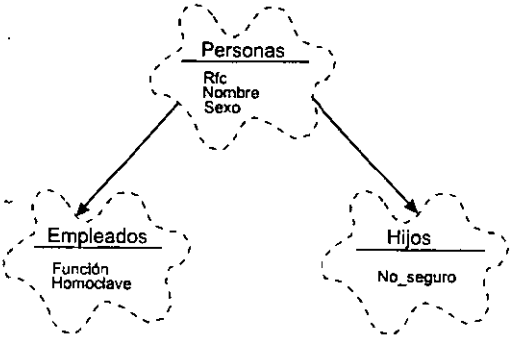


Figura 4.1

La primera alternativa de la que hablamos consistiría en crear una tabla para las subclases empleados e hijos y copiar los atributos de la clase base personas a ambas tablas, como se muestra abajo.

Empleados

RFC	NOMBRE	SEXO	FUNCION	HOMOCLAVE
PESJ-740215	Juan Pérez Sánchez	M	Pintor	SB4

Hijos

RFC	NOMBRE	SEXO	NO_SEGURO
PEGC-890512	Carlos Pérez García	M	01321

Figura 4.2

La segunda alternativa es almacenar los atributos de la clase base personas como una tabla por sí misma, relacionándola con las tablas que sustituyen a las subclases empleados e hijos a través de una llave primaria.

Personas

RFC	NOMBRE	SEXO
PESJ-740215	Juan Pérez Sánchez	M
PEGC-890512	Carlos Pérez García	M

Empleados

FUNCION	HOMOCLAVE
Pintor	SB4

Hijos

NO_SEGURO
01321

Figura 4.3

¿Cuál alternativa es mejor? No existe mejor solución para todos los casos. La primera alternativa es más rápida en el acceso a los datos porque que no necesita relacionar tablas, el inconveniente es que se incrementa el tamaño de la base de datos al existir duplicidades de información (los atributos que originalmente estaban en la clase base), además, si ocurrieran cambios a estos atributos heredados provocarían modificaciones a todas las tablas que representan a las subclases.

La segunda alternativa incluye menos redundancia, pero el acceso a los datos es más lento (por el proceso de relacionar tablas), además, se podrían presentar problemas si los identificadores necesitaran ser iguales en la tabla de la clase base (por ejemplo, si tuviéramos una clase base persona y las subclases maestros y alumnos, no habría posibilidad de que una persona fuera maestro y alumno al mismo tiempo).

BASE DE DATOS DEL SICPRH

Ahora estamos en condiciones de describir la base de datos del SICPRH, en esta actividad se ha considerado la teoría que describimos anteriormente.

En el AOO descubrimos las clases que conforman el dominio del sistema, pero en ese momento no nos interesó revelar el número de instancias que pueden estar involucradas en las relaciones entre clases, es decir, no especificamos la cardinalidad de las relaciones. En el diseño debemos hacerlo.

Los diagramas siguientes muestran este hecho.

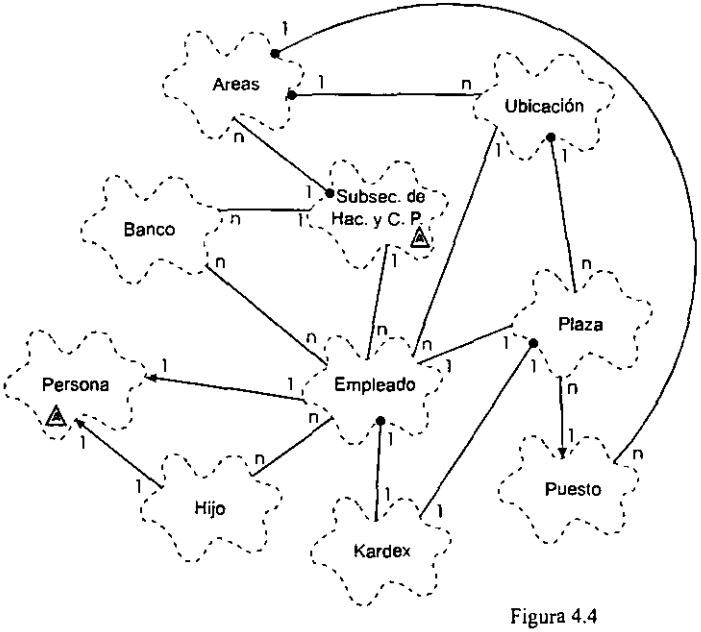


Figura 4.4

La notación de Booch emplea rótulos encima, abajo o a los lados de las líneas que expresan relaciones entre clases para denotar cardinalidad. Por ejemplo, como se puede ver en el diagrama de arriba, una instancia de áreas puede tener n instancias de la clase ubicación.

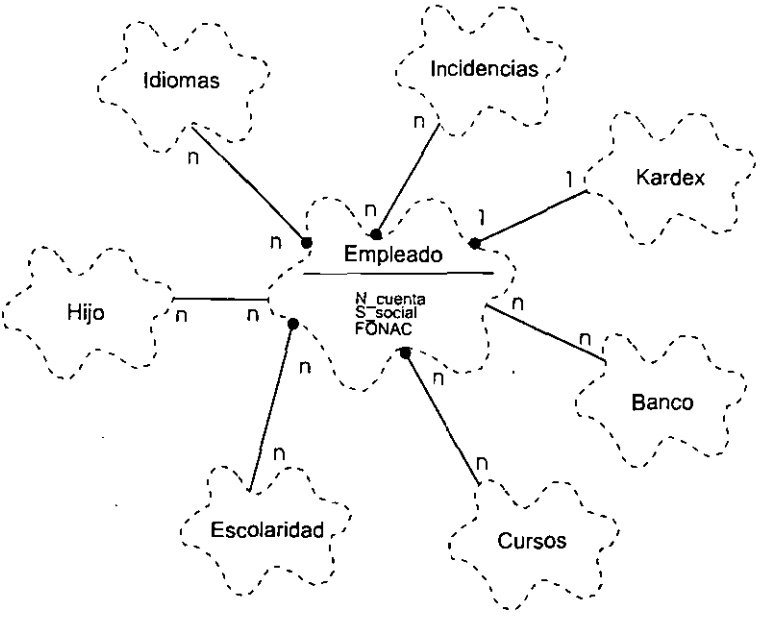


Figura 4.5

Las relaciones que existen entre las clases, con excepción de las relaciones de herencia, involucrarán la creación de campos comunes en las tablas que estarán relacionadas. En donde este campo (o conjunto de campos) represente un identificador del objeto (registro), estaremos hablando de una llave primaria; donde su importancia radique solamente en relacionar tablas, estaremos frente a un campo llave foránea.

Tomando en cuenta el primer diagrama de clases, podemos notar que Subsecretaría de Hacienda y Crédito Público y Persona son clases abstractas, por este motivo no tendrán representación en tablas. En caso contrario, la clase áreas se convierte en una tabla con los siguientes campos.

AREAS[14]

Nombre del campo	Tipo	Observaciones
CP1*	Carácter (3)	Clave de área presupuestal.
DESCRIPCION	Carácter (60)	Nombre del área presupuestal.

La clase ubicación da origen a una tabla con el mismo nombre. La relación que existe entre esta clase y la clase áreas será manejada por un campo (CP1) que se encuentra en ambas tablas. Este campo tiene roles distintos dependiendo de la tabla en la que se encuentre, en AREAS es un identificador que necesita unicidad, por lo que se convierte en una llave primaria; en UBICACION es sólo un atributo que expresa el área a la que pertenece cada ubicación, no necesita unicidad (puesto que pueden existir n ubicaciones para un área presupuestal), pero es necesario para relacionar ambas entidades, es lo que comúnmente se llama una llave foránea.

UBICACION

Nombre del campo	Tipo	Observaciones
UBICA*	Carácter (5)	Clave de ubicación.
DESCRIPCION	Carácter (80)	Nombre de la ubicación.
CP1**	Carácter (3)	Clave de área presupuestal.

La clase puesto está relacionada con las clases áreas y plaza. Debido a que los puestos son comunes a todas las áreas presupuestales, la tarea de especificar cuales puestos corresponden a cada área resultaría en redundancia para la base de datos. En cambio, la relación existente entre las clases puesto y plaza es una relación de herencia muy importante. La manera en cómo abordaremos esta cuestión será la de representar la clase base y subclase en tablas y relacionarlas, tal como se explicó en el subtema anterior.

Las tablas PUESTOS y PLAZAS (originadas por las clases que llevan el mismo nombre), estarán relacionadas por la conjunción de los campos CP2 y CP3. Esta conjunción es una llave primaria en la tabla PUESTOS y tiene un rol de llave foránea en la tabla PLAZAS.

[14] Para diferenciar los nombres de las tablas de los nombres de las clases, a los primeros los escribiremos en mayúsculas. Por otra parte, los campos que estén marcados con un asterisco (*) son campos llaves (llaves primarias), los que tengan dos asteriscos (**) son llaves foráneas.

PUESTOS

Nombre del campo	Tipo	Observaciones
* { CP2	Carácter (2)	Primer subidentificador de puesto.
* { CP3	Carácter (5)	Segundo subidentificador de puesto.
DESCRIPCION	Carácter (35)	Nombre del puesto.
NIVEL	Carácter (4)	Nivel del puesto en el tabulador.
SUBNIVEL	Carácter (3)	Subnivel para los niveles 27.
TIPO_PTO	Carácter (1)	Tipo de puesto: C = Confianza B = Base H = Honorarios
GRUPO	Carácter(1)	Grupo al que pertenece la plaza: O = Operativo E = Enlace M = Mando medio S = Subdirector D = Director
FIEF	Carácter (1)	Control utilizado por la DGRH.
SUELDO_NETO	Numérico (12,2)	Sueldo neto para ese puesto.
PRODUCTIVIDAD	Numérico (12,2)	Monto correspondiente al bono mensual de productividad.

En el diagrama de clases de la figura 4.4 se puede notar que la clase plaza está relacionada con las clases ubicación, puesto, empleado y kardex, sin embargo, la primera de estas relaciones no tiene sentido porque ubicación está contenido en la clase áreas (la verdadera relación es con áreas). Esas relaciones provocan la existencia de tres llaves foráneas: el campo CP1, la combinación de los campos CP2 y CP3, y por último, el campo RFC, los cuales relacionaran a la tabla PLAZAS con las tablas AREAS, PUESTOS y EMPLEADOS respectivamente. La llave primaria en esta tabla PLAZAS es el campo FOLIO, este último servirá para relacionarse con la tabla KARDEX.

PLAZAS

Nombre del campo	Tipo	Observaciones
FOLIO*	Carácter (4)	Identificador de la plaza.
CP1**	Carácter (3)	Clave del área pptal a la que pertenece la plaza.
** { CP2	Carácter (2)	1er. Subidentificador del puesto al que pertenece la plaza.
** { CP3	Carácter (5)	2do. Subidentificador del puesto al que pertenece la plaza.
CP4	Carácter (5)	Número de plaza.
RFC**	Carácter (13)	Rfc del empleado que tiene asignada esta plaza.
CVE_MOV**	Carácter (2)	Ultimo movimiento de la plaza

PLAZAS (Continuación)

Nombre del campo	Tipo	Observaciones
F_EFECTIV	Fecha	Fecha último movimiento de la plaza.
SITUACION	Carácter (2)	Situación de la plaza: A = Activa LC = En licencia con goce de sueldo LM = En licencia con medio sueldo LS = En licencia sin goce de sueldo V = Vacante R = En trámite de reubicación.
NOMBRE	Carácter (20)	En caso de plaza vacante o en licencia, este campo indica el nombre del último empleado que ostentaba la plaza
OBSERVACIONES	Carácter (20)	

El campo CVE_MOV de la tabla PLAZAS merece una atención especial. Éste es producto de una decisión de diseño estratégica acerca de la manera en cómo se controlarán los movimientos de plazas y empleados que se detallaron en el capítulo anterior. La manera más sencilla de manejar estos cambios es a través de un campo que nos permita aplicar los movimientos de empleados en un solo módulo de captura y relacionar la tabla PLAZAS con un catálogo de movimientos (tabla MOVIMIENTOS)

MOVIMIENTOS

Nombre del campo	Tipo	Observaciones
CVE_MOV*	Carácter (2)	Clave del movimiento.
DESCRIPCION	Carácter (60)	Descripción del movimiento, por ejemplo, licencia, renuncia, jubilación, etc..

De acuerdo al análisis previo, los sigs. movimientos son válidos.

CVE_MOV	DESCRIPCION
LC	Licencia con goce de sueldo
LM	Licencia con medio sueldo
LS	Licencia sin goce de sueldo
B	Baja
BC	Baja por conversión.
BR	Baja por renuncia.
NI	Nuevo ingreso.
RI	Reingreso
P	Promoción
R	Reubicación
C	Cambio de Adscripción

Veamos ahora la manera en cómo manejaremos el mapeo de las clases persona, empleado e hijo y sus relaciones de herencia hacia nuestra base de datos relacional.

Por ser una clase abstracta, persona no tendrá representación en tablas. La relación de herencia que posee esta clase base con las subclases empleado e hijo estará representada por dos tablas, una para cada subclase. Los atributos de la clase base persona serán copiados a ambas tablas.

La tabla EMPLEADOS, que representa a la clase empleado, posee como llave primaria al campo RFC, éste le permitirá relacionarse con las tablas IDIOMAS, INCIDENCIAS, KARDEX, CURSOS, ESCOLARIDAD, HIJOS y PLAZAS. Para relacionarse con las tablas BANCOS y UBICACIÓN se hará uso de las llaves foráneas CVE_BANCO, BANCO_SAR y UBICA. Además, EMPLEADOS mantendrá una relación con MOVIMIENTOS por medio del campo CVE_MOV: De esta manera se consideran todas las relaciones de empleado con las demás clases que se observan en las figuras 4.4. y 4.5 (salvo la relación con la clase Secretaría de Hacienda y Crédito Público, la cual es una clase abstracta).

EMPLEADOS

Nombre del campo	Tipo	Observaciones
RFC*	Carácter (13)	Registro Federal de Causante del empleado.
NOMBRE	Carácter (20)	Nombre de pila.
PATERNO	Carácter (15)	Apellido paterno.
MATERNO	Carácter (15)	Apellido materno.
CALLE[15]	Carácter (45)	Domicilio: calle, no. exterior e interior.
COLONIA	Carácter (30)	Domicilio: colonia.
DELEGACION	Carácter (30)	Domicilio: delegación.
CP	Carácter (5)	Domicilio: codigo postal
TELEFONO	Carácter (7)	Teléfono.
SEXO	Carácter (1)	Sexo: M = Masculino F = Femenino

[15] El atributo domicilio se ha descompuesto en los subatributos calle, colonia, delegación y código postal.

EMPLEADOS (Continuación)

Nombre del campo	Tipo	Observaciones
EDO_CIVIL	Carácter (1)	Estado civil: S = Soltero C = Casado U = Unión libre D = Divorciado V = Viudo
F_NAC	Fecha	Fecha de nacimiento.
SITUACION	Carácter (2)	Situación del empleado: A = En activo I = Inactivo LC = En licencia con goce de sueldo LM = En licencia con medio sueldo LS = En licencia sin goce de sueldo R = En trámite de reubicación
CVE_MOV**	Carácter (2)	Clave del último movimiento.
F_EFECTIV	Fecha	Fecha del último movimiento del empleado (nvo. ingreso, promoción, licencia, etc.).
CVE_QUINQUE**	Carácter (2)	Número de quinquenios laborados
FUNCION	Carácter (35)	Actividades que realiza el personal.
IMPORTE	Numérico (9,2)	Compensaciones (bonificación extra).
N_CUENTA	Carácter (18)	No. de cuenta de cheques para depósito bancario por concepto de sueldo.
CVE_BANCO**	Carácter (2)	Cve. banco que maneja la cuenta de cheques.
BANCO_SAR**	Carácter (2)	Cve. banco que maneja el SAR.
S_SOCIAL	Carácter (10)	Número de filiación al ISSSTE.
SINDICATO	Carácter (1)	¿Cotiza al S.N.T.H.? S = Si N = No
TURNO	Carácter (1)	Turno de labores: M = Matutino V = Vespertino C = Horario completo (matutino y vespertino)
F_GOBIERNO	Fecha	Fecha de ingreso al gobierno federal.
F_SHCP	Fecha	Fecha de ingreso a la S.H.C.P.
FOLIO_HGO	Carácter (7)	No. de contrato de seguro de vida con Aseguradora Hidalgo.
FONAC	Carácter (1)	¿Está inscrito al FONAC? S =Si N = No
UBICA**	Carácter (5)	Clave de ubicación a donde labora el empleado.

La tabla HIJOS hereda los atributos de la subclase persona. La relación con la tabla EMPLEADOS que se detalla en el diagrama de clases de la figura 4.4. y 4.5 genera la llave foránea RFC.

Esta tabla posee una llave primaria RFC_HIJO, de esta forma sólo podrá existir un solo registro para cada hijo, incluso en los casos en los que ambos cónyuges son empleados hacendarios.

HIJOS

Nombre del campo	Tipo	Observaciones
RFC_HIJO*	Carácter (10)	Rfc del hijo.
RFC**	Carácter (13)	Rfc del padre o madre hacendario.
NOMBRE	Carácter (20)	Nombre de pila del hijo.
PATERNO	Carácter (15)	Apellido paterno.
MATERNO	Carácter (15)	Apellido materno.
F_NAC	Fecha	Fecha de nacimiento.
SEXO	Carácter (1)	Sexo: (M ó F)

La clase kardex también genera una tabla con el mismo nombre, pero el manejo de esta abstracción sufrirá un cambio significativo. Conceptualmente, cualquier instancia de empleado tenía una y sólo una instancia de kardex; de igual manera, una instancia de plaza contenía un y sólo un kardex, esto se debía a que cualquier instancia de kardex era responsable de almacenar todos los movimientos de su clase agregada (ya fuera empleado o plaza), podíamos decir que esta clase es sumamente especializada.

La manera más sencilla de representar este comportamiento sería crear una tabla que centralice todos los movimientos tanto de empleados como de plazas. De esta manera, tendríamos una tabla KARDEX con llaves foráneas que la relacionen con las tablas EMPLEADOS Y PLAZAS, estas llaves serían los campos RFC y FOLIO respectivamente. La relación de uno a uno que existía entre las clases empleado y kardex y las clases plaza y kardex se convierte en relaciones de cardinalidad [1..N].

Además, se guardan todos los movimientos de los empleados y plazas en las llaves foráneas MOV_PLAZA y CVE_MOV, las cuales relacionan la tabla KARDEX con MOVIMIENTOS.

KARDEX

Nombre del campo	Tipo	Observaciones
RFC**	Carácter (13)	Rfc del empleado.
FOLIO**	Carácter (4)	Folio de la plaza.
NOMBRE	Carácter (20)	Nombre de pila.
PATERNO	Carácter (15)	Apellido paterno.
MATERNO	Carácter (15)	Apellido materno.
CP1	Carácter (3)	Cve. área pptal a la que pertenece la plaza.
CP2	Carácter (2)	1er. Subidentificador del puesto al que pertenece la plaza.
CP3	Carácter (5)	2do. Subidentificador del puesto al que pertenece la plaza.
CP4	Carácter (5)	Número de plaza.
NIVEL	Carácter (4)	Nivel del puesto.
SUBNIVEL	Carácter (3)	Subnivel para los niveles 27.
CVE_MOV**	Carácter (2)	Clave del movimiento de empleado.
MOV_PLAZA**	Carácter (2)	Tipo de movimiento de plaza: B = Baja LC = Licencia con goce de sueldo LM = Licencia a medio sueldo LS = Licencia sin goce de sueldo V = Vacante R = Reubicación
F_EFECTIV	Fecha	Fecha de efectividad del movimiento.

La abstracción banco genera la tabla BANCOS, ésta posee como llave primaria al campo CVE_BANCO. La relación que guarda esta abstracción con la clase empleado (que originalmente tenía una cardinalidad [n..n]), será manejada a través de dos relaciones de cardinalidad [1..1], estas relaciones serán encargadas a las llaves foráneas CVE_BANCO y BANCO_SAR de la tabla EMPLEADOS.

BANCOS

Nombre del campo	Tipo	Observaciones
CVE_BANCO*	Carácter (2)	Clave del banco.
DESCRIPCION	Carácter (20)	Razón social (nombre) del banco.

Debido a que una instancia de la clase empleado puede tener n instancias de la clase escolaridad, y al mismo tiempo una instancia de la tabla escolaridad puede tener n instancias de la clase empleado (a diferencia de la relación entre empleado y bancos en donde un empleado tiene dos asociaciones de uno a uno con banco), el manejo de las relaciones entre esas clases será diferente.

La teoría que se expuso anteriormente proponía manejar estos problemas a través de una tercera tabla, esta tabla es un catálogo.

C_ESCOLARIDAD

Nombre del campo	Tipo	Observaciones
CVE_NIVEL*	Carácter (3)	Clave del nivel de escolaridad.
DESCRIPCION	Carácter (30)	Nombre de nivel de escolaridad.

ESCOLARIDAD

Nombre del campo	Tipo	Observaciones
RFC**	Carácter (13)	Rfc del empleado.
CVE_NIVEL**	Carácter (3)	Clave del nivel de escolaridad.
PORCENTAJE	Carácter (3)	Porcentaje de créditos cubiertos en el nivel de escolaridad.
INSTITUTO	Carácter (20)	Institución educativa.

La tabla C_ESCOLARIDAD contendrá todos los niveles de escolaridad que puedan existir, es decir, ésta es un catálogo de niveles de escolaridad. La tabla ESCOLARIDAD concentrará los niveles de escolaridad de todo el personal, la llave foránea RFC servirá para relacionarse con EMPLEADOS, la llave foránea CVE_NIVEL servirá de enlace con el catálogo C_ESCOLARIDAD.

De esta manera, hemos convertido una relación de muchos a muchos en dos relaciones de uno a muchos, esta estructura permite que un empleado tenga varios niveles de escolaridad y que un nivel de escolaridad pueda ser compartido por varios empleados.

Esta misma solución se aplica a las relaciones entre las clases idiomas y empleado.

C_IDIOMAS

Nombre del campo	Tipo	Observaciones
CVE_IDIOMA*	Carácter (2)	Clave del idioma.
DESCRIPCION	Carácter (20)	Nombre del idioma.

IDIOMAS

Nombre del campo	Tipo	Observaciones
RFC**	Carácter (13)	Rfc del empleado.
CVE_IDIOMA**	Carácter (2)	Clave del idioma.
PORCENTAJE	Carácter (3)	Porcentaje de dominio del idioma.
INSTITUTO	Carácter (20)	Institución educativa.

Lo mismo sucede con la clase cursos.

C_CURSOS

Nombre del campo	Tipo	Observaciones
CVE_CURSO*	Carácter (3)	Clave del cursos.
TIPO	Carácter (1)	Tipo : C = Curso D = Diplomado
DESCRIPCION	Carácter (60)	Nombre del curso o diplomado.

CURSOS

Nombre del campo	Tipo	Observaciones
RFC**	Carácter (13)	Rfc del empleado.
CVE_CURSO**	Carácter (3)	Clave del curso.
F_INICIO	Fecha	Fecha de inicio.
F_TERMINO	Fecha	Fecha de termino.
INSTITUTO	Carácter (20)	Instituto educativo.
ESTADO	Carácter (1)	A = Aprobado. R = Reprobado.

En la clase incidencias también se presenta la misma relación de muchos a muchos con la clase empleado. Sin embargo, la realización de un catálogo de incidencias no es aconsejable porque el número de éstas es reducido (sólo tres tipos de incidencias nos interesan). En su lugar, crearemos una tabla que concentre las incidencias de todo el personal y tipificaremos el campo CVE_INCI.

INCIDENCIAS

Nombre del campo	Tipo	Observaciones
RFC**	Carácter (13)	Rfc del empleado.
CVE_INCI	Carácter (1)	Clave de incidencia: F = Falta total E = Económico L = Licencia Médica C = Cuidados Maternos
F_INICIO	Fecha	Fecha de inicio.
F_TERMINO	Fecha	Fecha de termino.
ESTADO	Carácter (1)	¿Aplicación de descuento? S = Si N = No.

Por último, el número de quinquenios trabajados por un empleado le retribuye cierto monto en sus percepciones, dicho monto debe ser considerado en el sistema. Esta complejidad en ese atributo (complejidad que nos fue informada de último momento) será manejada por una catálogo de quinquenios.

QUINQUENIOS

Nombre del campo	Tipo	Observaciones
CVE_QUINQUE*	Carácter (2)	Clave del quinquenio.
MONTO	Numérico (8,2)	Percepciones por concepto de quinquenios

Esta tabla estará relacionada con la tabla de empleados a través de la llave primaria CVE_QUINQUE.

Hasta el momento se han indicado las tablas que conforman la base de datos del SICPRH y se mostraron las llaves primarias y foráneas para cada tabla. Las figuras que se presentan a continuación exhiben las relaciones entre todas las tablas.

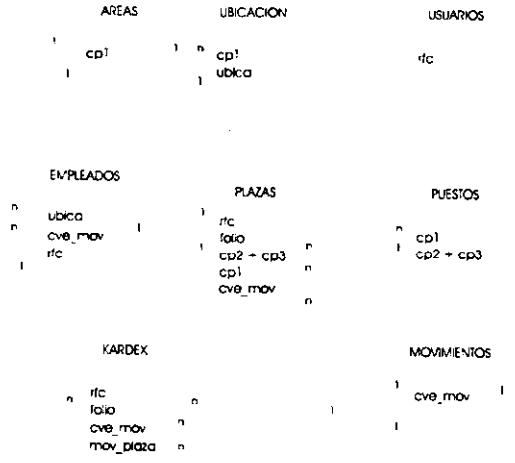


Figura 4.6

Ambos diagramas[16] son complementarios y deben capturarse como una sola base de datos en el SMBD.

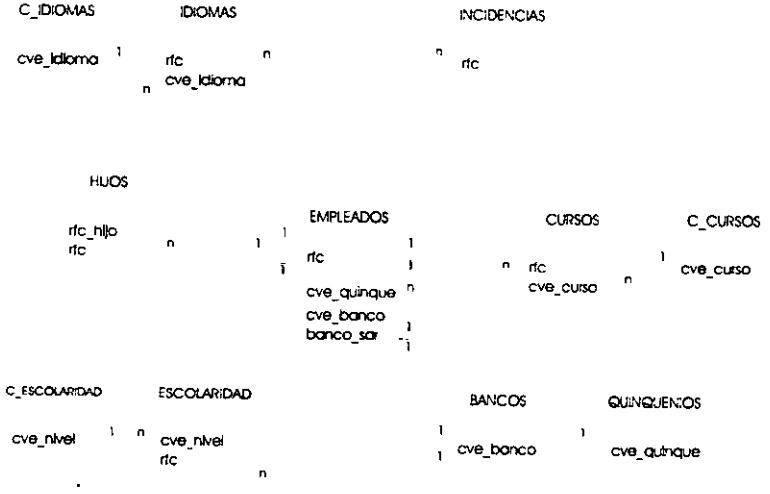


Figura 4.7

[16] Observe la tabla USUARIOS en la figura 4.6. Esta tabla aislada registrará a los usuarios con acceso al sistema, el detalle de esto se encuentra en Políticas para aspectos comunes del SICPRH.

Nótese el parecido de estos diagramas con los diagramas de clases de las figs. 4.4 y 4.5

La sig. lista muestra el área de trabajo (SELECT) que utilizará cada una de las tablas

Nombre de la tabla	Area de trabajo	Nombre de la tabla	Area de trabajo
Areas	1	C_escolaridad	10
Ubicación	2	Escolaridad	11
Puestos	3	C_idiomas	12
Plazas	4	Idiomas	13
Movimientos	5	C_cursos	14
Empleados	6	Cursos	15
Hijos	7	Incidencias	16
Kardex	8	Quinquenios	17
Bancos	9	Usuarios	18

Abajo se muestran los índices para cada tabla; los campos llave primaria son reconocidos por Visual FoxPro como índices principales, las llaves foráneas serán manejadas como índices normales.

Tabla	Nombre del índice	Campo o expresión al que hace referencia el índice	Tipo de índice
Areas	Cp1	CP1	Principal
Ubicación	Ubica	UBICA	Principal
Ubicación	Cp1	CP1	Normal
Puestos	Puesto	CP2 + CP3	Principal
Plazas	Folio	FOLIO	Principal
Plazas	Cp1	CP1	Normal
Plazas	Puesto	CP2 + CP3	Normal
Plazas	Rfc	RFC	Normal
Plazas	Cve_mov	CVE_MOV	Normal
Movimientos	Cve_mov	CVE_MOV	Principal
Empleados	Rfc	RFC	Principal
Empleados	Cve_quin	CVE_QUINQUE	Normal
Empleados	Cve_banco	CVE_BANCO	Normal
Empleados	Banco_sar	BANCO_SAR	Normal
Empleados	Cve_mov	CVE_MOV	Normal
Empleados	Ubica	UBICA	Normal
Hijos	Rfc_hijo	RFC_HIJO	Principal
Hijos	Rfc	RFC	Normal
Kardex	Rfc	RFC	Normal
Kardex	Folio	FOLIO	Normal
Kardex	Cve_mov	CVE_MOV	Normal
Bancos	Cve_banco	CVE_BANCO	Principal

(Continuación)

Tabla	Nombre del índice	Campo o expresión al que hace referencia el índice	Tipo de índice
C_escolaridad	Cve_nivel	CVE_NIVEL	Principal
Escolaridad	Rfc	RFC	Normal
Escolaridad	Cve_nivel	CVE_NIVEL	Normal
C_idiomas	Cve_idioma	CVE_IDIOMA	Principal
Idiomas	Rfc	RFC	Normal
Idiomas	Cve_idioma	CVE_IDIOMA	Normal
C_cursos	Cve_curso	CVE_CURSO	Principal
Cursos	Rfc	RFC	Normal
Cursos	Cve_curso	CVE_CURSO	Normal
Incidencias	Rfc	RFC	Normal
Quinquenios	Cve_quin	CVE_QUINQUE	Principal

IV.3.2. Diseño arquitectónico de los programas

Se dijo anteriormente que la arquitectura de un sistema OO está formada por la estructura de sus clases, objetos y sus relaciones. En el SICPRH tenemos una arquitectura de los datos (especificada por su base de datos) y una arquitectura de los módulos o programas.

Aunque esta última labor está muy relacionada con los diseños estructurados, la transición de los métodos de nuestras clases originales hacia una arquitectura de programas no resulta una tarea muy difícil, sobretodo cuando se ha hecho el seguimiento del sistema desde la fase de concepto, el AOO y culminando con algunas actividades de DOO. La razón es simple (aún cuando no tenemos el prototipo evolutivo que propone Booch), el conocimiento del problema y la definición de algunas decisiones de diseño están hechas, el siguiente paso es conciliar la funcionalidad del sistema (a través de una jerarquía de módulos) y la interface del usuario.

El SICPRH estará compuesto por los siguiente subsistemas:

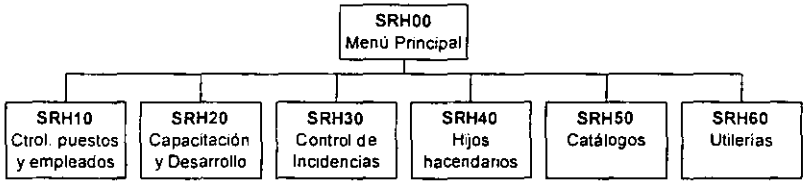


Figura 4.8

Los encabezados de cada recuadro indican el nombre del módulo, este nombre está formado por las letras SRH (Sistema de Recursos Humanos) y por un determinado número de dígitos que indican la posición del módulo dentro de la arquitectura del sistema.

El subsistema más importante es el de control de puestos y empleados, aquí se integrarán todos los programas que permitan los movimientos de plazas y de personal que se especificaron en los escenarios identificados en el AOO. Estos escenarios no representarán un módulo por sí mismos, al contrario, aprovechando que los movimientos posibles de una plaza son un subconjunto de los movimientos que se pueden dar en el personal, éstos se realizarán en un solo módulo, esta decisión de diseño nos permitirá ahorrar trabajo como programadores, nos facilitará el mantenimiento del sistema, y al mismo tiempo, le estaremos dando un mayor control al usuario sobre los movimientos que pueda hacer.

Los módulos de consulta mostrarán información a pantalla y permitirán al usuario la emisión de reportes. Los reportes, *queries*, formularios y/o archivos planos (código fuente) que sean parte de un mismo módulo tendrán el mismo nombre (el identificador para cada uno de éstos será su extensión de archivo).

El subsistema de control de puestos y empleados estará formado por los sigs. módulos:

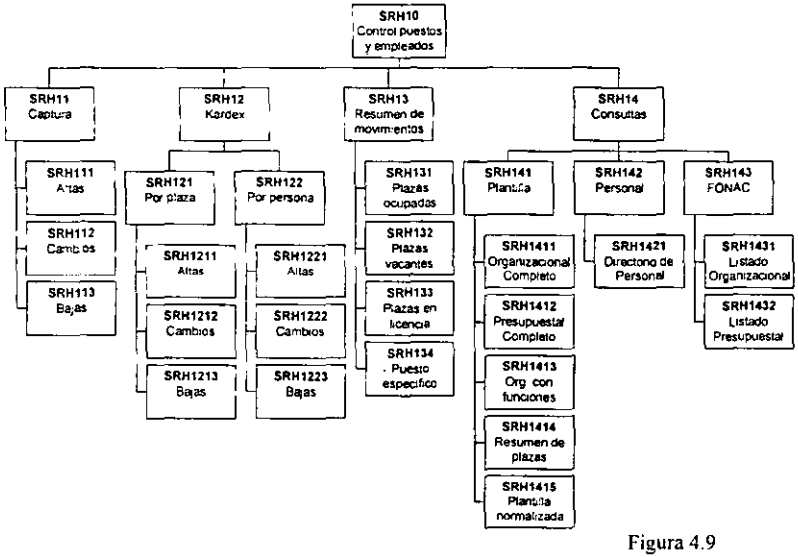


Figura 4.9

El subsistema de capacitación y desarrollo tendrá la siguiente estructura:

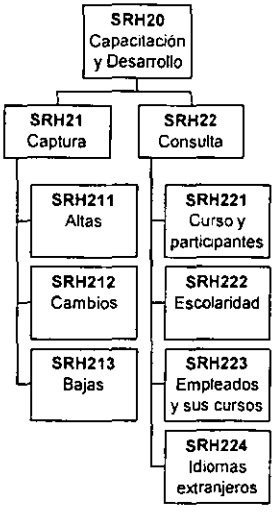


Figura 4.10

El subsistema de Control de incidencias y el Control de los hijos de empleados hacendarios tendrán la sig. estructura (Figs. 4.11 y 4.12 respectivamente).

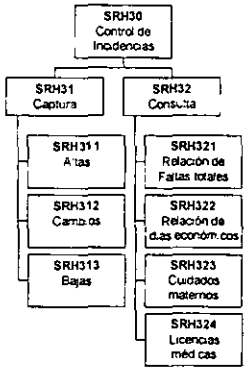


Figura 4.11

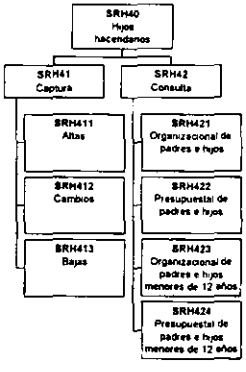


Figura 4.12

Catálogos es un grupo de módulos comunes a todos los subsistemas del SICPRH, sin embargo, como se verá posteriormente, cada clase de usuario será responsable de su propia información. En algunos módulos de estos catálogos estarán contemplados los comportamientos descritos por algunos escenarios secundarios, tal como la creación, modificación y eliminación de áreas presupuestales, etc.

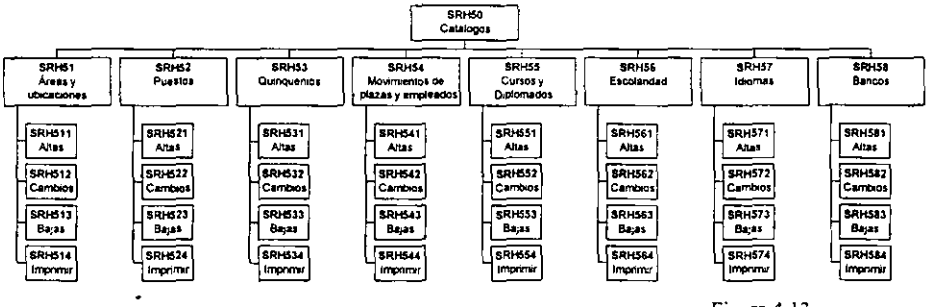


Figura 4.13

Por último, el SICPRH dará algunos servicios extra a sus usuarios, estos son básicamente las opciones de respaldar su información a discos flexibles y la de recuperar esta información desde estos discos hacia la base de datos del sistema.

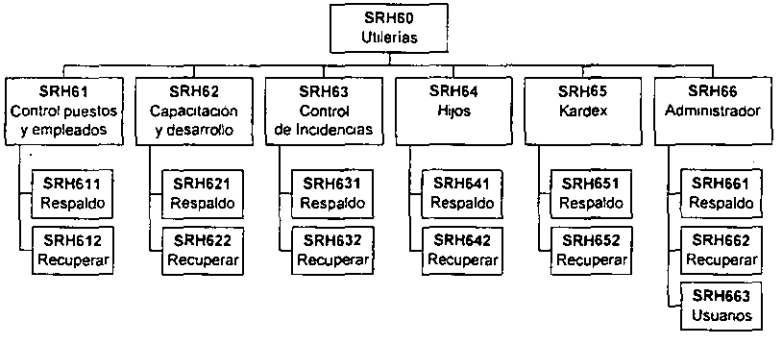


Figura 4.14

IV.3.3. Políticas para aspectos comunes del SICPRH

A continuación haremos una especificación de las políticas que se deben observar en la construcción del sistema, éstas se encuentran organizadas en los siguientes rubros:

INTEGRIDAD DE LOS DATOS

Proteger la integridad de los datos incluye asegurarse de que los usuarios autorizados no agregarán, cambiarán o eliminarán datos que harían que algunas partes de la base de datos fueran no válidas o no exactas.

El mantenimiento de la integridad de los datos en el ámbito de la base de datos se aplica si un usuario está cambiando valores mediante los formularios de la aplicación o

directamente en una ventana examinar (*browse*), por lo que ésta se aplicará en forma de integridad referencial sobre toda la base de datos.

La integridad referencial se implantará a nivel de base de datos mediante el Generador de integridad referencial de Visual FoxPro[17] siguiendo los siguientes lineamientos:

1. Para todas las tablas primarias (catálogos) de la base de datos, cuando se cambia un valor de la llave primaria, ese cambio se realiza también en las tablas secundarias (Cascada en actualización).
2. Los usuarios no podrán eliminar registros de alguna tabla primaria si existen registros relacionados en una tabla secundaria (Restringir en la eliminación).
3. Un usuario no podrá insertar registros en tablas secundarias que no coincidan con los registros de las tablas primarias (Restringir en la inserción).

Por otra parte, los índices que representan a las llaves primarias de alguna tabla serán del tipo Principal; los que representen llaves foráneas serán del tipo Normal. Los nombres de cada índice será idéntico al nombre del campo asociado, excepto para el índice llamado Puesto, el cual es la unión de dos campos (CP1+CP2).

SEGURIDAD

La seguridad de una base de datos incluye restringir el acceso a los datos. Los usuarios no autorizados no deben tener acceso a ciertas partes de la base de datos. Por este motivo se generará una tabla USUARIOS.

[17] El SMBDR que se va a utilizar para la implementación del sistema es VISUAL FOXPRO

USUARIOS

Nombre del campo	Tipo	Observaciones
RFC*	Carácter (13)	Rfc del usuario, sirve sólo como llave primaria.
NOMBRE	Carácter (20)	Nombre de pila del usuario
PATERNO	Carácter (15)	Apellido paterno
MATERNO	Carácter (15)	Apellido materno
TIPO_USUARIO	Carácter (1)	Grupo de usuarios: 1 = Administrador 2 = Control 3 = Operación 4 = Capacitación y desarrollo 5 = Selección
CONTRASEÑA	Carácter (10)	Contraseña cifrada

Antes de que un usuario obtenga acceso al SICPRH, éste debe identificarse como un empleado autorizado a través de un formulario de inicio de sesión. La siguiente tabla muestra los módulos a los que tiene acceso dicho usuario, dependiendo del grupo al que pertenezca.

Tipo de usuario	Acceso al sistema	Acceso a los catálogos	Acceso a las utilerías
Administrador.	SRH10, SRH20 SRH30, SRH40	SRH50	SRH60
Control.	SRH10	SRH501, SRH502 SRH503, SRH504 SRH505, SRH510	SRH61, SRH65
Operación.	SRH30	SRH509	SRH63
Capacitación y desarrollo.	SRH20	SRH506, SRH507 SRH508	SRH62
Selección.	SRH40		SRH64

El tener acceso a un módulo implica tener acceso a todos sus submódulos (véase las figuras 4.8 a la 4.14 en donde se especifica la arquitectura de los programas).

FORMULARIOS

El SICPRH utilizará tres tipos de formularios:

1. Formularios modales que devuelven un valor (por ejemplo, el formulario de inicio de sesión que solicita una contraseña al usuario).
2. Formularios para la introducción de datos (por ejemplo, los formularios para la captura de catálogos).
3. Formularios genéricos que sólo muestran información (por ejemplo, los formularios Acerca de e Introducción).

En los casos en los que la cantidad de datos que requiera un formulario sea tan grande que no permita desplegarse en una ventana completa, se utilizará los objetos PAGEFRAME.

INTERFAZ DEL USUARIO

La interfaz debe cumplir los estándares Windows y ser lo más atractiva e intuitiva posible. Deben observarse las condiciones siguientes:

- Resolución en pantalla: 800 x 600 pixeles.
- Fondo de las ventanas: Gris
- Estilo de las ventanas: En su mayoría del tipo Sistema.
- Fuente para los títulos: Arial, Negrita. Tamaño 10.
- Fuente para las etiquetas: Arial, Tamaño 8.
- Fuente para los GETS: Arial, Tamaño 8.
- Estilo de los GETS: Cincelado.

MANEJO DE ERRORES

Se deberán evitar los errores en tiempo de ejecución, si estos ocurren se deberá emitir un mensaje indicando el módulo en donde se registró el error y una descripción del error.

MULTIPLE ACCESO (RED)

Se utilizarán vistas para el acceso compartido de los datos, de esta manera se delegará a Visual FoxPro la responsabilidad de refrescar, bloquear y desbloquear registros.[18]

IV.3.4. Diseño detallado

Toda vez que se tiene un diseño arquitectónico completo (tanto de los datos como de los programas) podemos comenzar nuestro diseño detallado. El diseño detallado sería el equivalente a la especificación de los métodos de cada clase en el DOO. Para un sistema como el SICPRH, esto no es más que la especificación algorítmica de los programas (módulos) que lo conforman, generalmente expresado en términos de pseudocódigo.

"El pseudocódigo... consiste de un conjunto de órdenes. No es ocioso recordar que una orden es un enunciado imperativo y consta de un: ¿Qué se hará? y un ¿Sobre qué se actúa?

El conjunto de órdenes del pseudocódigo están expresadas por frases cortas en español en cada una de las cuales existe uno y sólo un verbo. Una conjunción o una unión [y/o] en la frase son indicativos de que es posible desglosarlas en dos partes. La no definición del objeto o la ambigüedad en su definición, generalmente son producto de una confusión en cuanto a lo que se quiere hacer realmente."[19]

La principal ventaja del uso de pseudocódigo sobre otras herramientas para expresar algoritmos (como diagramas de flujo) consiste en que se pueden omitir detalles. "Debido a que las frases están dadas en español, no se tiene relación con ningún lenguaje. Se es independiente de la implantación, lo cual nos evita pensar en los detalles hasta el momento que sea necesario."[20]

---

[18] Para mayores detalles sobre vistas consúltese *Visual FoxPro Manual del programador*. Microsoft Corporation, 1995 y *Visual FoxPro Guía de aplicaciones profesionales*. Microsoft Co. 1995

[19] Peñaloza Romero, Ernesto. *Fundamentos de Programación*. UNAM Aragón, México, D.F., 1994, p.p.18

[20] Idem.

A continuación presentaremos el pseudocódigo del módulo principal del SICPRH y los correspondientes al subsistema de control de puestos y empleados (SRH00 y SRH10, respectivamente). los demás módulos, aunque fueron desarrollados, no serán mostrados aquí por razones de espacio. Los nombres de los módulos son subrayados.

```
SRH00 (Programa principal)
INICIALIZA variables.
ABRE base de datos.
ABRE ventana.
LEE contraseña.
HAZ MIENTRAS (contraseña inválida y se desea continuar)
      LEE contraseña.
      LEE continuar.
FIN HAZ_MIENTRAS
SI (continuar es verdadero)
      OBTEN tipo_usuario de acuerdo a contraseña válida
      ASIGNA opcion con 1
      HAZ MIENTRAS (opcion sea diferente de 7)
            PINTA menu
                  1.- Control de puestos y empleados.
                  2.- Capacitación y desarrollo.
                  3.- Control de incidencias.
                  4.- Hijos hacendarios.
                  5.- Catálogos.
                  6.- Utilerías.
                  7.- Salir.
            LEE opcion
            HAZ SEGÚN CASO opcion
                  CASO opcion = 1
                        LLAMAR srh10(tipo_usuario)
                  CASO opcion = 2
                        LLAMAR srh20(tipo_usuario)
                  CASO opcion = 3
                        LLAMAR srh30(tipo_usuario)
                  CASO opcion = 4
                        LLAMAR srh40(tipo_usuario)
                  CASO opcion = 5
                        LLAMAR srh50(tipo_usuario)
                  CASO opcion = 6
                        LLAMAR srh60(tipo_usuario)
            FIN HAZ_CASO
      FIN HAZ_MIENTRAS
FIN SI
CIERRA base de datos.
LIBERA variables.
CIERRA ventana.
FIN DE PROGRAMA
```

```
SRH10 (Control de Puestos y empleados)
ABRE ventana.
SI (tipo_usuario es igual a '1' ó a '2')
      ASIGNA opcion con 1
      HAZ MIENTRAS (opcion sea diferente de 5)
            PINTA menu
                  1.- Captura.
                  2.- Kardex.
                  3.- Resumen de movimientos.
                  4.- Consultas.
                  5.- Regresar.
            LEE opcion
            HAZ SEGÚN CASO opcion
                  CASO opcion = 1
                        LLAMAR srh11
                  CASO opcion = 2
                        LLAMAR srh12
                  CASO opcion = 3
                        LLAMAR srh13
                  CASO opcion = 4
                        LLAMAR srh14
            FIN HAZ_CASO
      FIN HAZ_MIENTRAS
SINO
      PON MENSAJE "Acceso Denegado"
FIN SI
CIERRA ventana.
REGRESA


SRH11 (Captura de plazas y empleados)
ABRE ventana.
ASIGNA opcion con 1
LEE folio.
HAZ MIENTRAS (folio es diferente a ESCAPE y opcion es diferente a 3)
      SI (folio es encontrado en tabla plazas)
            POSICIONARSE en registro relacionado de tabla empleados.
            OBTEN variables de campo del registro de plazas.
            OBTEN variables de campo del registro de empleados.
            MUESTRA variables de campo del registro de plazas.
            MUESTRA variables de campo del registro de empleados.
            ASIGNA opcion con 1
            HAZ MIENTRAS (opcion sea diferente de 3)
                  PINTA menu
                        1.- Cambios.
                        2.- Bajas.
                        3.- Regresar.
                  LEE opcion
                  HAZ SEGÚN CASO opcion
                        CASO opcion = 1
                              LLAMAR srh112
                        CASO opcion = 2
                              LLAMAR srh13
                  FIN HAZ_CASO
            FIN HAZ_MIENTRAS
```

```
      SINO
            LLAMAR srh111(folio)
            LEE folio.
      FIN SI
FIN HAZ_MIENTRAS
CIERRA ventana.
REGRESA
```

SRH111 (Altas de plazas y empleados)

```
OBTEN variables de campo en blanco del registro de plazas.
OBTEN variables de campo en blanco del registro de empleados.
LEE variables de campo del registro de plazas (el campo situación sólo
puede contener los valores 'A', 'R' o 'V').
LEE variables de campo del registro de empleados. (el campo situación
sólo puede contener los valores 'A' o 'R').
PINTA menu
      1.- Guardar.
      2.- Cancelar.
LEE opcion
SI (opcion es igual a 1)
      GUARDA variables de campo de registro de la tabla plazas.
      GUARDA variables de campo de registro de la tabla empleados.
      GUARDA variables de campo de registro en tabla kardex.
FIN SI
LIBERA variables.
REGRESA
```

SRH112 (Cambios en datos de plazas y empleados)[21]

```
LEE variables de campo del registro de plazas.
LEE variables de campo del registro de empleados.
PINTA menu
      1.- Guardar cambios.
      2.- Cancelar.
LEE opcion
SI (opcion es igual a 1)
      SOBREESCRIBE variables de campo de registro de la tabla plazas.
      SOBREESCRIBE variables de campo de registro de la tabla empleados.
      GUARDA variables de campo de registro en tabla kardex.
FIN SI
LIBERA variables.
REGRESA
```

[21] Desde este módulo se podrán realizar asignaciones de personal a plazas vacantes, ya sea por nuevos ingresos o por reingresos

```
SRH113 (Bajas de plazas y empleados)
PINTA menu
      1.- Borrar (plaza y empleado)
      2.- Borrar empleado.
      3.- Desasignar empleado por promoción.
      4.- Cancelar baja.
LEE opcion
HAZ SEGÚN CASO opcion
      CASO opcion = 1
            GUARDA variables de campo de registro en tabla kardex.
            ELIMINA variables de campo de registro de la tabla plazas.
            ELIMINA variables de campo de registro de la tabla empleados.
            ELIMINA regs. relacionados con el empleado en otras tablas.
      CASO opcion = 2
            GUARDA variables de campo de registro en tabla kardex.
            BORRA el contenido de rfc del registro de plaza.
            ASIGNA situacion del registro de plaza con 'V'
            ASIGNA nombre con el nombre del ultimo empleado
            ELIMINA registro de la tabla empleados.
            ELIMINA regs. relacionados con el empleado en otras tablas.
      CASO opcion = 3
            GUARDA variables de campo de registro en tabla kardex.
            BORRA el contenido de rfc del registro de plaza.
            ASIGNA situacion del registro de plaza con 'V'
            ASIGNA nombre con el nombre del ultimo empleado.
            BORRA el contenido de ubica del registro de empleados.
FIN HAZ_CASO
LIBERA variables.
REGRESA


SRH121 (Kardex)
ABRE ventana
ASIGNA opcion con 1
HAZ MIENTRAS (opcion es diferente a ESCAPE y opcion es diferente a 3)
      PINTA menu
            1.- Por plaza
            2.- Por persona
            3.- Salir
      LEE opcion
      HAZ SEGÚN CASO opcion
            CASO opcion = 1
                  LLAMAR srh121
            CASO opcion = 2
                  LLAMAR srh122
      FIN HAZ_CASO
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana
REGRESA
```

```
SRH121_(Kardex por plaza)
ABRE ventana.
LEE folio.
HAZ MIENTRAS (folio sea diferente a ESCAPE)
      SI (folio es encontrado en tabla plazas)
            MUESTRA en un cursor todos los registros con ese folio.
            PINTA menu.
                  1.- Altas.
                  2.- Cambios.
                  3.- Bajas.
                  4.- Regresar.
            LEE opcion
            HAZ SEGÚN CASO opcion
                  CASO opcion = 1
                        LLAMAR srh1211
                  CASO opcion = 2
                        LLAMAR srh1212
                  CASO opcion = 3
                        LLAMAR srh1213
            FIN HAZ_CASO
      FIN SI
      LIMPIA ventana
      LEE folio.
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA


SRH1211_(Alta de movimientos de plaza en kardex)
ABRE ventana
OBTEN variables de campo en blanco de un registro de kardex.
LEE variables de campo del registro de kardex.
PINTA menu
      1.- Guardar.
      2.- Cancelar.
LEE opcion
SI (opcion es igual a 1)
      GUARDA variables de campo de registro en la tabla kardex.
FIN SI
LIBERA variables.
CIERRA ventana
REGRESA


SRH1212_(Cambios de datos de plazas en kardex)
ABRE ventana
POSICIONARSE en el registro seleccionado en el cursor.
LEE variables de campo del registro seleccionado en el cursor.
PINTA menu
```

```
      1.- Guardar cambios.
      2.- Cancelar.
LEE opcion
SI (opcion es igual a 1)
      SOBREESCRIBE variables de campo de registro de kardex en tabla.
FIN SI
LIBERA variables.
CIERRA ventana.
REGRESA
```

SRH1213 (Bajas de registros de plazas en kardex)

```
POSICIONARSE en el registro seleccionado en el cursor.
PINTA menu
      1.- Borrar
      2.- Cancelar.
LEE opcion
SI (opcion = 1)
      ELIMINA registro de la tabla kardex.
FIN SI
LIBERA variables.
REGRESA
```

SRH13 (Resumen de movimientos)

```
ABRE ventana.
ASIGNA opcion con 1
HAZ MIENTRAS(opcion es diferente de 5)
      PINTA menu
            1.- Plazas ocupadas.
            2.- Plazas vacantes.
            3.- Plazas en licencia.
            4.- Puesto específico.
            5.- Regresar.
      LEE opcion
      SI (opcion es diferente de 5)
            HAZ SEGÚN CASO opcion
                  CASO opcion = 1
                        LLAMAR srh131
                  CASO opcion = 2
                        LLAMAR srh132
                  CASO opcion = 3
                        LLAMAR srh133
                  CASO opcion = 4
                        LLAMAR srh134
            FIN HAZ_CASO
      FIN SI
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA
```

SRH131 (Resumen de movimientos: plazas ocupadas)
ABRE ventana.
ABRE tabla plazas.
SELECCIONA registros cuyo valor en situación sea diferente a 'V'
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LIBERA cursor.
CIERRA ventana.
REGRESA

SRH132 (Resumen de movimientos: plazas vacantes)
ABRE ventana.
ABRE tabla plazas.
SELECCIONA registros cuyo valor en situación sea igual a 'V'
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LIBERA cursor.
CIERRA ventana.
REGRESA

SRH133 (Resumen de movimientos: plazas en licencia)
ABRE ventana.
ABRE tabla plazas.
SELECCIONA registros cuyo valor en situación sea igual a 'LC' o 'LS' o 'LM'
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LIBERA cursor.
CIERRA ventana.
REGRESA

SRH134 (Resumen de movimientos: puesto específico)
ABRE ventana.
ABRE base de datos.
ASIGNA salir con verdadero
HAZ MIENTRAS salir sea falso
    LEE puesto
    SELECCIONA registros cuyo CP2 + CP3 sea igual a puesto
    ENVIA registros seleccionados a un cursor.
    MUESTRA registros seleccionados.
    LEE salir
FIN HAZ_MIENTRAS
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA

```
SRH14 (Consultas)
ABRE ventana.
ASIGNA opcion con 1
HAZ MIENTRAS(opcion es diferente de 4)
      PINTA menu
            1.- Plantilla.
            2.- Personal.
            3.- FONAC.
            4.- Regresar.
      LEE opcion
      HAZ SEGÚN CASO opcion
            CASO opcion = 1
                  LLAMAR srh141
            CASO opcion = 2
                  LLAMAR srh142
            CASO opcion = 3
                  LLAMAR srh143
      FIN HAZ_CASO
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA
```

```
SRH141 (Plantilla)
ABRE ventana.
ASIGNA opcion con 1
HAZ MIENTRAS(opcion es diferente de 6)
      PINTA menu
            1.- Organizacional completo.
            2.- Presupuestal completo.
            3.- Organizacional con funciones.
            4.- Resumen de plazas.
            5.- Plantilla normalizada.
            6.- Regresar
      LEE opcion
      HAZ SEGÚN CASO opcion
            CASO opcion = 1
                  LLAMAR srh1411
            CASO opcion = 2
                  LLAMAR srh1412
            CASO opcion = 3
                  LLAMAR srh1413
            CASO opcion = 4
                  LLAMAR srh1414
            CASO opcion = 5
                  LLAMAR srh1415
      FIN HAZ_CASO
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA
```

SRH1411 (Plantilla: Organizacional completa)

```
ABRE ventana.
LEE a_ubicacion
ORDENA tabla empleados con respecto al campo ubica
SELECCIONA registros de la tabla empleados cuyo valor de ubica sea igual
a a_ubicación
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

SRH1412 (Plantilla: Presupuestal completa)

```
ABRE ventana.
LEE a_cp1
ORDENA tabla plazas con respecto al campo cp1
SELECCIONA registros de la tabla plazas cuyo valor de cp1 sea igual a
a_cp1
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

SRH1413 (Plantilla: Organizacional con funciones)

```
ABRE ventana.
LEE a_ubicacion
ORDENA tabla empleados con respecto al campo ubica
SELECCIONA registros de la tabla empleados cuyo valor de ubica sea igual
a a_ubicacion y cuyo valor de situacion en tabla plazas sea igual a 'A' o
'R' (activa o en tramite de reubicacion)
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

```
SRH1414 (Plantilla: Resumen de plazas)
ABRE ventana.
ORDENA tabla plazas con respecto al campo CP1, CP2+CP3
AGRUPA tabla plazas con respecto al campo CP1, CP2+CP3
SELECCIONA registros de la tabla plazas cuyo valor en el campo situacion
sea igual a 'A' o 'R'
CONTABILIZA plazas por grupo CP2+CP3
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA


SRH1415 (Plantilla: Plantilla normalizada)
ABRE ventana.
LEE a_cp1
ORDENA tabla plazas con respecto al campo cp1
SELECCIONA registros de la tabla plazas cuyo valor de cp1 sea igual a
a_cp1 y cuyo valor en el campo situacion sea igual a 'A' o 'R'
MUESTRA datos de antigüedad y quinquenios de los registros seleccionados.
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte
      en impresora matricial.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA


SRH142 (Personal)
ABRE ventana.
ASIGNA opcion con 1
HAZ MIENTRAS(opcion es diferente de 2)
PINTA menu
      1.- Directorio de personal.
      2.- Regresar.
LEE opcion
HAZ SEGÚN CASO opcion
      CASO opcion = 1
            LLAMAR srh1421
FIN HAZ_CASO
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA
```

SRH1421 (Personal: Directorio de personal)

```
ABRE ventana.
LEE a_cpl
ORDENA tabla plazas con respecto al campo cpl
SELECCIONA registros de la tabla plazas cuyo valor de cpl sea igual a
a_cpl y cuyo valor en el campo situacion sea igual a 'A' o 'R'
MUESTRA datos personales del empleado (dirección, teléfono, edo. civil)
de los registros seleccionados.
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

SRH143 (FONAC)

```
ABRE ventana.
ASIGNA opcion con 1
HAZ MIENTRAS(opcion es diferente de 3)
PINTA menu
      1.- Listado organizacional.
      2.- Listado presupuestal.
      3.- Regresar.
LEE opcion
HAZ SEGÚN CASO opcion
      CASO opcion = 1
            LLAMAR srh1441
      CASO opcion = 2
            LLAMAR srh1442
FIN HAZ_CASO
FIN HAZ_MIENTRAS
LIBERA variables
CIERRA ventana.
REGRESA
```

SRH1431 (FONAC: Listado organizacional)

```
ABRE ventana.
LEE a_cpl
ORDENA tabla plazas con respecto al campo cpl
SELECCIONA registros de la tabla plazas cuyo valor de cpl sea igual a
a_cpl y cuyo valor en el campo situacion sea igual a 'A' o 'R' y cuyo
valor en el campo FONAC de la tabla empleados sea igual a 'S'
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
```

```
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

SRH1432 (FONAC: Listado presupuestal)

```
ABRE ventana.
LEE a_cp1
ORDENA tabla plazas con respecto al campo cp1
SELECCIONA registros de la tabla plazas cuyo valor de cp1 sea igual a
a_cp1 y cuyo valor en el campo situacion sea igual a 'A' o 'R' y cuyo
valor en el campo FONAC de la tabla empleados sea igual a 'S'
ENVIA registros seleccionados a un cursor.
MUESTRA registros seleccionados.
LEE imprimir.
SI imprimir es verdadero
      IMPRIME registros seleccionados con respecto al formato de reporte.
FIN SI
LIBERA cursor.
LIBERA variables.
CIERRA ventana.
REGRESA
```

Capítulo V

Implementación

V.1 La fase de evolución

Generalmente la fase que le sigue al diseño de un Sistema de Software es la etapa de Implementación o construcción. En la metodología de Booch es un poco más que eso: esta fase significa la evolución de la implementación, desde la codificación de los productos del diseño, pasando por versiones sucesivas hasta llegar al sistema terminado.

"El propósito de esta fase evolucionaria consiste en crecer y cambiar la implementación del sistema cuando sea necesario a través de un refinamiento sucesivo que nos guíe a la producción."[1]

Los principales productos de esta fase de evolución son versiones ejecutables del sistema, mismas que, en una fase temprana del desarrollo del software[2], son turnados al personal de aseguramiento de la calidad para su revisión.

[1] Booch, Grady. , op. cit. p.p. 257

[2] No debemos olvidar que Booch concibe un ciclo de vida iterativo e incremental, por esta razón es posible que en el desarrollo del proyecto del software se pase varias veces por la fase de Evolución.

Conforme se avanza en el desarrollo del proyecto, estas versiones ejecutables son entregadas en forma controlada a ciertos usuarios finales (los usuarios de las versiones alfa y beta). El equipo de desarrollo fija entonces sus expectativas para cada versión del sistema e identifica cada aspecto que se desea que los usuarios evalúen.

Las actividades propias de la fase de Evolución consisten en la aplicación del microproceso y el control de cambios.

El **control de cambios** es un intento por permitir modificaciones inesperadas al trabajo realizado previamente en el análisis o el diseño. Básicamente, los tipos de cambios que podemos esperar durante la evolución de un sistema son:

- Crear una nueva clase o nuevas colaboraciones entre clases.
- Cambiar la implementación de una clase.
- Cambiar la representación de una clase.
- Reorganizar la estructura de clases.
- Cambiar la interface de clases.

Cada tipo de cambio surge por diferentes razones y cada uno tiene su costo.

Por otra parte, las actividades del microproceso para esta fase de Evolución pueden resumirse de la siguiente manera:

IDENTIFICACION DE CLASES Y OBJETOS

Si es necesario se inventan abstracciones de bajo nivel que puedan usarse para la construcción de abstracciones de mayor nivel. Además se intenta descubrir aspectos comunes entre las abstracciones existentes que pueden ser usadas para la simplificación de la arquitectura del sistema.

IDENTIFICACION DE LA SEMANTICA
DE CLASES Y OBJETOS

Se convierten las descripciones de roles y responsabilidades de cada abstracción en protocolos concretos (léase protocolos codificados en el lenguaje de programación).

IDENTIFICACION DE LAS RELACIONES
ENTRE CLASES Y OBJETOS

Se refinan las relaciones entre las abstracciones en relaciones más orientadas a la implementación, por ejemplo, las asociaciones pueden convertirse en relaciones de instanciación o de uso.

Esta fase se ha completado exitosamente cuando la funcionalidad y la calidad de la última versión del software son suficientes para vender el producto.

V.2 Implementación del SICPRH

En el ciclo de vida del SICPRH la penúltima fase es la Implementación, pues, a diferencia de los sistemas basados en la metodología de Booch, hasta ahora no se ha escrito ninguna línea de código que nos permita evolucionar.

Por esta razón, las actividades en esta etapa coinciden con las últimas fases de un típico ciclo de vida de cascada.

V.2.1. Codificación y pruebas de código

Todo los pasos de la Ingeniería del Software que se han presentado hasta ahora van dirigidos hacia un objetivo final: traducir las representaciones del software a una forma que pueda ser comprendida por la computadora. Hemos llegado (finalmente) a la etapa de codificación.

El paso de codificación traduce una representación del software, dada por el diseño detallado, a una realización en un lenguaje de programación. El proceso típico de traducción continúa cuando un compilador acepta el código fuente como entrada y produce como salida un código objeto dependiente de la máquina.

El paso inicial de traducción (codificación) es punto fundamental dentro del contexto de la Ingeniería del Software. En este proceso puede aparecer "ruido" de muchas formas. La interpretación equivocada de las especificaciones del diseño detallado puede conducir a un código fuente erróneo. La complejidad o las restricciones del lenguaje de programación pueden conducir igualmente a un código fuente muy confuso que resulte difícil de probar y mantener.

Anteriormente esta etapa tenía tanto impacto en la construcción de sistemas de software que a menudo prescindía de las actividades del análisis y/o el diseño. Las características de los lenguajes de programación como manejo de memoria, uniformidad, ambigüedad, eficiencia del compilador, portabilidad, etc. no eran del todo satisfactorias, por lo que lo más normal era centrarse en la forma de cómo se solucionarían estos problemas en vez de cuestionarse si las características del programa satisfacían realmente las necesidades del cliente. La codificación era un fin en sí mismo.

En la actualidad la aparición de diversas herramientas para la construcción de software ha invertido esta situación. Las herramientas CASE, los generadores de código, la programación visual, entre otros, ha permitido darle más importancia al análisis del problema y al diseño de la solución, relegando a la codificación a un segundo plano.

El SICPRH será implementado en Visual FoxPro, tomaremos ventaja de la facilidad para crear y mantener bases de datos a través del SMBD, así como de la programación visual. "La programación visual es una forma de CASE que expresa el diseño de programas con gráficos, colores y hasta sonido. Los objetos se representan en forma visual y se pueden considerar como máquinas físicas que pasan de un estado a otro. La programación visual permite a los diseñadores de software introducir, comprender y reflexionar, hacer pruebas y controlar programas en la máquina mediante notaciones pictóricas."[3]

Los módulos que se presentaron en el diseño arquitectónico se convertirán en un tipo de objeto de Visual FoxPro denominados formularios. La llamada entre módulos (formularios) se realizará a través de botones de comando. Las consultas serán manejadas por *queries* y los reportes a impresora llevarán el formato de un *report*.

El comportamiento de cada módulo, expresado en pseudocódigo, no será traducido palabra por palabra (como generalmente se hacía en la construcción de programas en lenguajes de propósito general). Sin embargo, las especificaciones del diseño detallado son suficientes para darle comportamiento a cada formulario puesto que el pseudocódigo permite ocultar los detalles hasta el momento preciso.

Por otra parte, además de la codificación debemos contemplar la verificación del software[4]. Las pruebas de unidad centran el proceso de revisión en la menor unidad de diseño del software: el módulo. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes, con el fin de descubrir errores dentro del

[3] Martin, James., op.cit. p.p.6
[4] Para más información véase Verificación y Validación en el capítulo I.

ámbito del módulo. La complejidad relativa de las pruebas y de los errores descubiertos está limitada por el alcance estricto establecido por la prueba de unidad.

Las actividades que se realizan como parte de las pruebas de unidad de código son:

a) "Se examina la interfaz del módulo para asegurar que la información fluya de forma adecuada hacia y desde la unidad del programa que está siendo probada.
b) Se verifican las estructuras de datos locales para asegurarse que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de ejecución del algoritmo.
c) . Se prueban las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento.
d) Se ejercitan todos los caminos independientes (camino básicos) de la estructura de control con el fin de asegurarse que todas las sentencias del módulo se ejecutan por lo menos una vez.
e) Se prueban todos los caminos de manejo de errores"[5]

V.2.2 Integración y pruebas de aceptación

Una vez que los módulos funcionan bien por separado surge un conflicto mayor: ponerlos a trabajar juntos. La interacción entre módulos puede provocar problemas como que los datos se pierdan en una interfaz, que un módulo pueda tener un efecto adverso e inadvertido sobre otro, y lo que es peor, que las subfunciones, cuando se combinen, puedan no producir la función principal deseada.

[5] Pressman, Roger S., op.cit. p.p.669

La **prueba de integración** es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados en forma aislada y construir una estructura del programa que esté de acuerdo con lo que dicta el diseño.

La integración de los módulos es una actividad incremental. Puede ser ascendente o descendente. La **integración descendente** incluye los módulos moviéndose hacia abajo por la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo principal se van incorporando en la estructura, ya sea de la forma primero en profundidad o de forma primero en anchura.

La **integración ascendente**, como su nombre lo indica, empieza la construcción y la prueba de los módulos atómicos (o sea, módulos de los niveles más bajos de la estructura del programa) para terminar con el módulo de control principal.

Tras la culminación de la prueba de integración, el software está completamente ensamblado como un paquete, se han encontrado y corregido los errores de interfaz y podemos comenzar una serie final de pruebas sobre el software; la **prueba de validación** y la de aceptación.

Como se comentó en el capítulo I, las actividades de verificación y validación se aplican a lo largo de todo el proceso de desarrollo del proyecto, sin embargo, ahora es el momento de la prueba de validación final. La validación puede definirse de muchas formas, la más sencilla indica que la validación se logra cuando el software funciona de acuerdo con las expectativas del cliente establecidas en el documento de la especificación de los requisitos del software.

Aún cuando el sistema ha pasado satisfactoriamente las pruebas de validación, es difícil prever cómo un cliente usará realmente un programa. Se pueden interpretar mal las instrucciones de uso, se pueden usar regularmente combinaciones extrañas de datos y una

salida que puede estar clara para el que realiza la prueba, puede resultar ininteligible para un usuario normal. Es necesario hacer una **prueba de aceptación** del software que nos permita conocer cómo reacciona el sistema en interacción con sus usuarios.

Si el software se desarrolla como un producto que se va a usar por muchas personas, no es práctico realizar pruebas de aceptación formales para cada uno de ellos. La mayoría de los constructores de productos de software llevan a cabo un proceso denominado prueba alfa y beta para descubrir errores que parezca que sólo el usuario final puede descubrir.

Las **pruebas alfa** se llevan a cabo en un entorno controlado, generalmente consiste en conducir al cliente al lugar donde se desarrolla el software para que use el programa de forma natural, con alguna persona del equipo de desarrollo mirando por encima del hombro del usuario y registrando errores y problemas de uso.

La **prueba beta** es una aplicación en vivo del software en un entorno donde no puede ser controlado por el equipo de desarrollo. Los clientes registran todos los problemas (reales o imaginarios) que encuentran durante la prueba e informan a intervalos regulares. Como resultado de los problemas anotados durante la prueba beta, el equipo de desarrollo lleva a cabo modificaciones y así prepara una versión final del producto.

Conclusiones

El objetivo general de este trabajo de tesis consistió en solucionar un problema real por medio de un sistema de información.

La necesidad de tener un control adecuado de plazas y recursos humanos en la Subsecretaría de Hacienda y Crédito Público es una preocupación constante de la Dirección de Técnica Operativa, no obstante que los procesos manuales, aunados a un sistema de software obsoleto y mal diseñado, proporcionaban información tardía, confusa y sobre todo, poco confiable.

Se requería un replanteamiento total de los sistemas que se habían diseñado para la Subdirección de Recursos Humanos, pero no existía voluntad en esta instancia para hacerlo. Afortunadamente, los buenos resultados que se obtuvieron en los últimos sistemas de software que el Departamento de Informática implementó para otras Subdirecciones, en combinación con otros factores, hizo cambiar esta situación y se pudo aprobar la construcción del nuevo sistema.

La diversidad de actividades en esa Subdirección, a primera vista, hacía pensar que el sistema por implementar tendría una complejidad considerable. Aquí es donde entró en acción la Ingeniería del Software.

El establecimiento de un ciclo de vida híbrido (en parte Orientado a Objetos y en parte estructurado) fue un gran acierto. La recurrente referencia a la teoría del Análisis y Diseño OO tenía un fin muy preciso: demostrar que ambas actividades deben ser pensadas en forma independiente de los lenguajes de programación; es lamentable que la mayor parte de lo que se ha escrito en torno a las técnicas orientadas a objetos se basen precisamente en la programación OO. Se necesita un punto de vista más fundamental.

La construcción del SICPRH comenzó con la fase de Concepto, en esta etapa se recogieron y evaluaron los procedimientos, operaciones, funciones y necesidades de la Subdirección para consignarse en el documento del Concepto de Operaciones (ConOps).

El análisis orientado a objetos fue de mucha utilidad, primero porque fue relativamente fácil para mí y para mis clientes establecer lo que pertenecía al dominio del sistema y lo que estaba fuera de él, incluso nos dimos cuenta de que el sistema no era tan complicado como parecía al principio. Además, el modelo del sistema se realizó con diagramas que expresaban el vocabulario de la aplicación y esto permitió que los futuros usuarios validaran dicho modelo.

El diseño fue parecido; dicen que la Ingeniería más exitosa es la Ingeniería más sencilla, en esta fase tomamos los escenarios producto del análisis y observamos que se facilitaría notablemente su manejo si los integrábamos a todos en un campo y generábamos un catálogo de movimientos (este mecanismo simplificó aún más el sistema). Asimismo, la estructura de clases y sus relaciones producto del análisis, se convirtió sin mayores dificultades en la base de datos del sistema.

Finalmente se implementó el software con los productos del diseño.

Como pudo observarse, no existe ningún rompimiento brusco entre las fases del ciclo de vida del SICPRH. Todo lo contrario. resultó ventajoso realizar un análisis orientado a objetos e implementar el sistema en un lenguaje de programación visual y bajo la acción de un SMBDR.

Escoger alguna metodología no implica hacer de lado las técnicas de la Ingeniería del Software: existen actividades de esta Ingeniería que no deben dejar de hacerse, independientemente del paradigma o metodología que se use, tal es el caso de la Administración de Proyectos y del aseguramiento de la Calidad, por ejemplo.

La clave del éxito en el desarrollo de un proyecto de programación consiste en la aplicación inteligente de las herramientas de la Ingeniería de Software. Sería erróneo decir, por ejemplo, que la Orientación a Objetos es la panacea en la solución de sistemas de información.

Resulta más correcto decir que las técnicas para el análisis y el diseño orientado a objetos y la teoría de la Ingeniería del Software conjuntan una herramienta excepcional para la construcción de sistemas de información y es decisión del equipo de desarrollo la elección del camino que más se adecue a su situación.

Bibliografía

Libros

- *Booch, Grady.* ***Object-Oriented Analisys and Design with Applications.*** *2ª edición. Benjamin Cummings. Publishing Co. Redwood City, California, 1994*

- *Fairley, Richard.* ***Ingeniería de Software.*** *1ª edición. McGraw Hill., México, D.F., 1992*

- *Graham, Ian.* ***Object Oriented Methods.*** *2ª edición. Addison-Wesley. Great Britain, 1992.*

- *IEEE.* ***Software Engineering Standards Collection.*** *Institute of Electrical and Electronics Engineers, Inc. Los Alamitos, California, Computer Society Press, 1993*

- *Jacobson, Ivar et. al.,* ***Object Oriented Software Engineering.*** *4ª edición. Addison Wesley 1994.*

- *Martin, James & Odell, James J.* ***Análisis y Diseño Orientado a Objetos.*** *1ª edición. Prentice Hall. México, D.F., 1994*

- *Peñaloza Romero, Ernesto.* ***Fundamentos de Programación.*** *UNAM, Aragón.*

- *Pressman, Roger S.* ***Ingeniería del Software. Un enfoque práctico.*** *3ª edición. Mc Graw Hill.*

- *Thayer, Richard & Dorfman, Merlin.* ***Standards, Guidelines and Examples on System Software Requirement Engineering.*** *IEEE. Los Alamitos, California. Computer Society Press.*

- *Thayer, Richard & Dorfman, Merlin.* ***Software Engineering.*** *IEEE Computer Society Press, 1997.*

Manuales

- ***Condiciones Generales de Trabajo de la Secretaría de Hacienda y Crédito Público.***

- ***Diario Oficial de la Federación.***

- ***Manual de Procedimientos Administrativos de Recursos Humanos.*** *Tomo II. Dirección General de Personal de la Oficialía Mayor de Hacienda. 1995*

- ***Manual de Organización Específico de la Dirección de Técnica Operativa de la Subsecretaría de Hacienda y Crédito Público.*** *1997*

Otras fuentes

- ***Overview of the Software Engineering Standards Committe of the IEEE Computer Society.*** *Versión 3.2, September 15th, 1997*