CIENCIA Y SOCIEDAD
Volumen XXV, Número 1
Enero - Marzo 2000

OMT-Z: UNA METODOLOGÍA HÍBRIDA PARA ESPECIFICAR SISTEMAS DE INFORMACIÓN

Ramón Díaz*

RESUMEN

Las metodologías de análisis y diseño de sistemas computarizados informales (como la Tecnica de Modelación de Objetos - Objetct Modeling Tecnique), adolecen en la mayor parte, de falta de precisión. Esta debilidad se debe a que los diagramas y lenguajes naturales utilizados por estas metodologías para describir un sistema, no tienen bases matemáticas, por lo que las especificaciones no pueden ser validadas antes de construir el producto.

Cuando utilizamos una metodología informal en el desarrollo de un sistema de información corregir problemas de análisis y diseño en la fase de implementación tiene un costo muy alto y conduce a la larga a problemas de aseguramiento de la calidad.

Metodologías como OMT, poseen herramientas gráficas para describir los componentes de software y pasos bien estructurados para guiar el desarrollo. Su desventaja es que no están sustentadas matemáticamente, por lo que pueden ocurrir ambiguedades en el diseño, que no son detectadas hasta en la face de implementación, (y en el peor de los casos son detectactas en la etapa de mantenimiento).

Por otra parte, los métodos formales, como es el caso de Z, poseen una base matemática fuerte que nos permite verificar que el diseño generado es correcto. Un método de especificación formal asegura que nuestro diseño cumpla con una serie de propiedades matemáticas que son fácilmente verificables (sin implementar en un computador la especificación). Además, los métodos formales proveen a los analistas, diseñadores, usuarios, etc. de un lenguaje común que no se presta a ambiguedades.

(*) **Departamento de Maestría en Computación, Instituto Tecnologico de Costa Rica.**

Una de las desventajas de los métodos formales es: que los usuarios e ingenieros de software no están relacionados con la base matemática que los soporta. Por otra parte, al especificar utilizando métodos formales debemos llegar a un nivel de detalle que rios obliga a pensar más en las bases formales del sistema, haciendo del diseño una tarea más compleja.

Por las razones anteriores estamos planteando una metodología que utiliza un método formal (Z) y otro informal (OMT) para analizar y diseñar sistemas computarizados. Esta metodología utiliza OMT para describir el sistema por medio de objetos y relaciones entre estos, se utilizan las herramientas gráficas que provee OMT, y Z, para formalizar las especificaciones generadas por OMT.

OMT-Z es una metodología que provee al analista de una notación gráfica, con la cual exponer el sistema a usuarios nó expertos y un soporte formal para comprobar la válidez de la especificación.

En el resto de este articulo presentamos parte de la metodología híbrida OMT-Z. Por razones de espacio solo se presenta la formalización del modelo de objetos.

PALABRAS CLAVES: Sistemas computarizados, modelación, metodología OMT.

1. CONSTRUCCIÓN DE UN MÉTODO DE DISEÑO COMBINANDO Z CON OMT (OMT-Z)

OMT-Z es una metodología de análisis y diseño de sistemas, que combina los métodos OMT y Z para especificar sistemas computarizados. OMT-Z utiliza la notación gráfica de OMT como son: el Modelo de Objeto, Modelo Dinámico y Modelo Funcional para discribir las diferentes fases de un sistema. Estos modelos pertenecen a las notaciones que se conocen como informales, (debido a que no tienen una base matemática que las sustente). Los principales problemas de las notaciones informales son :

Incoherencias: La interpretación de la computación en estas metodologías es bastante subjetiva. La mayor parte de estas utilizan lenguaje natural para describir los procesos, por lo que las especificaciones pueden estar inconclusas, pueden contradecirse con la descripción de otros procesos o pueden interpretarse equivocadamente.

Incompletitud: Al especificar los procesos y estado del sistema no se describe claramente cuál es el estado del sistema, luego de ejecutar un proceso o los estados válidos del sistema en su vida útil.

2. Modelos de OMT agregando Z

A continuación se describen los cambios a realizar a cada modelo de OMT:

Modelo de Objeto: Para cada clase de objeto se agrega un esquema que describa sus atributos y los invariantes de estos atributos.

Modelo Dinámico: Para cada evento y/o acción especificar un esquema que describa el proceso realizado y el estado en que queda el objeto.

Modelo Funcional: Para cada proceso del diagrama especificar un esquema que describa el proceso realizado y el estado de las entidades involucradas resultantes.

En este documento solo se definen las técnicas utlizadas para formalizar el diagrama de objeto.

2.1. Modelo de Objetos en OMT Z

En OMT-Z se especifican formalmente las limitaciones e invariantes que debe mantener cada objeto como entidad individual y .el sistema como un todo. A continuación se presentan los pasos a seguir para formalizar el diagrama de objetos de OMT:

1. Construir esquemas genéricos que especifiquen para cada clase de objetos el conjunto que lo conforma, la relación de integridad entre los atributos del objeto y su identificación y los predicados que restringen la relación entre la clase de objeto y sus instancias.
2. Contruir esquemas genéricos para cada tipo de relación entre los objetos a modelar.

3. Para cada clase de objeto construir un esquema que defina sus atributos. Cada tributo debe ser definido previamente utilizando los tipos de datos válidos en Z.
4. Para cada clase de objeto instanciar el esquema genérico que define objetos. Los parámetros de la instanciación son ObjectoID (Identificación del Objeto) y Atributos Objecto (Atributos del objeto).
5. Instanciar los esquemas genéricos de relación con cada una de las relaciones entre objetos. Los párametros de la instanciación son: Objeto MaestroID (Identificación del objeto maestro) y ObejtoDetalleID (Identificación de los objetos detalles).
6. Los nombres tanto de las clases de objeto como las asociaciones, se podrán escribir con mayúscula. El nombre del registro será en singular y el nombre del conjunto de instancias en plural.

3. Caso de estudio

3.1. Descripción del Sistema

Es un sistema de seguridad controlado a través de una Alarma, una Central de Alarmas, una Línea Telefónica y un Computador.

La alarma se encuentra instalada donde el cliente, con sus respectivos dispositivos de activación (cables, bocinas, etc.). La central de alarmas se encuentra en las instalaciones de la compañía que presta el servicio de seguridad, y a la cual estan conectadas las alarmas de los clientes, (por medio de líneas telefónicas). A su vez la central esta conectada a un computador. Este computador monitorea los eventos recibidos por la central.

El objetivo del sistema es controlar los eventos ocurridos en cada instalación, dando un servicio eficiente de aviso a: autoridades militares y a clientes en caso de emergencia.

Como objetivo secundario: la empresa logra aumentar sus ingresos cobrando una tarifa por llamadas hechas a sus abonados (clientes).

A continuación describimos el funcionamiento del sistema:

El cliente mantiene la alarma en estado de encendido o apagado. Si la alarma esta en estado de encendido y se interrumpe el alambrado de seguridad se activa el dispositivo de sonado y este envía una señal de inicio de ruido a la Sirena (bocina). Si la alarma esta sonando y un usuario preciona el botón de apagado, ésta pregunta la clave de autorización, si el usuario esta autorizado, la alarma pasa al estado de apagado. Una vez que se resuelve el daño en el alambrado de seguridad el usuario puede encenderla.

La Línea Telefónica: Sirve como conección de la alarma y la central de alarmas. Si la alarma intenta comunicarse con la central de alarmas, y la línea telefónica esta ocupada, automáticamente ésta sigue intentado, hasta que la comunicación se logre.

La Central de Alarma: Actúa como un codificador y decodificador de los mensajes. Su función es " recibir " mensajes de la alarma y decodificarlo en una notación entendible por el computador. De manera similar, recibe mensajes del computador y los codifica en una notación entendida por la alarma.

El computador: Recibe mensajes de la central de alarma y las despliega en la pantallas del computador de acuerdo a la prioridad de la señal. Otras de sus funciones es calcular el cargo por llamadas a clientes y mantener la información de estas.

Las figuras 1 y 2 muestran el digrama de objetos del sistema y un gráfico explicativo respectivamente.

3.2. Identificación de Objetos y Clases

Se identifican las siguientes clases potenciales:

1. Alarma
2. Central de Alarmas
3. Cliente
4. Computador
5. Línea Telefónica
6. Eventos
7. Tipos de Señales
8. Operador
9. Factura de Cargo
10. Mensajes

3.3. Diccionario de Datos

Alarma:	Dispositivo instalado donde el cliente, esta controla el cableado de seguridad y emite señales adecuadas a la central de alarma.
Central de Alarma:	Dispositivo instalado en la empresa proveedora del sevicio de seguridad, que recibe las señales desde las alarmas de los clientes y las codifica en una notación entendida por el computador.
Cliente:	Persona a quien se le brinda el servicio de seguridad.
Computador:	Dispositivo que monitorea las señales recibidas y calcula el cobro de llamadas a clientes.
Línea Telefónica:	Dispositivo por medio del cual se comunican las alarmas y la central de alarmas.

Eventos:	Mensajes emitidos por las alarmas, o la central de alarma indicando la ocurrencia de un evento específico.
Operador:	Persona que recibe los mensajes desplegados por el computador y ejecuta acciones correctivas.
Factura de Cargo:	Documento enviado al cliente para el cobro de aranceles por llamadas de aviso.
Mensaje:	Señales emitidas por la línea telefónica.

4.4. Identificar Asociaciones

Una alarma tiene un cliente

Un cliente tiene muchas alarmas

Una alarma tiene una línea telefónica

Una línea telefónica tiene una alarma

Una central de alarmas tiene muchas alarmas

Una alarma tiene una central de alarmas

El computador tiene una central de alarmas

La central de alarmas tiene un computador

Un tipo de señal tiene muchos eventos

Un evento tiene un tipo de señal

Un cliente tiene muchas llamadas de aviso

Una llamada de aviso tiene un cliente

Una alarma tiene muchas señales

Una señal es de una alarma

3.5. Diagrama de Objetos

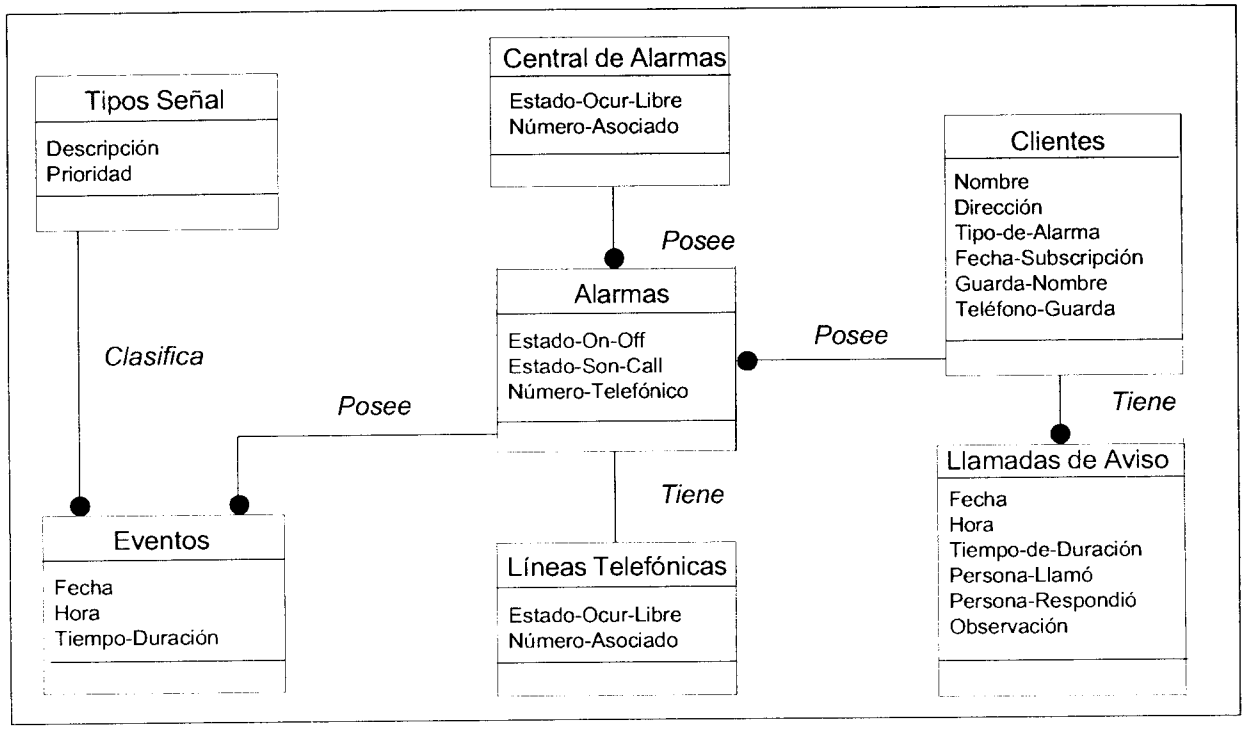


Figura 1. Diagrama de objetos sistema de control de alarma

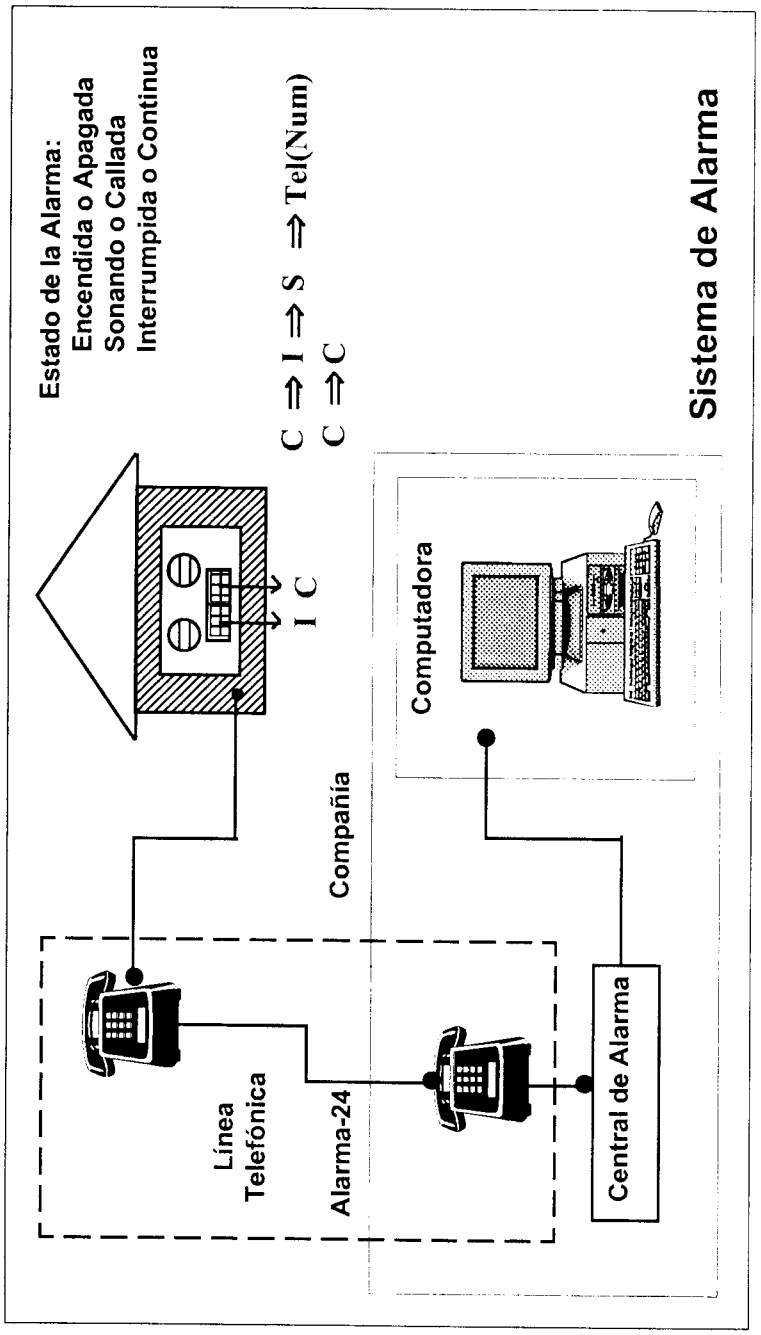


Figura 2. Diagrama del sistema de alarma computarizado

4.1. Definición de Tipos de Atributos

A continuación se define el tipo para los atributos que aparecen en cada clase de objeto dentro del digrama de objetos.

Definición de Atributos

ESTADO-ON-OFF :: = On | Off
ESTADO-SON-CALL :: = Son | Call
ESTADO-OCUR-LIBRE :: = Ocur | Libre
CLAVE = = seq_8 CHAR
NUMERO-TELEFONO = = seq_8 CHAR
DESCRIPCION = = seq_1 CHAR
NOMBRE = = seq_1 CHAR
TIPO-DE-ALARMA = = seq_1 CHAR
FECHA = = seq_6 CHAR
HORA = = seq_6 CHAR
OBSERVACION = = seq_0 CHAR
[Alarmald, Clienteld, Linea-Telefonicald, Eventold, Tipo-de-Eventold, Llamada-de-Avisold, Centralld]

4.2. Definición de Atributos de Objetos y Clases de Objetos

A continuación se define cada clase de objeto. Primero se definen sus atributos en un esquema separado y luego otro esquema con la clase en sí misma.

Clase de Objeto Alarma

AtributoAlarma

Clave: CLAVE
Estado-On-Off: ESTADO-ON-OFF
Estado-Son-Call: ESTADO-SON-CALL
Numero-Telefonico: NUMERO-TELEFONO

Este esquema define los atributos que posee todo objeto de la clase alarma.

ObjetosClaseAlarma

Objeto[AlarmaId, AtributoAlarma]

[conocidosClaseAlarma / conocidosClase,

objetosdeClaseAlarma / objetosdeClase]

Este esquema instancia el esquema genérico "Objeto" con los párametros AlarmaId y AtributoAlarma. Se renombra el conjunto conocidosClase por conocidosClaseAlarma y la función objetosdeClase con el nombre objetosdeClaseAlarma. Esto retorna la entidad ObjetosClaseAlarma heredando los predicados del esquema genérico "Objeto".

Si expandimos el esquema se vería como sigue:

ObjetosClaseAlarma

conocidosClaseAlarma: **F** AlarmaId

objetosdeClaseAlarma: AlarmaId → AtributoAlarma

---

conocidosClaseAlarma = dom objetosdeClaseAlarma

En lo que resta de este artículo, por razones de espacio, se omite la explosión y documentación de esquemas parametrizados para creación de clases de objetos.

Clase de Objeto Cliente

AtributoCliente

Nombre: NOMBRE
Direccion: DESCRIPCION
Tipo-de-Alarma: TIPO-DE-ALARMA
Fecha-Subcripcion: FECHA
Guarda-Nombre: NOMBRE
Telefono-Guarda: NUMERO-TELEFONO
Negocio-Residencia: DESCRIPCION
Encargado: NOMBRE
Clave: CLAVE

ObjetosClaseCliente

Objeto[ClienteId, AtributoCliente]
[conocidosClaseCliente / conocidosClase,
objetosdeClaseCliente / objetosdeClase]

Clase de Linea Telefonica

AtributoLineaTelefonica

Estado Ocur Libre: ESTADO-OCUR-LIBRE
Numero-Asociado: NUMERO-TELEFONO

ObjetosClaseLineaTelefonica

Objeto[Linea-TelefonicaId, AtributoLineaTelefonica]
[conocidosClaseLineaTelefonica / conocidosClase,
objetosdeClaseLineaTelefonica / objetosdeClase]

Clase de Central de Alarmas

AtributoCentralAlarma

Estado_Occur_Libre: ESTADO-OCUR-LIBRE
Numero-Asociado: NUMERO-TELEFONO

ObjetosClaseCentralAlarma

Objeto[CentralId, AtributoCentralAlarma]
[conocidosClaseCentralAlarma / conocidosClase,
objetosdeClaseCentralAlarma / objetosdeClase]

Clase de Eventos

AtributoEvento

Fecha: FECHA
Hora: HORA
Tiempo-Duracion: **N**

ObjetosClaseEvento

Objeto[EventoId, AtributoEvento]
[conocidosClaseEvento / conocidosClase,
objetosdeClaseEvento / objetosdeClase]

Clase de Tipo de Eventos

AtributoTipodeEvento

Descripcion: DESCRIPCION
Prioridad: **N**

ObjetosClaseTipoEvento

Objeto[Tipo-de-EventcId, AtributoTipodeEvento]
[conocidosClaseTipoEvento / conocidosClase,
objetosdeClaseTipoEvento / objetosdeClase]

Clase de Llamadas de Aviso

AtributoLlamadaAviso

Fecha: FECHA
Hora: HORA
Tiempo-de-Duracion: **N**
Persona-Llamo: NOMBRE
Persona-Respondio: NOMBRE
Observacion: OBSERVACION

ObjetosLlamadaAviso

Objeto[Llamada-de-AvisoId, AtributoLlamadaAviso]
[conocidosClaseLlamadaAviso / conocidosClase,
objetosdeClaseLlamadaAviso / objetosdeClase]

4.3. Definición de Relaciones entre objetos

Una vez hemos definido los esquemas de las clases de objetos y sus atributos, podemos especificar los esquemas de las relaciones entre las clases de objetos.

RelacionClienteAlarmas

RelacionUnoaMasUno[ClienteId, AlarmaId]
[conocidosClaseCliente / conocidosObjetoMaestro,
conocidosClaseAlarma / conocidosObjetoDetalle,
RelacionClienteAlarmas / RelacionMaestroDetalle]

Este esquema instancia uno de los esquemas genéricos de relación (ver anexo), primitivos en OMT-Z. El esquema genérico de "relación uno a más uno" es instanciado para formar la relación de Clientes a Alarmas, (ver diagrama de objeto). La instanciación renombra los conjuntos del esquema genérico *conocidosOjetoDetalle* y *conocidosObjetoDetalle* por *conocidosClaseCliente* y *conocidosClaseAlarma* respectivamente.

Si expandimos el esquema anterior se vería como sigue:

RelacionClienteAlarmas

conocidosClaseCliente: **F** ClienteId
conocidosClaseAlarma: **F** AlarmaId
RelacionClienteAlarmas: ClienteId $\leftrightarrow$ AlarmaId

---

dom RelacionClienteAlarmas = conocidosClaseCliente
ran RelacionClienteAlarmas $\subseteq$ conocidosClaseAlarma
$\forall$e: conocidosClaseCliente
· # {RelacionClienteAlarmas < {e} > } $\geq$ 1
RelacionClienteAlarmas~ $\in$ AlarmaId $\rightarrow$ ClienteId

Las primeras dos líneas del esquema definen los conjuntos de clientes y alarmas conocidos. Luego se define la relación "RelaciónClienteAlarma", que asocia cliente a sus alarmas. Luego se restringe la relación con los siguientes predicados :

1. El dominio de la relación cliente-alarma es igual a los clientes conocidos (es decir, todo cliente conocido por el sistema debe tener alarma).

2. El rango de la relación es un subconjunto de las.alarmas conocidas por el sistema.
3. Para todo cliente conocido por el sistema el número de alarmas registradas debe ser por lo menos una, es decir todo cliente debe tener al menos una alarma.
4. La relación invertida resulta en una función total de Alarmas a Clientes. Este último predicado restringe el segundo al decir que toda alarma debe tener un cliente asociado, en otras palabras todo el conjunto alarma entra en la relación.

Las siguientes relaciones no serán expandidas por razones de espacio.

Relación entre la Alarma y Línea Telefónica

RelacionAlarmaLineaTelefonica

RelacionUnoaUno[Alarmald, Linea-Telefonicald]
[conocidosClaseAlarma / conocidosObjetoMaestro,
conocidosClaseLineaTelefonica / conocidosObjetoDetalle,
RelacionAlarmaLineaTelefonica / RelacionMaestroDetalle]

Relación entre la Central de Alarmas y las Alarmas

RelacionCentralaAlarmas

RelacionUnoaMasUno[Centralld, Alarmald]
[conocidosClaseCentralAlarma / conocidosObjetoMaestro,
conocidosClaseAlarmay conocidosObjetoDetalle,
RelacionCentralaAlarmas / RelacionMaestroDetalle]

Relación entre Cliente y Eventos

RelacionClienteEventos

RelacionUnoaMuchos[Clienteld, Eventold]
[conocidosClaseCliente / conocidosObjetoMaestro,
conocidosClaseEvento / conocidosObjetoDetalle,
RelacionClienteEventos / RelacionMaestroDetalle]

Relación entre Tipo de Eventos y Eventos

RelacionTipodeEventoaEventos

RelacionUnoaMuchos[Tipo-de-EventoId, EventoId]
[conocidosClaseTipoEvento / conocidosObjetoMaestro,
conocidosClaseEvento / conocidosObjetoDetalle,
RelacionTipodeEventoaEventos / RelacionMaestroDetalle]

Nota: La relación entre el computador y la central de alarma no es importante en el dominio de la aplicación, por lo que no está siendo modelada.

Relación entre Central de Alarmas y Computador

RelacionCentraldeAlarmaComputador

RelacionUnoaUno[CentralId, ComputadorId]
[conocidosClaseCliente / conocidosObjetoMaestro,
conocidosClaseLlamadaAviso / conocidosObjetoDetalle,
RelacionClienteaLlamadasdeAvisos /
RelacionMaestroDetalle]

5. Diagrama Dinámico

5.1. Clase de Alarma

Escenarios:

1. El usuario enciende la alarma.
2. En este momento se verifica si hay interrupción en la línea de seguridad:
   - Si está interrumpida se dispara el dispositivo de activación de la sirena y la alarma pasa a estado sonando e interrumpida.
   - Si no, la alarma pasa a estado encendida, continua y callada.

3. Si la alarma esta sonando e interrumpida el usuario puede apagarla hasta resolver el daño, a la acción de apagado la alarma responde solicitando la clave de autorización al usuario:

   Si la clave es válida se pasa al estado interrumpida, sonando, apagada (no genera sonido).

   Si la clave no es válida, la alarma no responde nada.

4. Si la alarma está encendida (en estado continúa-callada) y se interrumpe la red de seguridad, se pasa a estado interrumpida- sonando. Se genera un mensaje a la sirena para que empiece a sonar.

5. Para cada evento (apagar, interrumpir, sonar, encender, etc.) se genera una llamada a la central de alarmas con un código que identifica el evento. La llamada se hace al número registrado en la alarma, donde pueden ocurrir dos acciones:

   - Número ocupado (o fuera de servicio), la alarma sigue intentando llamada.
   - Número desocupado, se genera la llamada de aviso a la central de alarma.

6. Apagar la alarma,si esta sonando ver paso (3), sino apagarla. Apagar la alarma es equivalente a pasar el interruptor de encendido a apagado.

5.2. Línea Telefónica

- *Escenario*:

  1. La alarma solicita tono de marcado a la línea telefónica.
  2. La línea telefónica responde con tono disponible.
  3. La alarma marca el número telefónico a llamar.
  4. El número marcado comienza a timbrar (en la Central de alarmas).

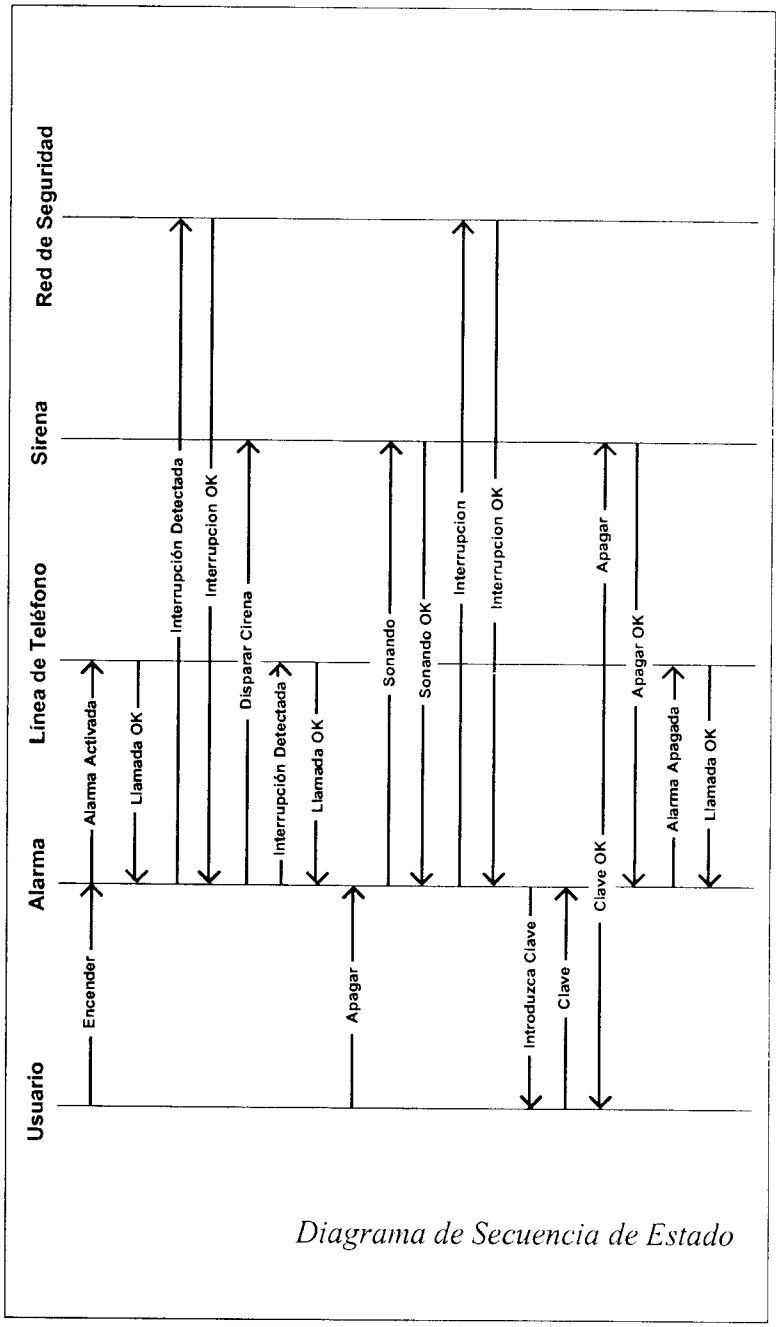


Diagrama de Secuencia de Estado

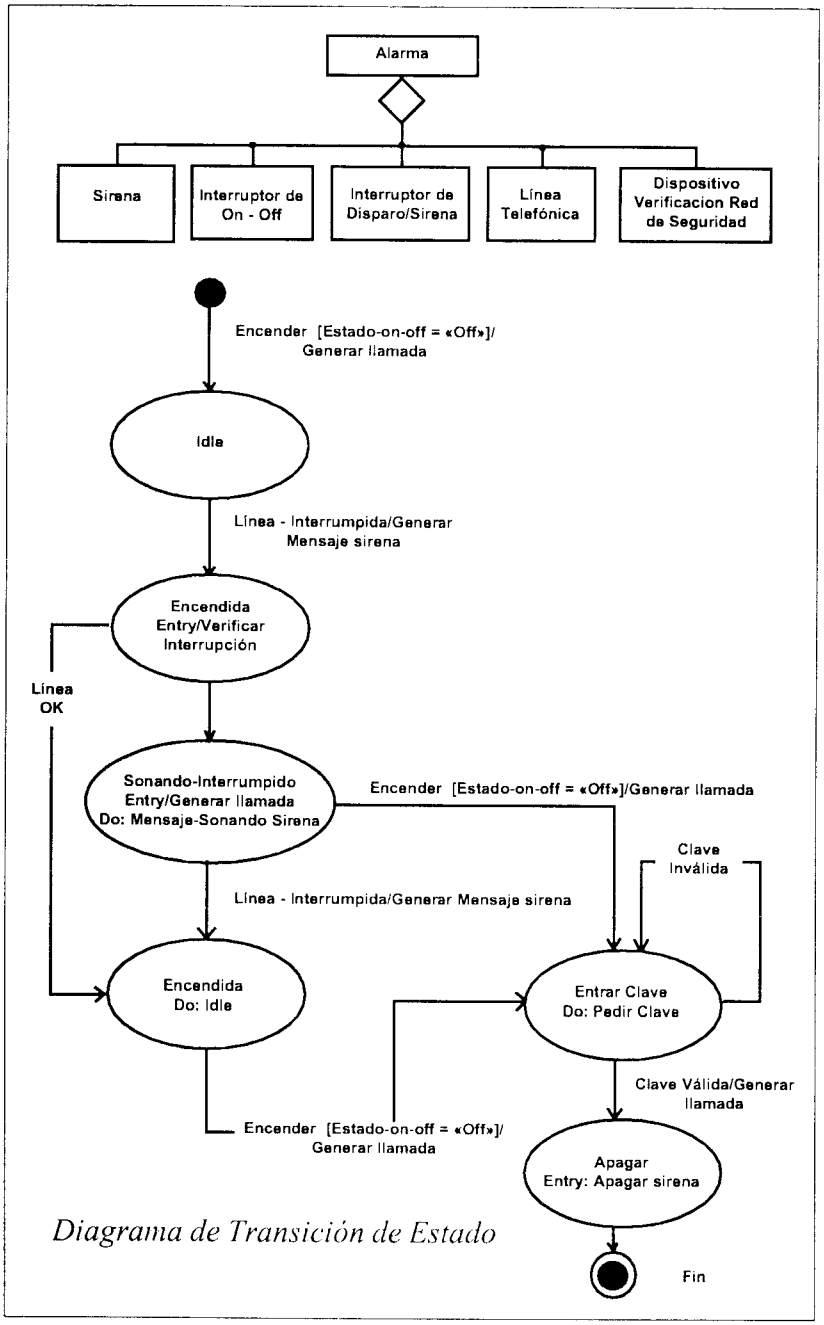


Diagrama de Transición de Estado

5. La señal de timbrado aparece en la alarma .
6. La central de alarma responde.
7. La señal de timbrado cesa en el número de llamada (Central de alarma).
8. La señal de timbrado cesa en la alarma.
9. Las líneas se conectan (Envío de Mensaje).
10. La alarma corta la comunicación.
11. Se desconectan los teléfonos.
12. La central de alarmas libera la línea telefónica.

Diagrama de Secuencias de Eventos

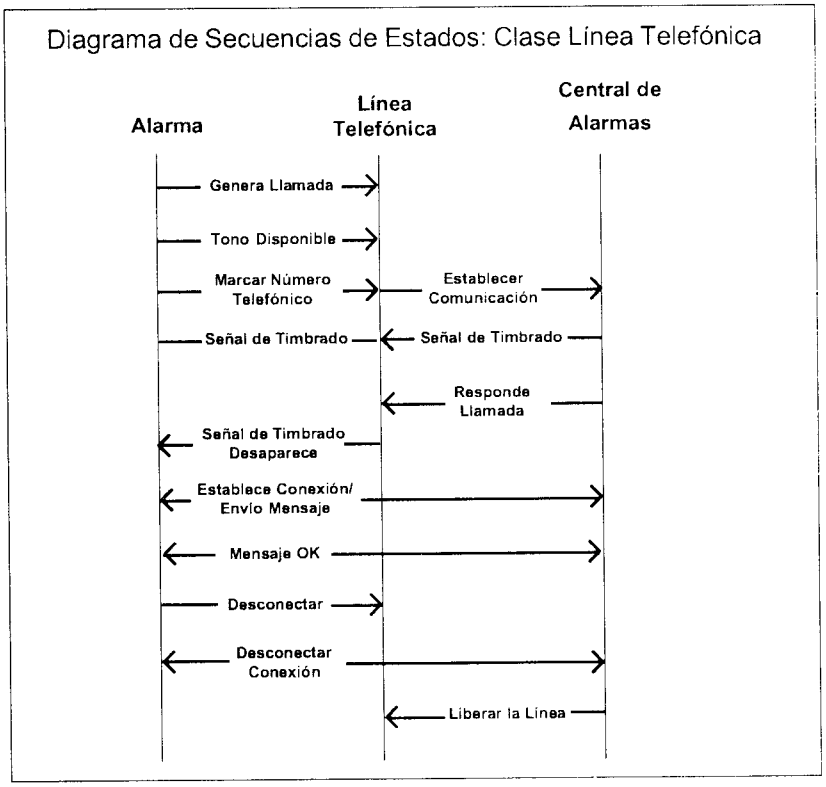
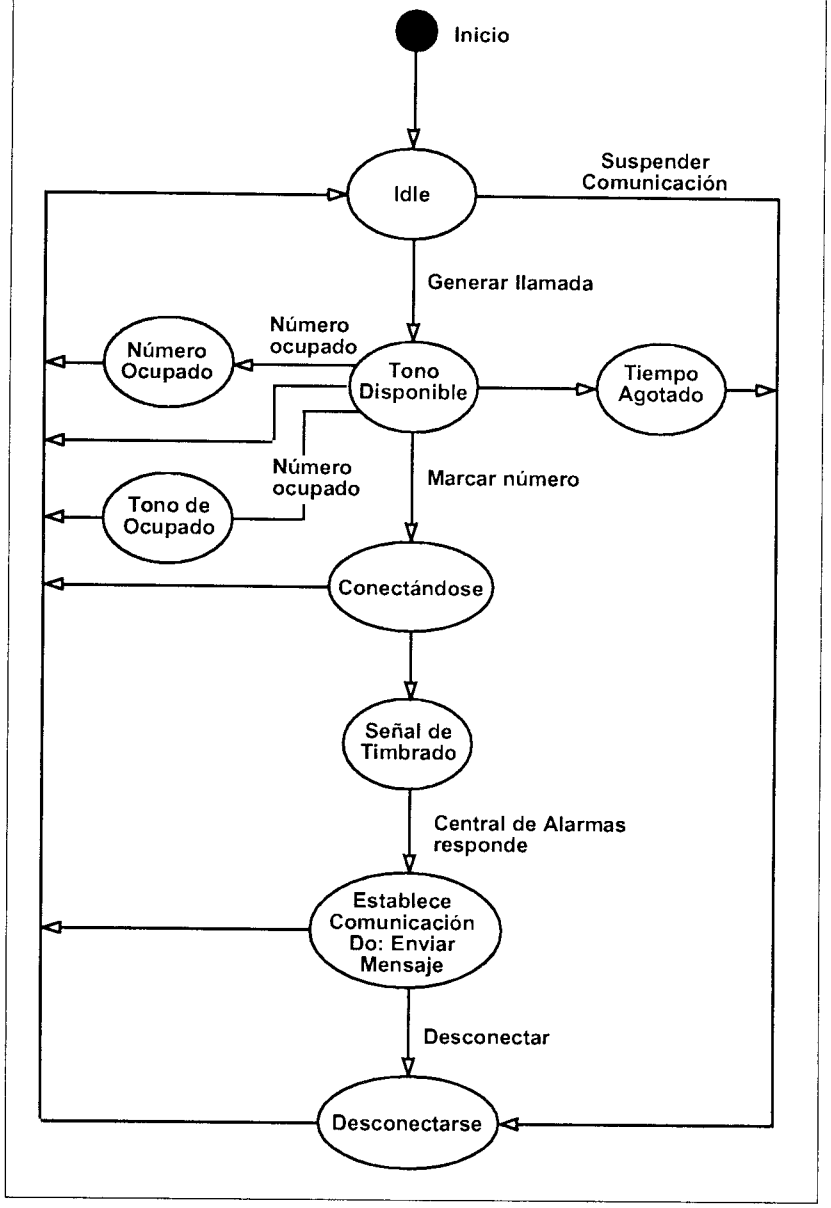


Diagrama de transición de Estado



5.3. Central de Alarmas

Escenarios:

1. Recibe un evento desde la alarma.
2. Codifica el evento en un lenguaje establecido (entendido por el computador).
3. Envía el mensaje al computador.

Diagrama de Secuencia de Eventos

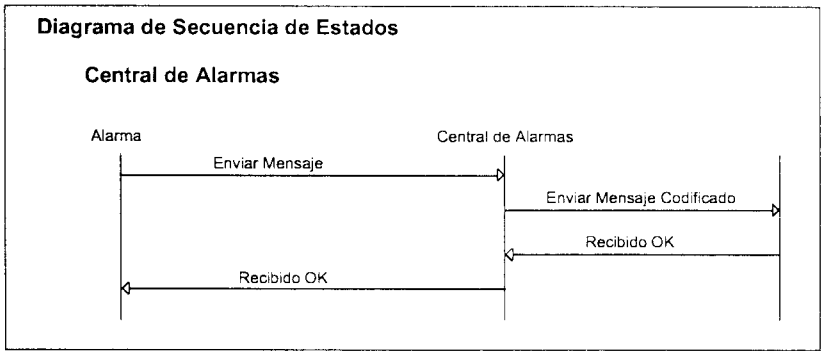
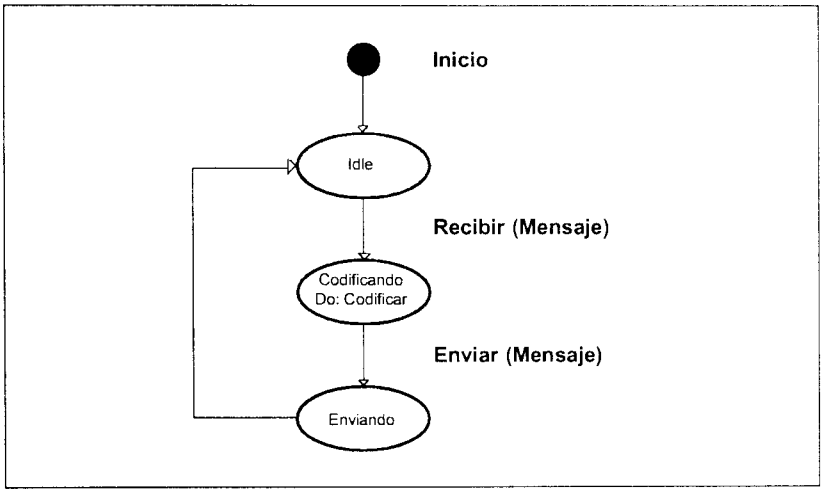


Diagrama de transición de Estado



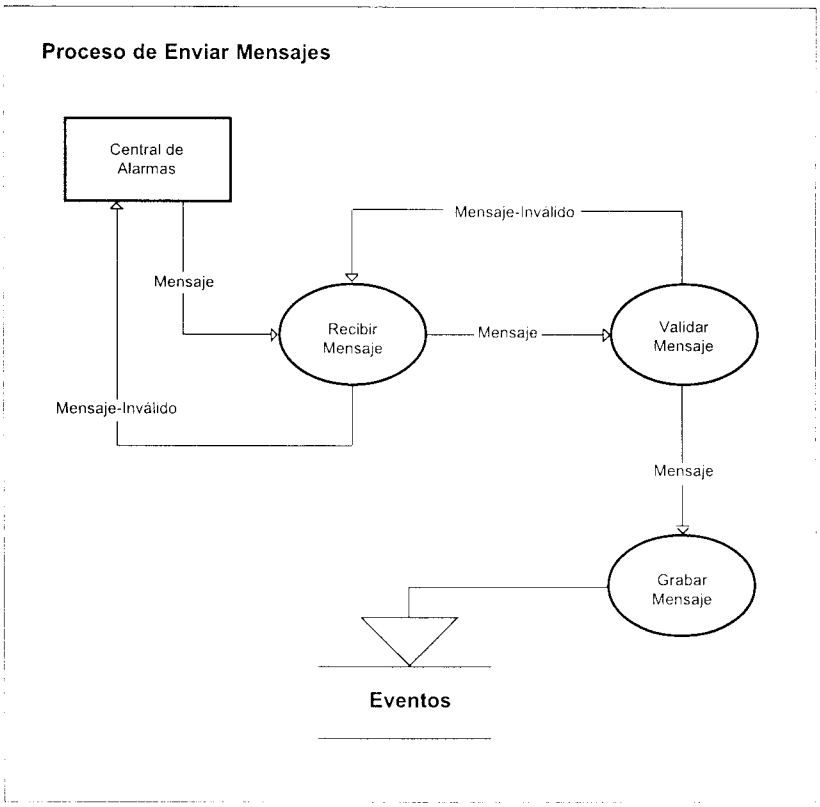
6. Diagrama Funcional

El sistema de control de alarmas tiene dos procesos que procesan datos, estos son:

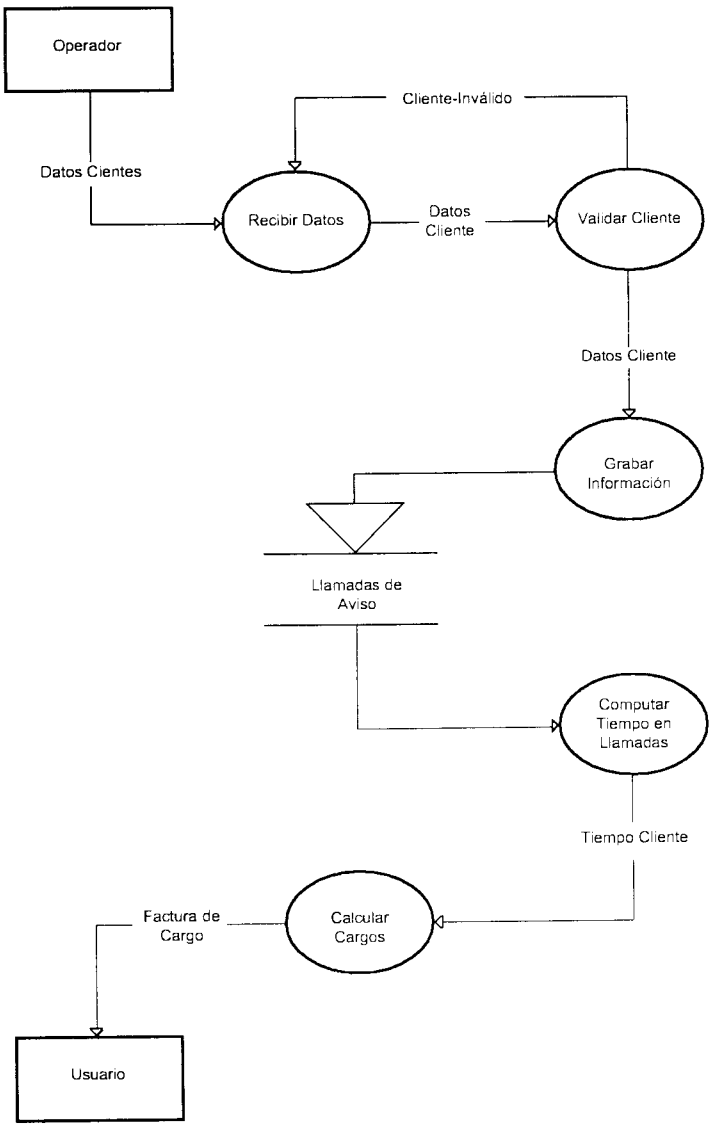
Enviar: Proceso por el cual la central de alarma envía los datos al computador. El computador debe validar los datos recibidos, si son válidos son almacenados, de lo contrario se envía un mensaje de retrasmisión a la alarma.

Registrar Avisos y Computar Cargos: Este proceso recibe los avisos y computa los cargos a los clientes.

A continuación se muestran lo diagramas funcionales para estos procesos.

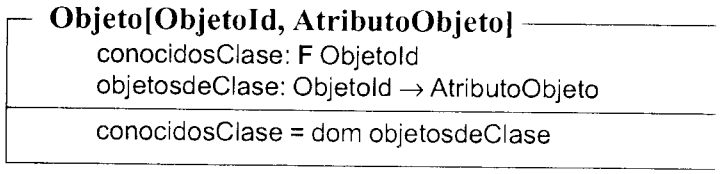


Proceso de Registrar Avisos y Computar Cargos



7. Esquemas Genéricos de OMT-Z

A continuación se describen los esquemas genéricos utilizados en este artículo.



Este esquema genérico es instanciado con una identificación de clase de objeto y los atributos de la clase de objeto a crear y nos retorna una nueva clase de objeto. Dentro del esquema primero se define un conjunto, conocidosClase de tipo ObjetoId (del tipo de la clase de objeto a instanciar), luego se define una función parcial de ObjetoId a los atributos del Objeto, es parcial indicando que no todos los ObjetoId del universo tienen atributos asociados. Finalmente, se define un predicado que asegura que todo objeto conocido por el sistema tiene atributos asociados.

Este es un esquema de relación abstracto que será utilizado por las relaciones entre objeto. Este esquema recibe como parámetro el ObjetoMaestroId y ObjetoDetalleId a entrar en la relación. Dentro del esquema se definen los conjuntos conocidosObjetoMaestro y ObjetoDetalle del tipo ObjetoMaestroId y ObjetoDetalleId respectivamente. Luego se define una relación abstracta (lo más general posible) que será restringi-

da posteriormente por los esquemas reales que utilicen el esquema relación. Se incluyen dos predicados que restringen el dom y ran de la relación a objetos conocidos por el sistema.

Todos los esquemas que definen relaciones incluyen el esquema abstracto Relación.

RelacionUnoaUnoOpcional
[ObjetoMaestroId, ObjetoDetalleId]

Relacion
RelacionMaestroDetalle: ObjetoMaestroId
→ ObjetoDetalleId

∀e: conocidosObjetoMaestro • {RelacionMaestroDetalle
<{e}> } ∈ {0,1}
RelacionMaestroDetalle~ ∈ ObjetoDetalleId →
ObjetoMaestroId

Relación Modelada:

Esquema utilizado para modelar relaciones de uno a uno opcional. Dentro del esquema se define la RelacionMaestroDetalle como una función parcial, ya que la función de objetoMaestroId a ObjetoDetalleId es opcional (0 ó 1). Luego se restringe la función a que existan uno ó cero pares ordenados, cuyo primer componente sea igual, esto es una propiedad de las funciones por lo que ya está dicho. Además, se especifica la función inversa como una función total de ObjetoDetallaId a ObjetoMaestroId, ésta es total ya que para cada elemento del rango se relaciona exactamente un elemento del rango.

RelacionUnoaMuchos
[ObjetoMaestroId, ObjetoDetalleId]
Relacion

RelacionMaestroDetalle ~ ∈ conocidosObjetoDetalle

Relación Modelada:

Esquema utilizado para modelar relaciones de uno a muchos (cero ó más). El esquema define una relación que al invertirla es una función total de ObjetosDetalleId a ObjetoMaestroId (porque cada elemento del rango se relaciona exactamente con un elemento del dominio).

RelacionUnoaMasUno
[ObjetoMaestroId, ObjetoDetalleId]
Relacion

dom RelacionMaestroDetalle = conocidosObjetoMaestro
∀e: conocidosObjetoMaestro • #{ RelacionMaestro Detalle <{e}> } ≥ 1
RelacionMaestroDetalle ~ ∈ ObjetoDetalleid → ObjetoMaestroId

Relación Modelada:

Esquema utilizado para modelar relaciones de uno a más de uno. El esquema define una relación restringida a que: 1. Todos los elementos del dominio entran en la relación, 2. Cada elemen-

to del dominio debe tener más de un elemento imagen en el rango, 3. La relación invertida define una relación total (ya que cada elemento del rango debe relacionarse exactamente con un elemento en el dominio).

RelacionUnoaConjunto
[ObjetoMaestroId, ObjetoDetalleId]

Relación
CI?: **N**

$\forall e$: conocidosObjetoMaestro • #{
RelacionMaestroDetalle <{e}> } ∈ { CI? }
RelacionMaestroDetalle ~ ∈ ObjetoDetalleId →
ObjetoMaestroId

Relación Modelada:

Esquema utilizado para modelar relaciones de uno a un conjunto de posibles número de elementos imágenes. El esquema define una relación restringida a que: 1. El número de imágenes de cada elemento del dominio debe estar en el conjunto de imágenes permitidos, 2. La relación inversa es una función total. El conjunto de imágenes permitidas debe ser dado al esquema como un dato de entrada.

RelacionMuchosaMuchos
[ObjetoMaestroId, ObjetoDetalleId]

Relacion

Relación Modelada:

Esquema utilizado para modelar relaciones de muchos a muchos.

RelacionUnoOpcionalaMuchos
[ObjetoMaestroId, ObjetoDetalleId]
Relacion

RelacionMaestroDetalle ~ ∈ ObjetoDetalleId → ObjetoMaestroId
∀e: conocidosObjetoDetalle • #{ RelacionMaestroDetalle <{e}> } ∈ { 0,1 }

Relación Modelada:

Esquema utilizado para modelar relaciones de uno opcional a muchos. El esquema define una relación restringida a que: 1. La relación invertida es una función parcial de ObjetoDetalleId a ObjetoMaestroId, es parcial por que puede ser 0 ó 1, 2. El segundo predicado es redundante ya que definir la relación invertida como una función asegura que cada elemento solo se relaciona con 0 ó 1 elemento del rango.

RelacionMuchosaMasUno
[ObjetoMaestroId, ObjetoDetalleId]
Relacion

Ve: conocidosObjetoMaestro • #{ RelacionMaestroDetalle < {e}> } ≥ 1

Relación Modelada:

Esquema utilizado para modelar relaciones de muchos a más de uno. El esquema define una relación restringida a que: 1. Cada elemento del dominio debe relacionarse con más de un elemento en el rango.

RelacionUnoOpcionalaUnoOpcional
[ObjetoMaestroId, ObjetoDetalleId]

Relacion
RelacionMaestroDetalle: ObjetoMaestroId
→ ObjetoDetalleId

RelacionMaestroDetalle ~∈
ObjetoDétalleId → ObjetoMaestroId

Relación Modelada:

Esquema utilizado para modelar relaciones de uno opcional a uno opcional. El esquema define una función restringida a que: 1. Es parcial de ObjetoMaestroId a ObjetoDetalleId, debido a la opcionalidad, 2. Es inyectiva ya que cada elemento distinto del dominio se relaciona con cero o con un elemento del rango.

RelacionUnoOpcionalMasUno
[ObjetoMaestroId, ObjetoDetalleId]

Relacion

∀e: conocidosObjetoMaestro • # { RelacionMaestroDetalle
< { e }> } ≥ 1
RelacionMaestroDetalle ~
∈ ObjetoDetalleId → ObjetoMaestroId

Relación Modelada:

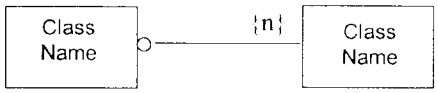
Esquema utilizado para modelar relaciones de uno a más de uno. El esquema define una relación restringida a que: 1. Todos los elementos del dominio entran en la relación, 2. Cada elemento del dominio debe tener más de un elemento imagen en el rango, 3. La relación invertida define una función parcial, ya que cada elemento de su dominio puede estar asociado con uno o ningún elemento del rango.

RelacionUnoOpcionalaConjunto
[ObjetoMaestroId, ObjetoDetalleId]

Relacion
CI?: **N**

$\forall e$: conocidosObjetoMaestro • # {
RelacionMaestroDetalle < {e} > } $\in$ { ΘCI? }
RelacionMaestroDetalle ~ $\in$ ObjetoDetalleId $\rightarrow$
ObjetoMaestroId

Relación Modelada:



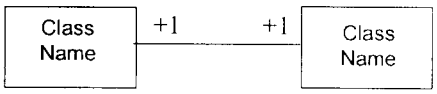
Esquema utilizado para modelar relaciones de uno opcional a conjunto. El esquema define una relación restringida a que: 1. Todos los elementos del dominio tiene un número de imágenes que están dentro del conjunto permisible, 2. La relación invertida define una función parcial, debido a la opcionalidad permitida.

RelacionUnoMasUnoMas
[ObjetoMaestroId, ObjetoDetalleId]

Relacion

$\forall e$: conocidosObjetoMaestro • # {
RelacionMaestroDetalle < {e} > } ≥ 1
$\forall e$: conocidosObjetoDetalle • # {
RelacionMaestroDetalle ~ < {e} > } ≥ 1
dom RelacionMaestroDetalle = conocidosObjetoMaestro
ran RelacionMaestroDetalle = conocidosObjetoDetalle

Relación Modelada :

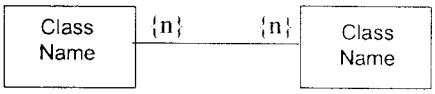


Esquema utilizado para modelar relaciones de uno más a uno más. El esquema define una relación restringida a que: 1. Todos los elementos del dominio tiene más de una imagen, 2. La relación invertida resulta en una relación en la que todos los elementos del dominio tienen más de una imagen, 3. la relación es total en ambos lados.

RelacionConjuntoaConjunto
[ObjetoMaestroId, ObjetoDetalleId]
Relacion

CM?: **N**
CD?: **N**
∀e: conocidosObjetoMaestro • # {
RelaciónMaestroDetalle < {e} > } ∈ { ΘCM? }
∀e: conocidosObjetoDetalle • # {
RelaciónMaestroDetalle ~ < {e} > } ∈ { ΘCD? }

Relación Modelada:



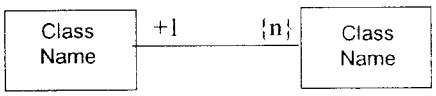
Esquema utilizado para modelar relaciones de conjunto a conjunto. El esquema define una relación restringida a que: 1. El número de imágenes de cada elemento del dominio debe estar en el conjunto dado (igual para la relación inversa).

RelacionUnoMasaConjunto
[ObjetoMaestroId, ObjetoDetalleId]

Relacion
CI?: **N**

∀e: conocidosObjetoMaestro • # {
RelacionMaestroDetalle < {e} > } ∈ { ΘCI? }
∀e: conocidosObjetoDetalle • # {
RelacionMaestroDetalle ~ < {e} > } ≥1
RelacionMaestroDetalle~ = conocidosObjetoMaestro

Relación Modelada:



Esquema utilizado para modelar relaciones de conjunto a conjunto. El esquema define una relación restringida a que: 1. El número de imágenes de cada elemento del dominio debe estar en el conjunto dado, 2. La relación invertida resulta en una relación en la cual el número de imágenes de cada elemento del dominio debe ser más de uno.

RelacionUnoaUno
[ObjetoMaestroId, ObjetoDetalleId]

Relacion
RelacionMaestroDetalle: ObjetoMaestroID →
ObjetoDetalleId

dom RelacionMaestroDetalle = conocidosObjetoMaestro

Relación Modelada:

Esquema utilizado para modelar relaciones de uno a uno. El esquema define una función restringida a que: 1. Es parcial de

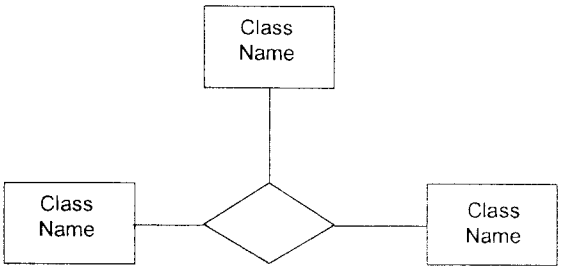
ObjetoMaestroId a ObjetoDetalleId, 2. Es inyectiva porque cada elemento distinto del rango es imagen de un elemento distinto en el dominio. Además, el dominio de la función es igual al conjunto de ObjetosMaestro conocidos por el sistema.

RelacionTernariaCompleta
[ObjetoUnoId, ObjetoDosId, ObjetoTresId]

RelacionUnoaUno[ObjetoUnoId, ObjetoDosId]
[conocidosObjetoUnoId / conocidosObjetoMaestro,
conocidosObjetoDosId / conocidosObjetoDetalle,
relacionObjetoUnoIdaObjetoDosId /
RelacionMaestroDetalle]
conocidosObjetoTresId: **F** QbjetoTresid

$\forall$p: conocidosObjetoTresId •
$\exists_1$ e: relacionObjetoUnoaObjetoDosId •
(first(ObjetoTresId).ObjetoTresId) =
conocidosObjetoTresId.ObjetoTresId

Relación Modelada:



Para definir la relación ternaria se divide el problema en dos partes: 1. Primero se define una relación uno a uno entre las dos primeras clases de objeto que entran en la relación, y 2. Se agrega un predicado que asegure que todo primer componente de la primera relación se asocia a un componente de la clase de objeto tres (como la relación es de uno-a-uno, total entre las tres clases esto asegura que existe una terna única de forma ObjetoUnoId-ObjetoDosId-ObjetoTresId).

Nota: El esquema genérico presupone que las clases de objetos que entra en la relación tienen un atributo identificado con el nombre del objeto en si, que este identificador está en la primera posición del récord y que este identificador es la llave de acceso al objeto.

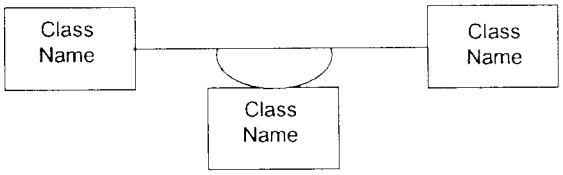
El esquema ternario eventualmente puede ser redefinido por el usuario para permitir otro tipo de relaciones, por ejemplo, asociaciones múltiples entre las clases de objetos.

RelacionConAtributos
[ObjetoUnoId, ObjetoDosId, ObjetoTresId, AtributoLiga]

conocidosObjetoUno: **F** ObjetoUnoId
conocidosObjetoDos: **F** ObjetoDosId
conocidosObjetoTres: **F** ObjetoTresId
objetosdeClase: ObjetoTresId → AtributoLiga

∀p: conocidosObjetoTres • first(e) ∈
conocidosObjetoUno
∀p: conocidosObjetoTres • seconds(e) ∈
conocidosObjetoDos

Relación Modelada:

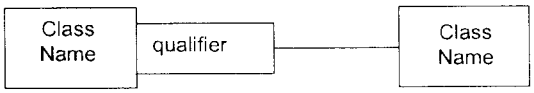


Para definir la relación con atributos en la liga se relizan los pasos siguientes: 1. Se invoca el esquema genérico con los Identificadores de las dos clases que entran en la relación, el identificador de la clase de objeto que se crea en la liga y el esquema que contiene sus atributos. 2. Se define una función parcial para la liga. Esta muestra que todo identificador en la liga tiene un conjunto de atributos asociados. 3. Todo primer componente del identificador de objeto liga, debe existir entre los cono-

cidos de la primera clase de la relación. De la misma manera todo segundo componente debe existir entre los conocidos de la segunda clase de la liga.

Nota: El esquema genérico presupone que el récord de la entidad creada en la liga posee un identificador en forma de tupla. El primer componente de esta tupla es de tipo identificador de la primera clase a relacionar, el segundo es de tipo identificador de la segunda clase a ligar.

Relación Modelada:



Este tipo de cualificador es utilizado para convertir relaciones de muchos-a-uno, a relaciones de uno-a-uno. Para hacer esto se le agrega un identificador (cualificador) a la clase donde se da la relación de muchos.

El esquema genérico creado para modelar este tipo de relaciones tiene los siguientes componentes: 1. Recibe como parámetros el identificador del objeto maestro y detalle, el identificador que cualifica la relación (que debe existir dentro de los atributos del objeto maestro) y el esquema que representa el conjunto de atributos de la clase de objeto maestro, 2. Se define una relación entre los objetos maestro y detalle, 3. Se restringe el esquema con dos predicados, el primero asegura que todo cualificador contenido en la clase maestra tiene su imagen en la clase detalle (para cada valor del cualificador en la clase maestra existe un objeto en la clase detalle identificado con este valor), segundo se define la inversa de la relación que resulta en una función total.

8. Conclusión y Recomendaciones

Consideramos que la utilización de métodos formales en la ingeniería del software podría ser una de las soluciones a los problemas de calidad del software. Construir sistemas con una base matemática formal nos permite verificar y asegurar la calidad de los mismos.

Utilizando métodos híbridos aprovechamos las fortalezas de cada planteamiento y desechamos sus debilidades.

Los métodos formales adolecen de complejidad de utilización y falta de estructura. Sin embargo utilizan notaciones que nos permiten razonar acerca de la corrección de nuestras especificaciones sin necesidad de implementarlas.

Por otro lado, los métodos informales son más estructurados y utilizan notaciones gráficas que proveen un entendimiento de los conceptos del diseño más fácilmente. Pero están pobremente formalizados y en su mayoría se prestan a interpretaciones subjetivas.

Nosotros como ingenieros de software abogamos por metodologías de análisis y diseño de sistemas que provean: una base formal sólida que nos permita encontrar los errores en nuestras especificaciones antes de que se conviertan en errores reales en nuestros sistemas de información. Esta base formal debe ser dotada de herramientas informales que faciliten nuestra interacción con usuarios y personas no familiarizadas con las notaciones formales. Nuestro objetivo es buscar metodologías alternas que combinen las existentes. No estamos de acuerdo con que se creen nuevos planteamientos de desarrollo de software, por el contrario creemos que los existentes son suficientes para mejorar el estado actual de incertidumbre en materia de sistemas de información. Debemos relacionar las fortalezas de los métodos y metodologías actuales con el objetivo de obtener verdaderas metodologías que resuelvan los problemas de calidad del *software* y de esta forma guíen el desarrollo de sistemas por el cami-

no del éxito y la satisfación total de las necesidades de nuestros clientes.

OMT-Z es una metodología híbrida de análisis y diseño de sistemas que combina un planteamiento formal (Z), con uno informal (Object Modeling Tecnique), con el propósito de lograr flexibilidad al especificar notaciones gráficas con las cuales se pueda conceptualizar el diseño y el poder análitico del cálculo de predicado y álgebra relacional.

Nuestra experiencia con OMT-Z ha sido satisfactoria. Utilizamos la parte informal de OMT-Z para levantar los requerimientos del sistema, describir su estructura inicial y verificar que todos los requerimientos estaban cubiertos por el diseño. Una vez que obtuvimos la especificación informal, la formalizamos utilizando la parte formal de OMTZ. Formalizar la especificación nos permitió descubrir y corregir inconsistencias en ésta.