DISEÑO Y ESPECIFICACIÓN DE UNA HERRAMIENTA SOFTWARE DE APOYO A LAS ESPECIFICACIONES DE SISTEMAS DE INFORMACIÓN

-ESFOSIN-

LUIS ALBERTO ESTEBAN VILLAMIZAR

UNIVERSIDAD INDUSTRIAL DE SANTANDER Centro de Documentación y Bibliografía BIBLIOTECA		No. Clasificación XI 7791
No. Adquisición	Fecha Recibo 21 ABR 1998	
No. Inventario 90016	Precio	Dpto. Solicitante

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS
Escuela de Ingeniería de Sistemas e Informática
Maestría en Informática
Línea de Investigación en Ingeniería del Software
Bucaramanga
1998

BIBLIOTECA UIS

DISEÑO Y ESPECIFICACIÓN DE UNA HERRAMIENTA SOFTWARE DE APOYO A LAS ESPECIFICACIONES DE SISTEMAS DE INFORMACIÓN

-ESFOSIN-

LUIS ALBERTO ESTEBAN VILLAMIZAR

Tesis de grado como requisito para optar el título de
Magister en Informática

Director
RICARDO LLAMOSA VILLALBA
Ingeniero de Sistemas
Doctor Ingeniero en Telecomunicaciones

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS
Escuela de Ingeniería de sistemas e Informática
Maestría en Informática
Línea de Investigación en Ingeniería del Software
Bucaramanga
1998

Nota de aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Bucaramanga. 13 de abril de 1998

A mis padres, gestores de mi formación, por quienes hoy soy… quien soy.

A mis abuelos.

A mi hermana Rosalba, su esposo Gregorio y a mi sobrino William Alexander, de quien espero, muy pronto me supere.

A mi familia en Bucaramanga quienes me brindaron su hogar.

A Judith.

A mi compañero de pregrado, Nelson Mollogon Cote, quien hasta los últimos momentos de su corta existencia, siempre se preocupo por estudiar.

A todos los lectores de este proyecto.

AGRADECIMIENTOS

El autor desea expresar sus agradecimientos a:

Dr. Ricardo Llamosa Villalba, director de este trabajo de grado, por su valiosos aportes en pro de mi formación profesional y personal.

Dr. Fernando Ruiz Díaz, director de la maestría en Informática, por su permanente apoyo y colaboración, durante la etapa inicial de mis estudios de posgrado.

A mi familia en Bucaramanga (tío Domingo, señora y familia) por recibirme como un hijo dentro de su hogar, durante estos últimos tres años.

A mis compañeros de posgrado, que de una u otra forma me apoyaron en lo personal y profesional.

A los directivos, docentes y administrativos de la Escuela de Ingeniería de Sistemas e Informática.

A mis estudiantes de pregrado, que con sus inquietudes han fomentado mi superación profesional.

CONTENIDO

BIBLIOTECA UIS

BIBLIOTECA UIS

BIBLIOTECA UIS

LISTA DE ILUSTRACIONES

LISTA DE ANEXOS

BIBLIOTECA UIS

GLOSARIO

Abstracción: Concepción mental de un sistema.

Acoplamiento: Lograr que varios módulos se integren para lograr un mejor rendimiento del sistema.

Adaptabilidad: Mejoramiento progresivo de un sistema con el fin de estar a la par con la tecnología existente.

Arquitectura del Programa: Diseño interno del programa.

Aserción: Respuesta que nos da un sistema.

Calidad del Software: Verificar si el software cumple con las expectativas del mercado.

Cohesión: Unión de varios procedimientos para crear un sistema mayor.

Componentes: Cada una de las partes de un sistema (un todo).

Complejidad. Se aplica a los objetos compuestos sonados por partes o la reunión de varias cosas en donde hay que considerar muchos aspectos que no son fáciles de comprender y resolver.

Confiabilidad: Se refiere al correcto funcionamiento de un sistema y su potencia para reaccionar, tal como se prevé, ante entrada o estímulos previstos e imprevistos.

Configuración: Sistema particular de componentes interelacionados.

Criterio de Aceptación: Ítems que se establecen para determinar si el sistema cumple con las especificaciones requeridas.

Datos e Información: Los datos son representaciones abstractas de hechos (eventos, ocurrencias o transacciones) u objetos (personas lugares etc.). Cuando estos se ordenan en un contexto adecuado

BIBLIOTECA UIS

por medio de un procesamiento, adquieren significado y proporcionan conocimiento sobre los hechos u objetos que los originan, transformándose en lo que se denomina información [22].

Delineamiento: Guía impuestas para seguir en el desarrollo de un sistema.

Diagrama de Flujo de Datos: Diagrama que muestra el flujo de datos a través de procesos o procedimientos, representando los archivos, las dependencias y el origen de los datos fuentes.

Diadatos: Gráficos para describir detalladamente la estructura de un sistema de información. El gráfico del modelo entidad relación es considerado un diadato.

Diaprocesos: Gráficos para visualizar el comportamiento de un sistema de información y los estados o faces por las que pasa un proceso. Un diagrama de estados es considerado un diaproceso.

Diseño Arquitectónico: Definición de pautas para la construcción de sistemas.

Diseño Detallado: Descripción pormenorizada de cada uno de los componentes de un sistema.

Diseño Preliminar: Primer diseño realizado con el fin de tener una imagen global del sistema.

Diseño: Descripción narrativa o gráfica de un sistema.

Documentación: Serie de técnicas utilizadas para definir las especificaciones y para presentar el diseño del sistema.

Eficiencia: Serie de servicios o ventajas que un sistema presta a los usuarios finales.

Especificación Formal: Explicación detallada en lenguaje técnico.

Especificación: Explicación detallada de las funciones y parámetros que debe cumplir el sistema.

Fase de Operación y Mantenimiento: Fase en la cual se realizan mejoras o correcciones a un sistema con el fin de seguir siendo válidos.

Fase de Requerimientos: Etapa del desarrollo en el que se fija el tipo de necesidades que debe satisfacer el sistema.

Forzar Estándares: Obligar a que el sistema cumpla determinados estándares.

Generador Automático de Pruebas: Software que produce una serie de datos con el fin de verificar el correcto funcionamiento del sistema.

BIBLIOTECA UIS

Herramienta Automática para Diseño: Programa que requiere de ciertas entradas para desarrollar una aplicación.

Herramienta de Verificación Automática: Software automático de validación.

Integración: Unión de una serie de módulos con el fin de formar un sistema.

Ítem de Configuración: Criterios para la escogencia de los componentes de una configuración determinada.

Lenguaje Orientado a la Aplicación: Lenguaje con el que el usuario ejecuta ciertos procesos en una aplicación.

Mantenimiento Adaptativo: Realizar cambios a un sistema con el fin de que este no quede obsoleto en el tiempo.

Metodología para el Desarrollo: Serie de pasos o etapas que se siguen en la construcción de un sistema.

Modelo Físico: Representación a escala de un sistema.

Modelo Lógico: Representación lógica del funcionamiento de un sistema.

Precisión: Margen de error que se le da a un sistema en un proceso determinado..

Prototipo: Versión preliminar de un sistema.

Prueba de Aceptación: Prueba realizada para ver si el sistema está trabajando de acuerdo a las especificaciones

Prueba Formal: Validación del funcionamiento del sistema, observando el comportamiento de la máquina, con respecto a las especificaciones y requisitos establecidos en el diseño.

Rata de Fallas: Porcentaje de fallas observadas en un sistema ante un número determinado de pruebas.

Sistema: Grupo de componentes interelacionados para realizar una tarea.

Software de Aplicación: Cualquier programa de ingreso de datos, actualización, consulta, o informe que procese datos para el usuario.

BIBLIOTECA UIS

Tolerancia a Fallas: Resistencia que tiene un sistema para que un error externo no lo afecte y pueda seguir funcionando normalmente.

Validación: Hacerle pruebas a un sistema para saber si este reacciona correctamente.

Verificación: Comprobar que los resultados arrojados por el sistema, en los procesos de validación, sean correctos. (comparar con los resultados reales).

BIBLIOTECA UIS

0. INTRODUCCIÓN

Cada una de las etapas de la ingeniería del software tiene factores críticos que determinan la calidad del producto deseado; lo que implica tener una metodología adecuada en cada una de estas etapas[1].

La fase de modelado y especificación de sistemas de información es uno de los aspectos claves en el área de la Ingeniería del Software y en general, el modelado siempre ha sido aceptado por ingenieros, científicos, artistas y gestores, como una técnica incalculable para presentar ideas, ayudar a la comprensión e incluso, predecir nuevas formas de hacer las cosas[25]. Las buenas técnicas de modelado están apoyadas por normas rigurosas y convencionales que evitan ambigüedad, inconsistencia e incompletitud y facilitan la comunicación.

Esta Tesis demuestra como un modelo[2] de especificación formal de sistemas de información apoya la etapa de realización de especificaciones, mediante políticas metodológicas que generan automáticamente las especificaciones formales, a partir de las descripciones gráficas informales.

El proyecto " **ES**pecificaciones **FO**rmales de **S**istemas de **In**formación " - ESFOSIN-, consta de tres partes: un modelo de especificación, un conjunto de reglas metodológicas y un prototipo de herramienta software.

Este documento, describe los resultados obtenidos con ESFOSIN, en cuatro partes principalmente:

La primera (Capítulos 1 y 2)- definición y fundamentación, describe lo que se desea lograr y los conceptos que facilitarán la comprensión de las ideas planteadas.

La segunda (Capítulos 3, 4 y 5) - planteamiento -, describe el modelo propuesto para la generación de especificaciones formales, la especificación de la herramienta que facilita el proceso, y una aplicación sencilla, siguiendo los delineamientos metodológicos propuestos.

---

[1] Por lo general estas metodologías se fundamentan en un conjunto de criterios y herramientas.

[2] Planteado como un prototipo software.

BIBLIOTECA UIS

La tercera parte (Capítulos 6 y 7) - síntesis de resultados -, muestra las conclusiones de la Tesis (producto del estudio realizado) y las recomendaciones para posteriores trabajos, enmarcados dentro de la misma línea de investigación.

Finalmente el documento dispone de los anexos, que aclaran la terminología y las temáticas comúnmente utilizadas en el área de sistemas, y más precisamente, en el área de modelado y especificación de sistemas de información.

BIBLIOTECA UIS

1. DESCRIPCIÓN DEL PROBLEMA

1.1 DEFINICIÓN DEL PROBLEMA

La Ingeniería del software es considerada una disciplina que integra conocimientos, métodos[3], herramientas[4] y procedimientos[5] para el desarrollo de software[25]. Esta Ingeniería presenta diversas áreas de estudio e investigación, cada una de las cuales, tiene como visión la obtención de un producto final de muy alta calidad. Una de ésta áreas, enfoca su atención a los sistemas de información; que sin duda, hoy es una necesidad para cualquier tipo de organización que pretenda desempeñar su función con eficacia y eficiencia.

En el desarrollo de sistemas de información de alta calidad, se debe seguir un proceso metodológico muy riguroso, muchas veces complejo, que sin la ayuda de herramientas adecuadas y guías didácticas se hace difícil de realizar, obstaculizando el logro de un nivel mínimo de calidad.

Como la fase de especificaciones, es base y fundamento para el desarrollo de sistemas de información, esta debe contar con herramientas[6] adecuadas, que apoye la fase de análisis.

1.2 VISIÓN

"Aseguramiento de la calidad software mediante la formalización de las etapas de producción, con miras hacia la generación automática de código".

---

[3] Suministran la estructura básica y las instrucciones de diseño, para desarrollar el software. Indican "Cómo" construir técnicamente el software. Incluyen tareas tales como: planeación de proyectos, análisis de requerimientos del sistema, diseño de las estructuras de datos, codificación, pruebas y mantenimiento.

[4] Suministran un soporte automático o semiautomático para los métodos, por lo general se integran todas las herramientas de tal forma que la información genera por una de ellas, pueda ser usada por otra, creando lo que se conoce como Ingeniería del Software Asistida por Computador (CASE).

[5] Son los elementos integradores de los métodos y las herramientas, facilitando un desarrollo racional y oportuno del software. Definen la secuencia en que se aplican los métodos, las entregas de resultados requeridos, los controles que ayudan a asegurar la calidad y las directrices para la evaluación del progreso del proyecto informático.

[6] Dichas herramientas se fundamentan en un modelo de especificación apropiado, con unos delineamientos metodológicos propios.

BIBLIOTECA UIS

1.3 OBJETIVOS

1.3.1 Objetivo General (Misión)

Diseñar y especificar una herramienta que sirva de apoyo al proceso de especificación formal, a partir de descripciones informales[7] de los sistemas de información.

1.3.2 Objetivos Específicos

- Diseñar un modelo de especificación que permita la generación de especificaciones formales a partir de descripciones gráficas de los sistemas de información. Dicho modelo se fundamenta en la descripción de un sistema de información a partir de la representación de su estructura y de su comportamiento.

- Presentar los delineamientos metodológicos a seguir en la etapa de especificación de acuerdo al modelo propuesto, describiendo los principales pasos y productos para la generación de especificaciones.

- Diseñar un prototipo exploratorio que permita la identificación de las principales componentes del modelo propuesto.

1.4 METAS

- ↪ Identificar y definir las condiciones necesarias para la generación de especificaciones formales a partir de descripciones informales de sistemas de información medianos[8].

- ↪ Describir gráfica y textualmente el modelo de especificación y la función e integración de cada una de sus componentes

- ↪ Documentar el modelo con una descripción global y detallada de cada una de sus componentes, de tal forma que permita en un futuro, la creación de una herramienta software que sirva de apoyo al proceso de especificación[9].

BIBLIOTECA UIS

- Describir la técnica o lenguaje de especificación formal a ser utilizada
- Definir las reglas de transformación entre los diagramas descriptores del sistema de información y el lenguaje de especificación formal
- Diseñar y desarrollar un prototipo exploratorio de herramienta software que soporte los aspectos básicos del modelo propuesto[10].

1.5 JUSTIFICACIÓN

1.5.1 Antecedentes

1.5.1.1 Académicos

La Línea de Investigación en Ingeniería del Software - LIS[11], de la Universidad Industrial de Santander, desde su creación en 1990, ha tenido como visión y preocupación constante el análisis y generación de herramientas[12] de apoyo al aseguramiento de la calidad en sistemas. Para ello ha trabajado, con diferentes proyectos de pregrado y posgrado, en el estudio de metodología, estándares, guías, técnicas.... que permitan lograr su misión.

LIS ha desarrollado algunas herramientas software encaminadas hacia el mejoramiento de la calidad en los sistemas, entre los que se destacan benchmarking, modelamiento de procesos, planeación estratégica, procesos de diagnóstico, workFlow, estimación de recursos software, procesos de producción, control de calidad de cursos de educación superior y gestión de procesos, entre otros.

Actualmente, el Centro de Innovación y Desarrollo Línea de Investigación en Ingeniería del Software - CIDLIS, fue premiado como grupo de excelencia por COLCIENCIAS mediante

---

7 Representadas en forma gráfica y descriptiva.

8 Consideramos sistemas de información medianos, desde el punto de vista del prototipo exploratorio, a aquellos cuya estructura puede ser descrita con no más de 50 entidades.

9 Dicha herramienta recibirá descripciones gráficas del sistema y generará especificaciones formales.

10 El prototipo es construido con el fin de apoyar la identificación de componentes del modelo, más no para la validación del modelo completo.

11 Hoy reconocida como Centro de Innovación y Desarrollo Línea de Investigación en Ingeniería del Software - CIDLIS.

12 Estas herramientas, no solo son programas, sino también metodologías, procedimientos, guías, etc.

BIBLIOTECA UIS

convocatoria a centros y grupos de Investigación, en todo el País. Ha logrado la aprobación, por parte de COLCIENCIAS, del proyecto "Gestión Académica y Administrativa de instituciones de Educación Superior"- GAYA. Ha presentado proyectos en asocio con el Servicio Nacional de Aprendizaje -SENA y con empresas privadas como Proyectos y Construcciones.

Esta Tesis está enmarcado dentro de la misión y los delineamientos del CIDLIS, de acuerdo a los objetivos planteados.

1.5.1.2 Sociales

Aunque, desde ya hace algún tiempo existen numerosos métodos y técnicas formales dentro de la ingeniería del software, también es cierto que existe una apatía, por parte de los analistas, hacia el uso de los formalismos, lo cual directa o indirectamente, repercute en la calidad del producto final.

El tradicionalismo metodológico y la supuesta 'experiencia practica', de muchos analistas, que apoyados en la idea: "... Hay que terminar rápidamente la etapa de diseño para que al final del proyecto quede tiempo suficiente para resolver los problemas causados por haber realizado muy rápidamente el diseño..." [2, 23], hacen que el proceso de especificaciones sea muy informal[13].

1.5.1.3 Económicos

La deficiente especificación de un producto, en la etapa de análisis y diseño, repercute con mayores costos en las posteriores etapas, en las cuales se invierte mucho tiempo y recursos, tratando de eliminar ambigüedades, inconsistencias y demás imprevistos causados por el informalismo admitido en las primeras etapas de concepción. De igual forma, el mantenimiento se hace tediosos y costoso, debido a la documentación inexacta, incompleta o inexistente de dicha especificación.

1.5.1.4 Tecnológicas

En las pasadas dos décadas se han desarrollado varios lenguajes de especificación formal para remplazar las técnicas de especificación en lenguaje natural. Los desarrolladores de estos lenguajes están en el proceso de construcción de entornos interactivos que:

---

[13] Lo cual no garantiza que la calidad de los productos software sea la ideal.

BIBLIOTECA UIS

- Faciliten al analista crear una especificación basada en el lenguaje de un sistema o software.

- Llamen a herramientas automáticas que traduzcan las especificaciones basadas en lenguaje en código ejecutable y

- Faciliten al cliente utilizar el código ejecutable del prototipo para refinar los requerimientos formales.

Los lenguajes de especificación tales como PSL, RSL, IORL, GYPSY, OBJ [25] y muchos otros, están haciendo un esfuerzo para conseguir un paradigma de ingeniería del software automatizada:

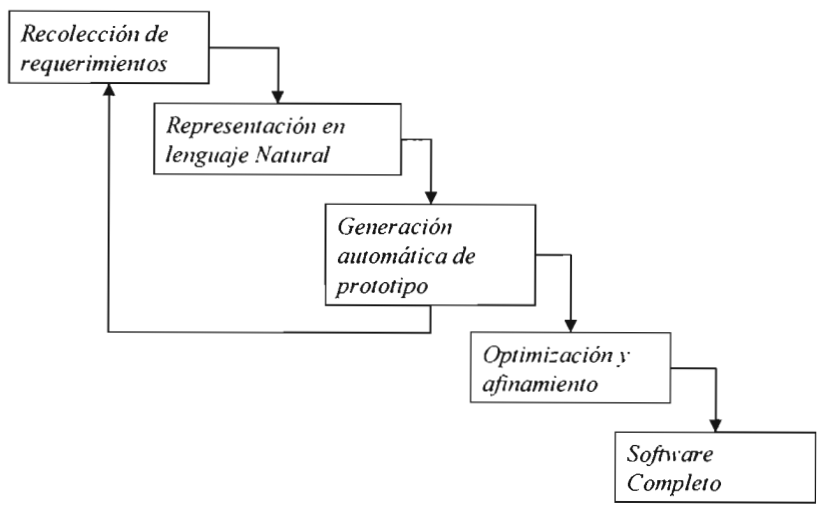

Figura 1 Paradigma de la Ingeniería del Software Automatizada

La aplicación de este paradigma se hace viable con algunos lenguajes de especificación, como VDM (Método de desarrollo de Viena), Z[14], B, RSL, OBJ que presentan un alto componente de definiciones matemáticas.

Algunas técnicas no formales como MER[15], [9, 16, 18] pretende describir la parte estructural de un sistema; en tanto que otras técnicas como las maquinas de estado [16] y/o los estándares ISO 9000[17], describen el comportamientos.

BIBLIOTECA UIS

Otros métodos de especificación formales e informales, que se están utilizando en la etapa de diseño [25], se muestran en la siguiente tabla:

Nombre del método/herramienta	Categoría[18]
Desarrollo de sistemas estructurados de datos	M,T
DesignAid	T
Excelerator	T
Higher Order Software (HOS)	M,T
Information Engineering Workbench	T
Desarrollo de sistemas de Jackson	M,T
Construcción lógica de sistemas (CLS)	M
PSL/PSA	F,T
Software Requirement Enginnering Metodology (SREM) and (SYSREM)	F,T
Structured Analysis (SA)	M,T
Structured Analysis Design Technique (SADT)	M,T
Herramienta de Análisis Estructurado	T
System Development Methodology (SDM)	M,T
Technology for Automated Generation of System (TAGS)	F,T

Tabla 1 Tecnicas y Herramientas para el diseño

1.5.2 Estado del arte

Del estudio teórico realizado se concluye los siguientes ítems relacionados con el ambiente temático de interés.

- **Especificaciones:** las especificaciones son cada vez más importantes dentro de las actividades y procesos que conlleva la ingeniería del software, y de su claridad y exactitud depende la calidad del software [3, 6, 8, 13, 16, 19, 22, 25].

- **Métodos formales:** se han realizado grandes investigaciones para introducir métodos formales en todas las etapas de desarrollo de Software; aún no aceptados en la práctica. Una razón para esta apatía hacia los procesos de especificación formales son sus estrictas

---

[14] Con algunas extensiones hacia el diseño orientado a objetos como lo son VDM++ y Z++ [2] [3]

[15] Basado en el Modelo Entidad Relación Extendido.

[16] Utilizadas para TCASII [10].

[17] El cual describe la parte procedimental del sistema.

[18] M= Técnica manual de análisis; T= Herramienta automatizada; F= Lenguaje de especificación formal.

BIBLIOTECA UIS

interpretaciones semánticas y su naturaleza conceptual concisa[19]. Este rechazo, también se debe a que las metodologías tradicionales[20] prevalecen en la industria, haciendo usos de descripciones gráficas y/o informales en lenguaje natural[21] [2, 5, 9, 10, 11, 16, 17, 21].

- **Las fortalezas y debilidades de la especificación formal e informal**, son en parte complementarias. Mientras que las especificaciones informales tienen la ventaja de que son fáciles de entender para el usuario facilitando el proceso de extracción de requisitos, por otro lado las especificaciones formales proporcionan concisión, precisión y la base matemática casi suficiente para la generación de código en forma automática [3, 5, 8, 9, 10, 16, 17, 18, 25].

- **Uso de métodos formales:** los métodos formales no reemplazan los métodos informales predominantes. Aunque las ventajas de usar métodos formales son muy valiosas, el proceso de escribir especificaciones formales implica un conocimiento adecuado[22], lo que puede conducir a errores, si es probado por una persona no versada en formalismos [3, 4, 6, 16, 18, 25, 26].

- **Formales Vs. Informales**: recientemente se han realizado diferentes estudios para cerrar las brechas existentes entre la descripción informal y la especificación formal. Los efectos laterales de ser informal son: ambigüedad, pobre ordenamiento de requisitos, contradicciones, incompletitud o especificación parcial y/o imprecisas. Por otra parte, llega naturalmente a los usuarios y los métodos usados son relativamente fáciles de aprender. También se ha examinado formalidad e informalidad, como dos caras opuestas de un espectro, entre las cuales se encuentran diferentes métodos semiformales de especificación [3, 9, 10, 16, 17, 25].

- **Calidad Software**: han emergido estándares que reconocen por completo la necesidad y el potencial de las especificaciones formales, como base para lograr alta calidad del software y aplicaciones de seguridad. Aparte de cada área de aplicación en particular, es claro que

---

[19] Lo cual no promueve un buen medio de comunicación entre los profesionales del software.

[20] Sobre todo en las etapas iniciales de concepción de un sistema, que son críticas para asegurar calidad.

[21] Lo cual facilita la presencia de la ambigüedad, las contradicciones y la incompletitud.

[22] En notaciones e interpretaciones matemáticas.

BIBLIOTECA UIS

el desarrollo del software puede beneficiarse del uso de técnicas de especificación convenientes para cumplir algunos requerimientos importantes [6, 8, 18, 25]:

- Permitir más que un simple nivel de formalismos (formular abstracción)
- Proveer conceptos fáciles para modularizar.
- Soportar desarrollo iterativo con especialización.

1.5.3 Importancia de la Tesis

Dado que: la etapa de especificaciones es base fundamental[23] en el desarrollo de sistemas de información [25], los formalismos garantizan un mejor nivel de calidad [16], la metodología orientada por objetos se está imponiendo en las diferentes etapas dentro del desarrollo de los sistemas de información [10, 26], las herramientas gráficas son cada vez más indispensables en el diseño de software [9, 11, 15], actualmente, las técnicas de modelado visual son herramientas invaluables de apoyo al desarrollo [11].... hacen que esta propuesta, está enmarcada dentro de una temática de interés general y actual dentro de la ingeniería del software

1.6 METODOLOGÍA DEL PROYECTO

1.6.1 Estructura organizacional

Básicamente el proyecto tiene tres componentes: Especificación, Ingeniería y Producción. Dentro del aspecto de especificaciones se encuentra todo lo referente a los recursos, actividades y productos que conllevan a la descripción de cada una de las componentes del modelo; la fase de ingeniería hace referencia a la construcción y refinamiento del modelo global, fundamentados en las componentes identificadas en la etapa de especificación y finalmente la producción comprende el diseño y la aplicación que soporta y valida dicho modelo.

---

[23] Por lo tanto esta fase debe tener prioridad en el desarrollo de herramientas que le apoyen en su objetivo, con el fin de lograr productos de alta calidad.

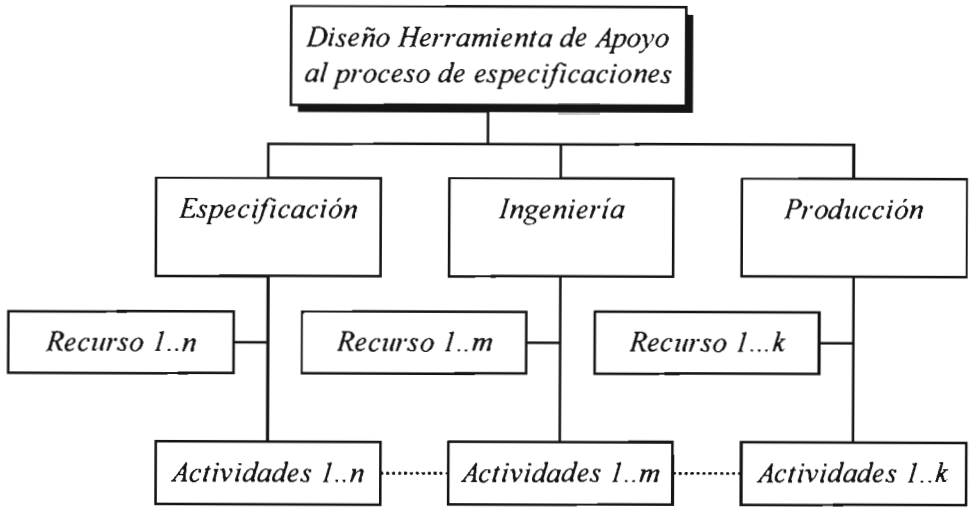

Figura 2 Estructura organizacional del proyecto

1.6.2 Estructura Funcional

Para el desarrollo del proyecto se utilizó una metodología inductiva que permitió analizar inicialmente los pequeños aspectos que conforman cada una de las componentes del modelo para obtener una descripción funcional y estructural de cada componente individual[24]. Este proceso inductivo permitió la construcción de un prototipo[25] software, para soportar los aspectos más relevantes del modelo inicial. Con estos productos, se hizo el refinamiento repetitivo de modelo, prototipo y documentación, hasta lograr la misión y los objetivos propuestos. Cada una de las actividades realizadas para el logro de los objetivos se plantearon y ejecutaron metodológicamente, analizando el estado actual del estudio referente a la temática, sus prerequisitos[26] y posrequisitos[27].

---

[24] Sin descuidar la integración de dichas componentes ni la visión global del modelo.

[25] Con el cual se valida y verifica el modelo.

[26] Condiciones necesarias para poder realizar la actividad.

[27] Estado final del estudio después de desarrollar la actividad.

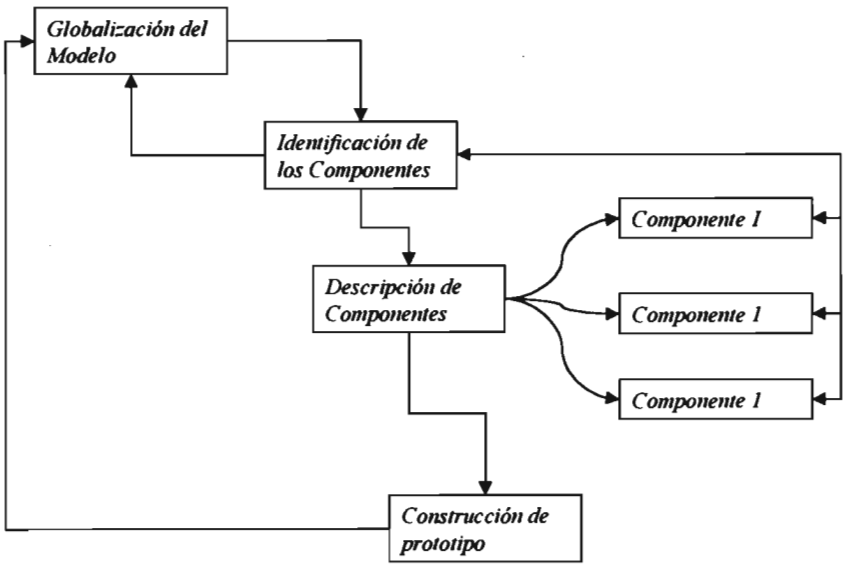

Figura 3 Estructura Funcional del Proyecto

BIBLIOTECA UIS

2. FUNDAMENTACIÓN TEÓRICA

2.1 SISTEMAS DE INFORMACIÓN

Un sistema de información[28] es un sistema[29], cuya componente estructural es la información y su objetivo es apoyar la toma de decisiones.

Todo sistema de información debe incluir las siguientes características [22]:

- Es un sistema abierto, interactúa con su ambiente mediante el intercambio de información y se adapta a las necesidades del ambiente que lo contiene.
- Las entradas al sistema están constituidas por información[30] y su salida por información de apoyo a la toma de decisiones.
- Cada una de las componentes del sistema, cumple un rol distintivo dentro de la organización[31], como parte importante en la toma de decisiones.

---

[28] En los anexos, se presentan otras definiciones relevantes.

[29] Conjunto de elementos físicos o abstractos interrelacionados que operan en conjunto a fin de lograr un objetivo.

[30] El concepto de dato e información es relativo de acuerdo a quien lo necesite (ver terminología y acrónimos).

[31] Generalizando y tomando como referencia el modelo social, podemos considerar una organización como una coordinación racional de procesos dentro de un grupo de elementos básicos con el propósito de lograr una misión (la organización es un sistema).

Todo sistema de información se representa mediante un modelo que refleja su estructura y comportamiento. Dicho modelo se representa o describe con diferentes técnicas y metodologías[32] formales e informales. Dentro del conjunto de técnicas informales para la descripción de un sistema de información, están las técnicas gráficas con las cuales se representa la estructura[33] y el comportamiento[34] del sistema en cuestión.

El presente trabajo recurrió a las técnicas gráficas de especificaciones informales, apoyadas en asistentes de descripción, para modelar gráficamente el sistema y así obtener una especificación Formal.

2.2 TÉCNICAS DE ESPECIFICACIÓN [35]

No hay un único estándar en las técnicas de especificación de los sistemas de información, por el contrario las técnicas son diversas y cada una posee sus ventajas y desventajas [8].

2.2.3 Técnica de especificación formal[36] (TEF)

Formalmente, un lenguaje de especificaciones formales, es una tripla: $< Sin, Sem, R >$ donde *Sin* y *Sem* son conjuntos y *R* es un subconjunto del producto cartesiano de *Sin* x *Sem*[37]. Donde *Sin* es llamado el dominio sintáctico del lenguaje. *Sem* es el dominio semántico y *R* es una relación satisfecha.

Un lenguaje de especificaciones formales (LEF) provee una notación[38], un universo de objetos[39] y una regla precisa para definir que objetos satisfacen cada especificación. Por lo tanto, una especificación es una frase escrita en términos de los elementos del domino sintáctico, denotada

---

[32] La mayoría de estas técnicas y metodologías son informales, mientras que otras, son formales exigiendo la utilización de reglas estrictas para la construcción del modelo, sin dar oportunidad a la ambigüedad ni a la inconsistencia.

[33] Como por ejemplo el diagrama del modelo entidad relación.

[34] Como las maquinas de estados.

[35] Una especificación es una alternativa de descripción, a la implementación del sistema. Su objetivo primario es capturar el "Qué" en el interrogante: "¿Qué hace el sistema?", en lugar del "Cómo".

[36] En las Especificaciones formales de Software, el diseño de arquitecturas, provee los elementos lógicos para especificar; luego es un requisito esencial, previo a la especificación formal.

[37] Una relación entre los conjuntos Sin y Sem.

[38] Su dominio sintáctico.

[39] Su dominio semántico.

BIBLIOTECA UIS

mediante un subconjunto del dominio semántico. Describe exactamente un sistema de información, sin dar posibilidad a la ambigüedad, la contradicción o la incompletitud. Los LEF son similares a los lenguajes de programación ya que disponen de una Sintaxis[40] y una Semántica[41]

Sin una apropiada semántica matemática, una técnica de descripción no puede ser llamada formal, por cuanto es la base para deducir los conceptos y conclusiones que se infieren y administran. El cálculo lógico es la base conceptual y metodológica de la semántica [19].

VDM y Z, son métodos formales especialmente apropiados para la el diseño de sistemas, en el cual dos de las actividades más importantes son descomposición[42] y refinamiento[43]. Las notaciones como VDM involucran el uso de un número de símbolos especializados que debe ser memorizado[44], otros lenguajes incorporan una notación mnemónica que es menos extraña. El lenguaje de especificación, Z usa gráficas[45] para estructurar especificaciones.

Las técnicas formales utilizan reglas estrictas para construir un modelo del sistema apoyados en sistemas de máquinas de estados, sobre las cuales es posible aplicar fórmulas matemáticas precisas con el fin de verificar si cumplen propiedades importantes.

---

[40] Conjunto de palabras y reglas de escritura validas dentro de la técnica.

[41] Único significado de las formas sintácticas.

[42] La descomposición es el proceso de partición de un sistema en pequeños módulos. Los analistas puede escribir especificaciones para capturar las interfaces entre estos módulos.

[43] El refinamiento involucra trabajo en diferentes niveles de abstracción, refinando cada uno de los módulo sobre un nivel general, para crear colecciones de módulos a un nivel inferior. Cada paso de refinamiento requiere mostrar que una especificación a un nivel, satisface la especificación a un nivel más alto.

[44] Esto conduce a dar importancia extra a la curva de aprendizaje.

[45] Esto mejora su legibilidad y fomenta el desarrollo incremental de especificaciones.

BIBLIOTECA UIS

Las técnicas formales han sido utilizadas en los últimos años en la descripción formal de protocolos de comunicación[46] y desarrollo de servicios en el contexto de los estándares de la ISO.

La forma más simple de especificación formal, es la axiomática[47], donde un sistema se representa como un conjunto de funciones. Cada función se especifica usando precondiciones y poscondiciones que son los predicados[48] sobre las entradas y salidas de una función.

Estos métodos simples hacen ver el problema de escalabilidad de la especificación, en dos dimensiones[49]: el tamaño[50] y la complejidad[51] de la especificación.

Las especificaciones de gran tamaño, son difíciles de comprender y por eso es importante que un lenguaje de especificación contenga una estructura que permita su desarrollo incremental. Las consideraciones a tener en cuenta para crear una especificación formal incluyen:

- Las condiciones sobre las que el componente del software se debe desempeñar.
- Condiciones de errores en las entradas.
- El rendimiento de componentes cuando se presentara un error.
- La transformación de las entradas.
- El efecto de los parámetros de entrada, sobre la componente.

Finalmente, la siguiente tabla, muestra el análisis DOFA[52] de los métodos formales:

---

[46] La naturaleza formal de estas especificaciones hace posible aplicar ciertos modelos matemáticos a los protocolos durante su desarrollo, para validar su corrección. Pues el uso de especificaciones escritas en lenguaje natural da la ilusión de comprensión, pero lo único que generan son especificaciones informales que frecuentemente contienen ambigüedades y ofrecen obstáculos para las pruebas y la corrección de un protocolo.

[47] Esta técnica se usa actualmente, sobre todo, para pequeños sistemas o para pequeños componentes del sistema.

[48] Una expresión booleana que es cierta o falsa y cuyas variables son los parámetros de la función que se desea especificar. Los predicados incluyen: Operadores (tales como =, >, <, no, y, o), cuantificadores universal y existencial, el operando que se usa para representar el conjunto sobre el que se aplica el cuantificador.

[49] Para proveer una manera sistemática de pensar y razonar sobre especificaciones, los métodos formales pueden ayudar en el manejo de ambos tipos de complejidad.

[50] Las herramientas pueden ayudar al tamaño de especificación.

[51] Una inherente complejidad resulta desde la complejidad interna y/o complejidad de interfaces.

[52] Debilidades, Oportunidades, Fortalezas, Amenazas.

Fortalezas	Debilidades
• Provee conocimientos y comprensión de los requerimientos del software y de su diseño. • Dada una especificación formal de un sistema y una definición completa del lenguaje de programación formal, es posible probar que un programa cumple con su especificación. • Tienen la posibilidad del procesamiento automatizado, la animación de una especificación para proveer un prototipo del sistema y el análisis matemático para evaluación de consistencia • Facilitan la uniformidad en la comunicación[53], ya que en la práctica, aparte de quienes escriben la especificación, hay varios tipos de lectores: los clientes, programadores, verificadores, y las máquinas[54]. • En el análisis de requerimientos, ayudan a aclarar el conjunto de requerimientos informalmente constatables de un cliente y da a conocer contradicciones, ambigüedades, e incompletitudes.	• Fuerte notación matemática, no adecuada para la mayoría de analistas tradicionales. • Se han dirigido más esfuerzos investigativos al desarrollo de notaciones sin el apoyo de herramienta.
Oportunidades	**Amenazas**
• Pueden usarse como una guía para la creación de pruebas para cualquier componente particular del sistema. • Pueden aplicarse en todas las fases de desarrollo de sistema[55]. • Probar la satisfacción de una especificación frecuentemente genera suposiciones adicionales[56], que deben ser ejecutadas para ser válido. Un método formal provee el lenguaje para afirmar esta demostración y la	• La gestión del software es inherentemente conservadora y existe indisposición para adoptar nuevas técnicas si el resultado final no es inmediatamente obvio. • Los clientes probablemente no están familiarizados con especificaciones formales y pueden indisponerse frente a las actividades desarrolladas que ellos no comprenden y por lo tanto no pueden controlar.

BIBLIOTECA UIS

estructura para efectuarla. • La comprobación[57] formal de un sistema, es imposible sin una especificación formal. • En la validación, los métodos formales ayudan a probar[58] y depurar. • Algunas clases de sistemas son difíciles de especificar usando técnicas actuales.	• Los analistas y programadores no se han entrenado en técnicas formales de especificación. • Hay ignorancia generalizada de técnicas de especificación actualizadas y de sus usos.

Tabla 2 Análisis DOFA de los métodos formales

2.2.1 Lenguajes Lógicos Concurrentes (LLCs)

La idea de esta familia de técnicas es la representación de sistemas de estados mediante lenguajes lógicos, que describen las transiciones en forma de hechos. Concretamente el modelo de ejecución de un sistema de este tipo, es un conjunto de procesos concurrentes que se comunican mediante la instanciación de variables compartidas y se sincronizan mediante la espera de variables no instanciadas.

2.2.2 Especificación formal Z

Utilizan notación matemática para describir en forma precisa las propiedades de un sistema obteniendo un modelo del sistema independiente de cualquier lenguaje de programación, contrario a lo que sucede con las técnicas LLCs.

Una especificación Z, está compuesta de módulos denominados esquemas, utilizados para describir aspectos estáticos y dinámicos del sistema. Aspectos estáticos como:

- Posibles estados del sistema
- Invariantes que se deben mantener cuando el sistema pase de un estado a otro.

Y aspectos dinámicos como:

---

[53] Siempre y cuando todos los tipos de lectores manejen adecuadamente el rigor de la notación formal.

[54] Algunos lenguajes pueden ser más apropiados para los usuario de especificación que para otro lectores.

[55] El gran beneficio frecuentemente viene dado por el proceso de formalización más que el resultado final.

[56] Llamadas pruebas obligatorias de demostración.

[57] La comprobación es el proceso de mostrar que un sistema satisface su especificación.

[58] Se pueden usar para generar casos de prueba de caja negra.

- Las posibles operaciones.
- Las relaciones entre las entradas y las salidas y
- Los cambios de estado.

2.2.3 Lenguaje de especificaciones formales orientado a objetos VDM++[59]

VDM++ es una extensión de VDM (Método de Desarrollo de Viena) pero ofreciendo clases, objetos y herencia; con características adicionales en formalismos para especificar permisos de invocación a secuencias de métodos y control de herencia.

VDM SL es el llamado lenguaje base para VDM, y en éste, una especificación contiene una simple definición de estados, tipos y variables. Para especificaciones de gran tamaño, estos estados contienen un gran número de estructuras de datos complejas, las cuales debido a la imposibilidad de partición, son difíciles de escribir y de entender. VDM++ nació de la inquietud sobre la falta de modularidad y de soporte para el desarrollo evolutivo con creciente funcionalidad.

2.2.3.1 Clases objetos y estados

Una clase describe el posible conjunto de propiedades y de comportamientos de un conjunto de objetos pertenecientes a esta.

Un objeto es considerado la encarnación de una clase. Los objetos se comunican con los demás mediante el envío de mensajes.

Una clase se considera un objeto derivado de alguna metaclase. Esta afirmación forzó a considerar la posibilidad de cambiar el estado del objeto representado en una clase y en consecuencia de esto guía el cambio en el comportamiento de estos objetos derivados.

El estado de un objeto está compuesto por sus variables de instancia con sus respectivos valores[60].

El estado general de un sistema[61], está definido como el ambiente en el cual el programa[62] es ejecutado. Este estado, consiste de alguna composición de estados de todos los objetos existentes.

BIBLIOTECA UIS

2.2.3.2 El espacio de trabajo

La descripción de clases puede ser vista como la parte estática de la especificación del sistema global, mientras que el espacio de trabajo comprende los elementos dinámicos del sistema.

El espacio de trabajo es la unidad en el cual una seudo - ejecución de una especificación podría tomar lugar. La introducción de un espacio de trabajo, crea la posibilidad para el diseño y la implementación interactiva e iterativa.

El desarrollo de un espacio de trabajo en particular, puede comenzar con la declaración de variables temporales y la creación de uno o más objetos pertenecientes a las clases disponibles.

Dado un conjunto de clases se puede crear una gran variedad de sistemas, dependiendo del modo como son creados los objetos de esa clase y del modo y orden en que ellos se comunican con los demás. Como consecuencia de este enfoque, un conjunto dado de clases puede ser descrito y utilizado, en diferentes espacios de trabajo. Cada espacio de trabajo representa el comportamiento dinámico de una historia personificada de las clases y en consecuencia un sistema individual.

En VDM++, existe la necesidad de introducir una estructura separada del espacio de trabajo como una clase ordinaria con un simple método, el método inicial[63].

2.2.3.3 Estructura semántica de VDM++

Una especificación de un sistema en VDM++ se define como:

Especificación del sistema ::=

Lista de descripciones de clases,

Descripción Opcional de Espacio de trabajo.

---

[59] Los comentarios respecto a las especificaciones formales en esta sección fueron tomados de [4]

[60] Los valores son entre otros objetos, valores propios de VDM o una combinación de ambos.

[61] Dentro de una especificación.

[62] Visto como una serie de expresiones

[63] Una seudo - ejecución de una especificación iniciaría con este método (el inicial).

BIBLIOTECA UIS

7791

Una descripción de una clase contiene un identificador, una especificación de variables de instancia y métodos; en el caso de herencia se especifica una superclase. Los nuevos elementos, en VDM++, son el control sobre la herencia y sobre la secuencia de métodos (trace)

Descripción de clase ::=

CLASS *Identificador_de_clase,*

Identificador opcional de superclase,

Sección de variables de instancia,

Lista de métodos,

Sección opcional control de herencia de métodos

Sección opcional de secuencias (Trace)

END *Identificador_de_clase.*

Una clase puede ser especificada como subclase inicial de una superclase. El identificador de superclase se lee ES SUBCLASE DE *Identificador_de_clase.*

Una o más variables de instancia, pueden ser forzadas a contener valores de alguno de los tipos básicos de VDM, un valor de una clase existente o un tipo construido[64]. En las variables de instancia se permiten expresiones constantes.

La lista de métodos permite la definición de métodos para ser descritos como otra especificación completa, usando pre y poscondiciones (no definidas hasta ahora) o como responsabilidad de la superclase o de la subclase.

Una especificación completa de un método contiene

- Un Identificador con tipos de información como parámetros
- Estado de referencia

---

[64] Usando tipos básicos y clases definidas.

900161

- Variables de instancia temporales
- Pre y poscondiciones.

Identificador_de_metodo (*parametro_1*: tipo, *parametro_2*:tipo....)

Pre_condiciones: [expresiones booleanas que describen los prerequisitos para la ejecución del método]

Pos_condiciones: [expresiones booleanas que describen los resultados deseados al ejecutar el método]

Una especificación de un método contiene la especificación de una poscondición que define la operación a realizar. Una extensión permitida para la especificación de métodos es que puede contener expresiones que autoricen al método para crear objetos y para invocar métodos de otros objetos.

En este estilo de especificaciones hay dos posibles valores de retorno:

Precondición entonces *Poscondición* y ¬ *Precondición* entonces *parte excepcional.*

Es responsabilidad del objeto que llama el método asegurar que la precondición se cumpla. Siempre que la precondición no se cumpla, el contrato entre el objeto que llama y el llamado no es aplicado, dejando libertad para que el objeto llamado aborte la ejecución, para retornar un valor o promover una excepción. Pasada la excepción el control enfoca la responsabilidad sobre el objeto que llama. La poscondición también puede contener parte excepcional.

2.2.3.4 Creación de objetos e invocación de mensajes

Para cada una de las clases, implícitamente se define el método "new"[65].

Después de la evaluación de la expresión *Identificador_de_clase "!" "new"*, se crea un nuevo objeto de la clase *"Identificador_de_clase"* mencionada en el llamado al método.

La forma general de invocación de un mensaje es:

---

[65] Disponible para cada uno de los objetos pertenecientes a una misma clase.

BIBLIOTECA UIS

expresión"!"*nombre_de_mensaje(parámetros*).

También se permiten la invocación de cascadas de mensajes.

2.2.3.5 Herencia

Una característica clave en cualquier metodología o lenguaje orientado a objetos es la habilidad para usar alguna forma de herencia. puesto que esto permite una forma incremental de desarrollo orientado al rededor de la reutilización máxima.

2.2.3.5.1 Herencia estructural

La estructura compuesta, construida por la enumeración de las variables de instancia de un objeto, forma una representación del objeto y determinan la definición de su estado. Puesto que esta representación es completamente determinada por la definición de la clase. se puede asociar un tipo especial a cada clase definida. Estos tipos extendidos están compuestos por las variables de instancia definidas en la superclase y por variables de instancia adicionadas en la definición de la subclase.

Sintácticamente se denota una representación simple de herencia estructural para indicar la superclase en la definición de la subclase

IS SUBCLASS OF *Identificador_de_clase*

2.2.3.5.2 Herencia funcional

La herencia de métodos en un ambiente fuertemente tipificado puede llevar a conflictos con la noción de tipificados. Lo relevante de este problema puede ser evitado con la introducción del concepto de herencia controlada.

Para soportar la noción de herencia controlada. se introduce en la especificación de la clase. una cláusula de control. la cual contiene una numeración explícita de los métodos heredados de la superclase. La sintaxis para cada controlador de herencia dentro de la definición de la subclase es:

Clausula de control de herencia ::=

INHERIT [*lista_control_herencia*] + END INHERIT

lista_control_herencia ::=

FROM *nombre_de_clase lista_métodos* | ALL_SUPER

nombre_de_clase ::=

SUPER | *identificador_de_clase*

lista_métodos ::=

ALL| *nombre_de_método* ,[*nombre_de_métodos*] | []

2.2.3.6 Secuencias

El propósito de la parte de secuencias, en una definición de clase es especificar una restricción en el comportamiento dinámico de un objeto durante su existencia.

Una estructura de secuencias (Trace), define el conjunto de secuencias permitidas, de invocación de métodos de una clase para todos los objetos de la clase

Acceso de secuencia ::=

ACCESS TRACES+ *lista_de_subsecuencias* + FIN SECUENCIA

lista_de_subsecuencia ::=

SUBTRACE Identificador_de_subsecuencia + lista_de_secuencias + WITH + condiciones_de_secuencias + END SUBTRACE

lista_de_secuencias ::=

nombre_secuencia + " = " + "{" + lista_de_metodos | nombre_subsecuencia + "}"

lista_de_métodos ::=

nombre_método | nombre_método + lista_de_métodos | []

Ejemplo

ACCES TRACES

Subtrace T7

a_T7 = {init, first, next }

t_T7 = { init; (first; next*)*}

With

t_T7 first=7 and t_T7 next >8:

BIBLIOTECA UIS

end subtrace

END ACCES TRACES

Una estructura de subsecuencia consiste en la especificación de una lista de nombres de métodos y una expresión de secuencia construida con el uso de operadores estándares de secuencia y algunas funciones sincronizadoras predefinidas. La traducción de una estructura de secuencias a una maquina de estados finitos es directa.

Puesto que los objetos son considerados como entidades pasivas, los métodos son invocados desde el flujo de control de un objeto invocador y por consiguiente una secuencia provee una alternativa fácil de usar para el registro histórico en precondiciones.

2.3 MODELADO DE SISTEMAS DE INFORMACIÓN

En el Anexo - Modelado de sistemas de Información - se presenta en detalle la descripción de técnicas para el modelado de sistemas de información. Esta sección se encarga de describir la relación y los conceptos básicos de dos técnicas que son columna vertebral del modelo de especificaciones propuesto en esta Tesis.

En el Modelo Entidad Relación Extendido[66], las entidades definen una estructura estática de información sobre las cuales se establecen relaciones de asociación o de herencia[67]. Para la construcción de estructuras más complejas, se define la operación de agregación[68] [69].

Toda entidad, ya sea simple o compleja, refleja un comportamiento dinámico, que pude ser descrito mediante diagramas de estados[70], estos estados describe la entidad en determinado instante, mediante el conjunto de valores asignados a cada uno de sus atributos.

---

[66] Las características Extendidas del Modelo Entidad Relación son la definición de entidades fuertes y débiles, relaciones de herencia, y agregaciones (ver anexos).

[67] Las relaciones de herencia establecen que una entidad hija adquiere toda la estructura de una entidad padre junto con las relaciones asociadas a dicha entidad.

[68] La operación de agregación consiste en crear entidades cuyos atributos también son entidades o algún tipo de datos básico

[69] Las estructuras de agregación, se ven como relaciones de alto grado (grado mayor de dos).

[70] Ver definiciones básicas de Diagramas de Estados, en los anexos.

BIBLIOTECA UIS

El cambio en los valores de dichos atributos implica el cambio de estado, por lo que se presenta una transición de estado, disparada por una condición o evento ocurrido en el ambiente del sistema.

Cada transición es una asociación entre dos estados[71], a la cual se le asocia una condición necesaria y suficiente para que se haga efectiva. A dicha condición se le conoce con el nombre de precondición, y el cumplimiento de dicha precondición en determinado instante, hace que se genere un evento.

Para una entidad en un sistema de información, se pueden definir infinitos[72] estados y por esto es tarea del analista de sistemas, identificar los estados relevantes de cada una de las entidades, y definir las posibles transacciones entre dichos estados.

Las transiciones determinan el paralelismo o la secuencialidad de los estados. (ver Figura 4).

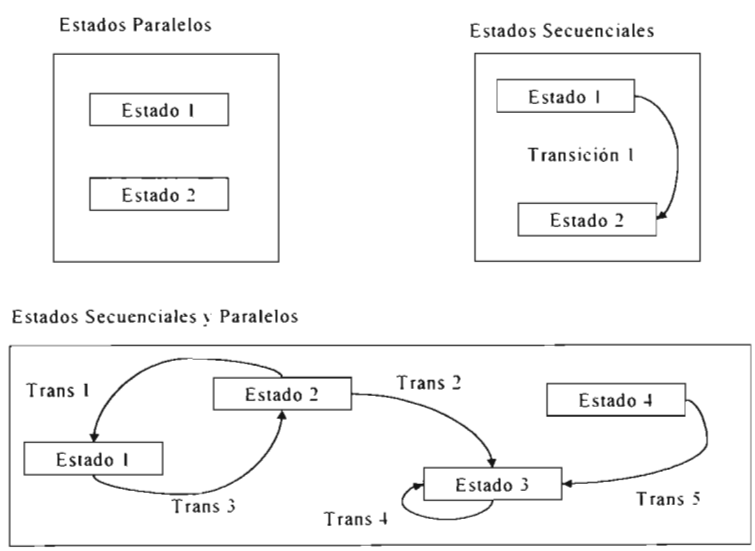

Figura 4 Representación gráfica de Estados secuenciales y paralelos

Para la representación de estados complejos se utiliza el concepto de superestados[73], con los cuales las transiciones pueden modelarse entre superestados, o entre estados y superestados.

---

[71] Un estado origen y otro destino

[72] Sin embargo no todos estos estados son relevantes

[73] Un superestado se ve como un estado compuesto de otros estados.

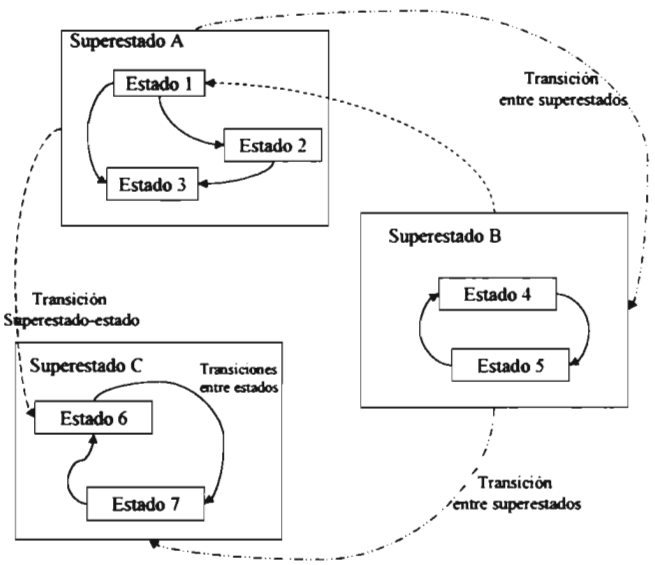

Figura 5 Representación gráfica de superestados y sus transiciones

BIBLIOTECA UIS

3. ESPECIFICACIÓN DE LA HERRAMIENTA SOFTWARE

3.1 INTRODUCCIÓN

ESFOSIN tiene por objetivo, asistir al analista en la etapa de especificaciones de sistemas de información. Su interfaz gráfica permite al analista describir la estructura del sistema y su comportamiento, para obtener de forma automática una especificación textual formal en VDM++.

ESFOSIN presenta una interfaz gráfica que interactúa[74] con el usuario para la representación de cada modelo y de la documentación.

3.2 ESTRUCTURA GENERAL DE LA HERRAMIENTA

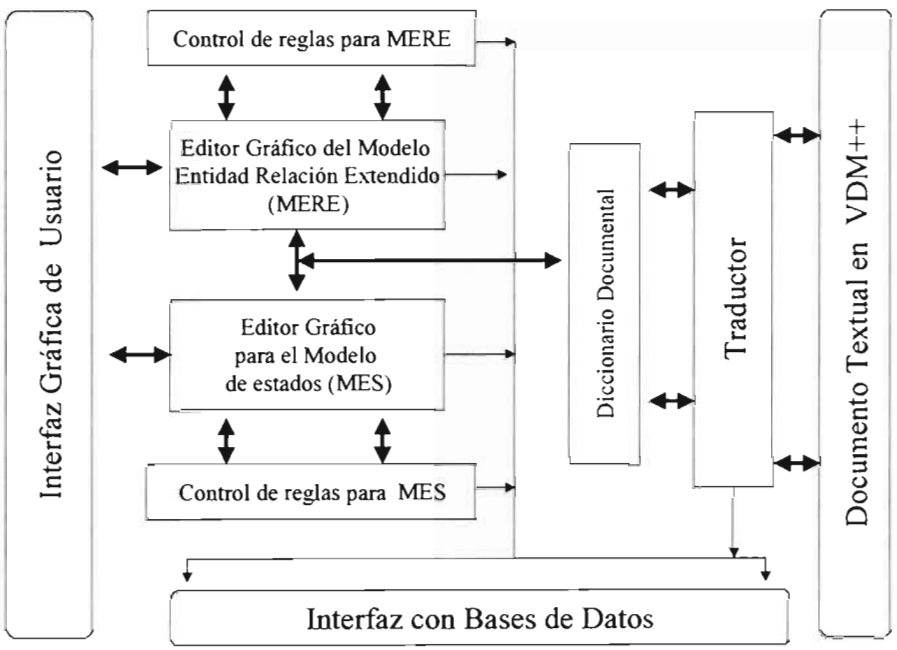

Figura 6 Estructura General de la Herramienta propuesta

[74] Controlando que se cumplan cada una de las reglas que regulan el modelo entidad relación y las máquinas de estado.

3.3 DESCRIPCIÓN DE LA INFORMACIÓN

ESFOSIN manipula, organiza y administra la información proveniente de los modelos estructural y de comportamiento de los sistemas de información.

ESFOSIN, posee una estructura de datos, que contienen información[75] sobre los datos manipulados por el sistema de información que se desea especificar. El siguiente diagrama entidad describe la estructura de metadatos.

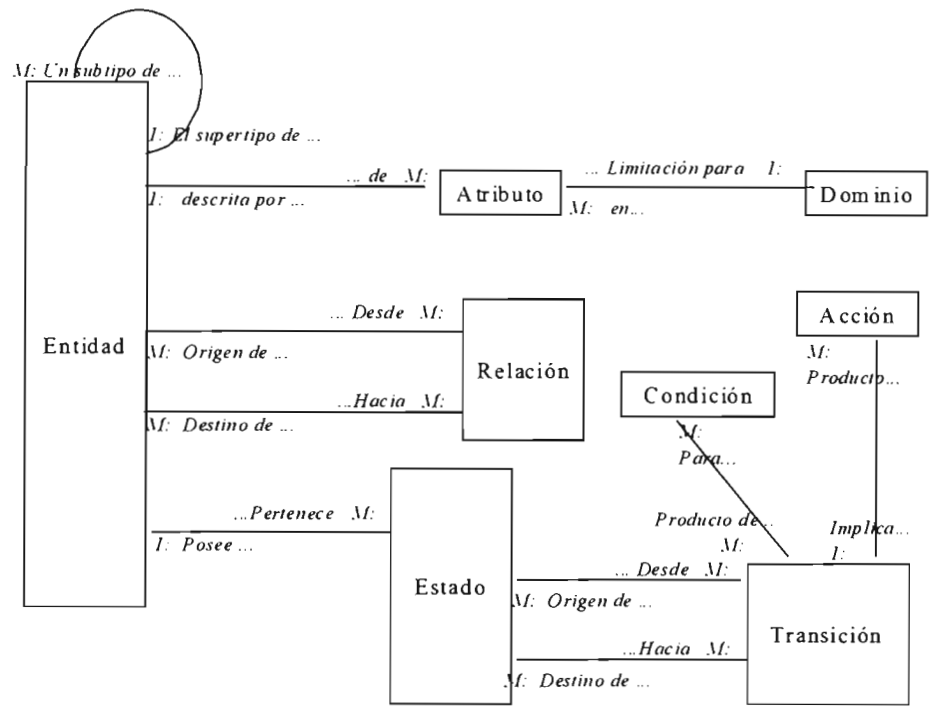

Figura 7 Modelo Entidad Relación para ESFOSIN

Los atributos para las entidades definidas en el anterior modelo son:

Entidad	Atributos
Atributo	Nombre, tipo, valor inicial, valor máximo, valor mínimo
Dominio	Nombre, tipo, rango
Relación	Nombre, cardinalidad, opcionalidad
Entidad	Nombre, tipo, definición, comentario, subtipo, volumen máximo, restricciones generales.
Estado	Nombre, definición, descripción

[75] A esta información sobre los datos, "datos de los datos", comúnmente se le conocen como metadatos.

Transición	Nombre, condición, acción, entrada.
Precondición	Descripción
Poscondición	Descripción
Comentario	Descripción.
Definición	Termino, comentario.
Función	Nombre, tipo, parámetros, definición, comentarios

Tabla 3 Descripción de entidades para ESFOSIN

La siguiente figura describe los procesos[76], realizados por ESFOSIN de acuerdo a la representación de diagramas de flujo de datos.

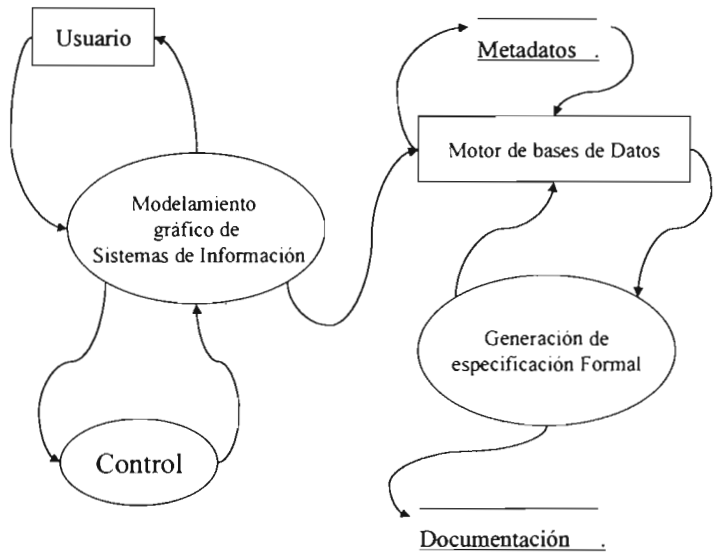

Figura 8 Diagrama de flujo de datos de primer nivel para ESFOSIN.

3.3.1 Descripción de la Interfaz del sistema

ESFOSIN posee dos interfaces diferentes: con un motor de bases de datos[77] y con el usuario[78]. Opcionalmente puede tener una tercera interfaz con un procesador de palabras[79]

---

[76] A nivel macro.

[77] El motor de base de datos contiene los metadatos del sistema de información. correspondientes a la descripción gráfica.

[78] Para la definición de las componentes estructurales y de comportamiento del sistema de información propuesto.

[79] Con el fin de exportar la documentación y los datos textuales y gráficos. generados por ESFOSIN.

3.4 DESCRIPCIÓN FUNCIONAL

3.4.1 Partición funcional

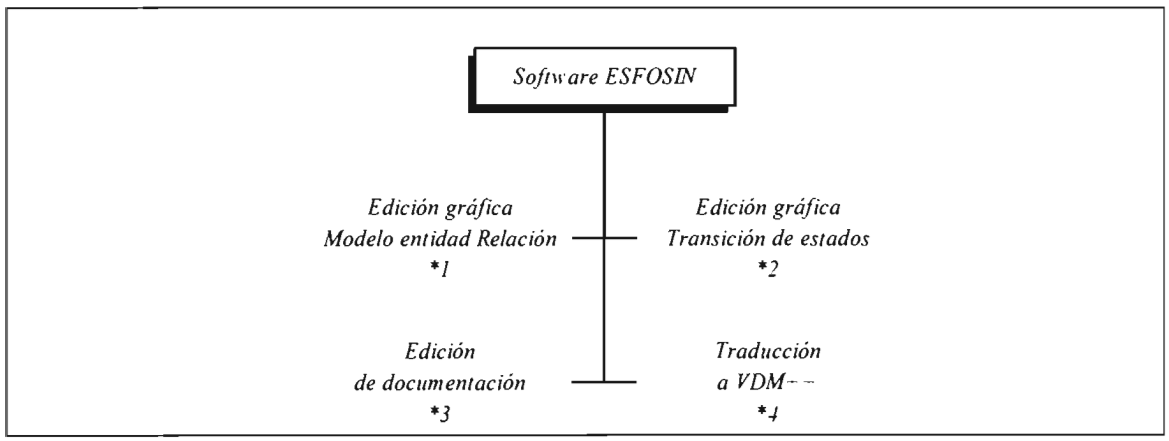

Figura 9 Partición funcional del software ESFOSIN

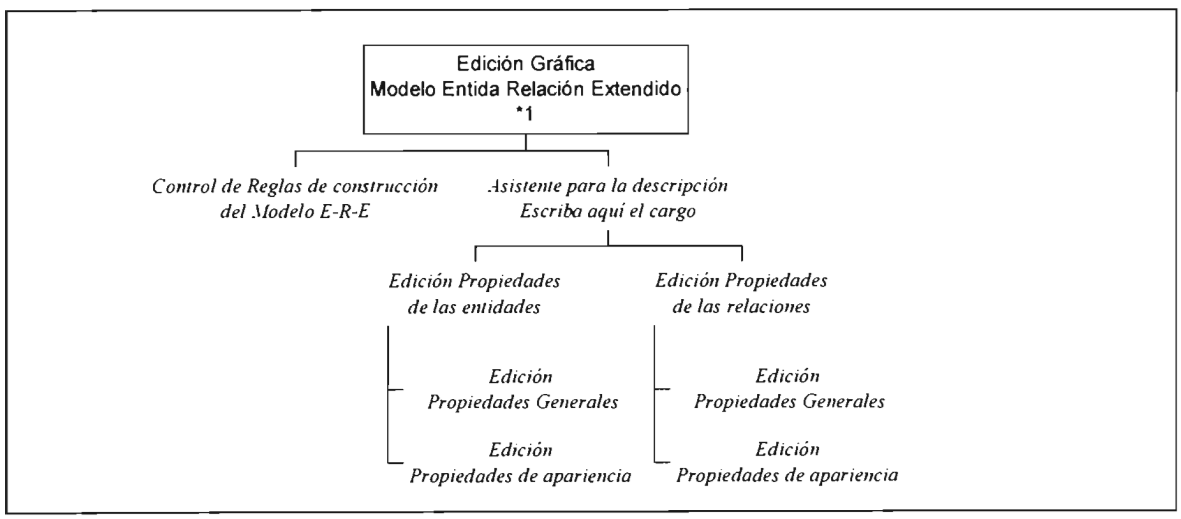

Figura 10 Partición funcional de la edición gráfica del MERE

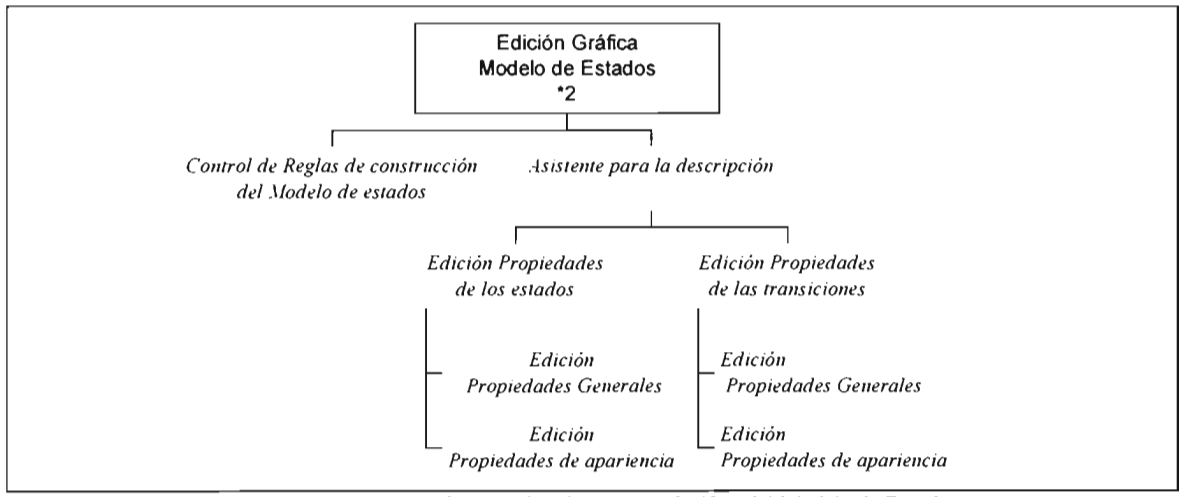

Figura 11 Partición funcional de la Edición Gráfica del Modelo de Estados

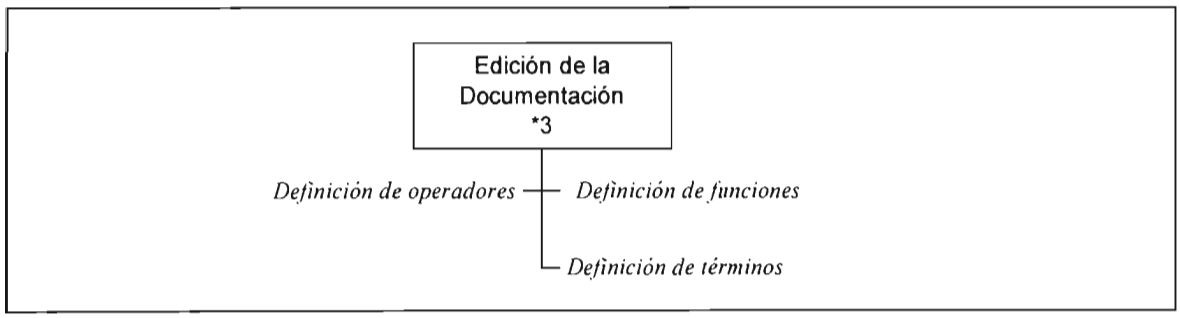

Figura 12 Partición funcional de la edición de Documentación

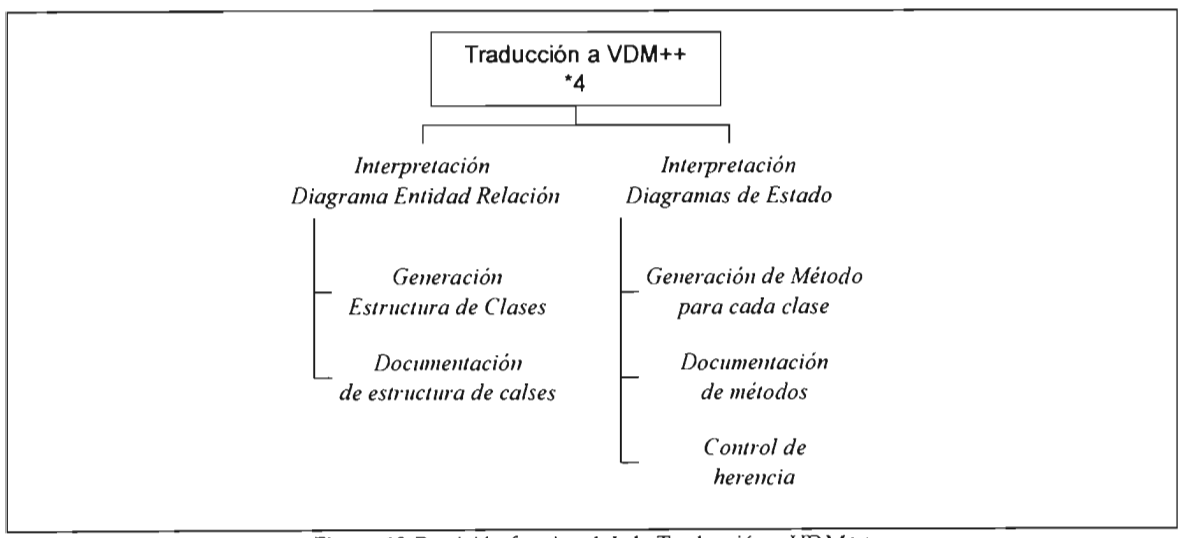

Figura 13 Partición funcional de la Traducción a VDM++

BIBLIOTECA UIS

3.4.2 Descripción funcional

3.4.2.1 Edición Gráfica del modelo entidad relación

Proceso mediante el cual se edita el modelo entidad relación[80] del sistema de información. La descripción de cada una de las componentes, representadas como objetos gráficos[81], se realiza mediante asistentes.

3.4.2.1.1 Reglas de control en la construcción del modelo Entidad relación extendido

Controla la construcción del modelo estructural, mediante un conjunto de reglas que definen las prohibiciones y recomendaciones que regulan el modelado entidad relación extendido.

Estas reglas son:

- Cada tipo de entidad debe poseer un único nombre diferente a los demás tipos de entidades.
- Cada entidad debe poseer por lo menos dos atributos, de los cuales al menos uno es un identificador único (clave primaria).
- Cada uno de los atributos debe tener definido un tipo.
- No se deben permitir la construcción de relaciones no válidas tales como:
- Relación muchos a muchos, obligatorio en ambos extremos.
    - Relación recursiva muchos a uno, opcional en uno y obligatorio en muchos.
    - Relación recursiva muchos a uno, obligatorio en ambos extremos.
    - Relación recursiva muchos a uno, obligatorio en el extremo uno y opcional en el extremo muchos.
    - Relación recursiva uno a uno, obligatorio en un extremo y opcional en el otro.

---

[80] Con sus características extendidas

[81] De acuerdo a la notación expuesta en la sección, modelado de sistemas de información, de este documento (Ver Anexos).

BIBLIOTECA UIS

- Relación recursiva uno a uno, obligatorio en ambos extremos
- Relación recursiva muchos a muchos, obligatoria en ambos extremos.
- Relación recursiva muchos a muchos, obligatorio en un extremo y opcional en el otro.

- Se debe advertir la creación de relaciones consideradas como casos extraños, pero válidas tales como:
  - Relación uno a uno opcional en un extremo y obligatorio en el otro.
  - Relación uno a uno opcional en ambos extremos.
  - Relación uno a uno obligatorio en ambos extremos.
  - Relación muchos a muchos, obligatoria en un extremo y opcional en el otro.

3.4.2.1.2 Asistencia para la descripción

Proceso mediante el cual se le asignan las características a cada uno de los objetos utilizados en el modelado estructural del sistema de información.

3.4.2.1.2.1 Edición de propiedades de las entidades

Herramienta mediante la cual, se describen las características propias de cada una de las entidades, al igual que las características correspondientes a la apariencia gráfica de cada objeto entidad.

Mediante la edición propiedades generales, se dan las características básicas de las entidades, tales como:

- Nombre representativo válido.
- Atributos básicos[82].
- Clasificación (fuerte o débil).

---

[82] Clasificados como atributo de clave primaria (cuando son características esenciales de la entidad), como atributo descriptivo complementario (cuando representan característica accidentales). También se dispondrá de un listado de atributos que se pueden asignar a cada entidad.

BIBLIOTECA UIS

- Comentarios.
- Definiciones asociadas.
- Restricciones de cardinalidad[83].
- Restricciones generales[84].

Cada atributo debe ser descrito mediante:

- Un nombre representativo válido
- Un tipo (numérico, alfanumérico, booleano, fechas)
- Valores iniciales o por defecto
- Rango de valores válidos
- Comentario
- Definiciones asociadas

Por otro lado la edición de propiedades de apariencia[85], da formato al aspecto[86] de la entidad, dentro del gráfico.

3.4.2.1.2.2 Edición de propiedades de relaciones

Herramienta mediante la cual se otorgan características básicas y de apariencia de todas y cada una de las relaciones representadas en diagrama entidad relación extendido.

Las propiedades generales describen las características básicas de las relaciones son:

- Nombre representativo válido.

---

[83] Esta restricción hace referencia a la cantidad máxima y mínima, de instancias de una entidad que pueden pertenecer a dicha clase.

[84] Las restricciones generales son enunciados expresados formalmente o en lenguaje natural, que determinan las condiciones para que una entidad pueda o no pertenecer a un determinado tipo o clase de entidad.

[85] Esto solo es una ayuda didáctica para el analista, que le permite diferenciar los componentes, mediante colores y estilos.

[86] Color de letra, color de fondo, color de borde, tamaño de la entidad, tipo de letra, posición

BIBLIOTECA UIS

- Cardinalidad[87].
- Características de obligatoriedad[88].
- Comentarios.
- Definiciones asociadas.
- Papeles desempeñados por cada uno de los extremos[89].
- Restricciones de participación[90]
- Restricciones generales[91].

Las relaciones de herencia[92] [93] son consideradas de forma especial. Se describen mediante un comentario general, el control de herencia en los atributos[94], el control de herencia de los métodos[95], las restricciones propias de este tipo de relación (unión[96], mutua exclusión[97], partición[98] e intersección[99]) y mediante restricciones generales.

---

87 En una relación entre la entidad "A" y la entidad "B", la cardinalidad referencia al número de entidades del tipo "A" que están relacionadas con la entidad "B". (una a una, una a muchas, mucha a muchas)

88 Establece la necesidad de que una entidad deba estar o no relacionada con otra.

89 Los papeles expresan la función que desempeña cada entidad, dentro de una relación. Por ejemplo, en la relación A es padre de B, la entidad A cumple el rol de papá y la entidad B cumple el rol de hijo.

90 Estas restricciones hacen referencia a la cantidad máxima y mínima de veces en las que puede participar una entidad en una relación (si la relación es uno a uno, este máximo y mínimo será uno).

91 Las limitaciones generales pueden ser informales, en lenguaje natural, o pueden ser totalmente formalizadas como expresiones algebraicas o expresiones de la lógica de primer orden.

92 Elemento esencial del paradigma orientado a objeto.

93 Dentro de una relación de herencia, "A" es un "B", "A" cumple el papel de entidad especializada y "B" de entidad generalizada.

94 Permite seleccionar atributos particulares, que no hagan parte de la herencia. (esto es útil cuando existe la posibilidad de herencia múltiple).

95 Permite seleccionar los métodos (obtenidos por medio de la transición de estados), que una entidad especializada, no hereda de la entidad generalizada.

96 Las restricciones de unión indica que cada entidad en un tipo de entidades generalizada, debe también ser un miembro de por lo menos un tipo de entidad especializada.

97 Indica que los conjuntos de entidades especializadas, son disyuntos.

98 Se define esta restricción, cuando en una relación de herencia, se presenta al mismo tiempo las restricciones de unión y de mutua exclusión.

99 Estas restricciones se utilizan cuando un conjunto de entidades generalizadas, se obtienen de la intersección de los conjuntos de entidades especializadas.

BIBLIOTECA UIS

3.4.2.2 Edición Gráfica del diagrama de transiciones de estado

Encargada de dar soporte al diseño gráfico, para la construcción de los diagramas de estados para cada una de las entidades definidas en el gráfico representativo del modelo entidad relación extendido[100].

3.4.2.2.1 Reglas de control para la construcción del diagrama de transiciones de estado

El conjunto de reglas que definen las prohibiciones y recomendaciones reguladoras del modelado mediante diagramas de transición de estados, está determinado por:

- Todo estado debe poseer un nombre válido.
- No debe existir un estado tal que no tenga por lo menos una transición asociada a él.
- Solo debe existir uno y solo un estado inicial
- Toda transición debe poseer un estado origen y un estado destino plenamente identificados
- La destrucción de un estado, implica la destrucción de las transiciones asociadas a este estado.

3.4.2.2.2 Asistencia para la descripción

Interfaz para la descripción de cada uno de los estados y de las transiciones involucradas en el modelo de comportamiento, para cada una de las entidades.

Las propiedades de los estados, definen el comportamiento del objeto (entidad) a través de su ciclo de vida. Cada estado se describe mediante:

- Nombre válido
- Tipo de estado (inicial o no inicial)
- Comentarios
- Definiciones

---

[100] Tanto bajo como alto nivel (entidades complejas).

BIBLIOTECA UIS

- Restricciones de tiempo real[101] y restricciones generales.
- Valores de estado[102]

La edición de las propiedades de las transiciones, se realiza por medio de un asistente para asignación de valores a cada una de las características básicas y de apariencia:

- Nombre válido
- Comentarios
- Definiciones
- Expresión booleana que involucra los atributos de la entidad a la cual se le esta asociando un diagrama de transición de estados y que indica que si se puede pasar de un estado a otro (precondición)
- Expresión booleana que indica la acción de salida provocada por el cumplimiento de las condiciones para la transición de un estado a otro (poscondición)
- Restricciones de tiempo[103] y restricciones generales.
- Datos de entrada y de salida involucrados en la transición.
- Eventos necesarios para hacer efectiva la transición (disparadores).
- Acciones durante la transición.

3.4.2.3 Edición de documentación

Herramienta encargada de recopilar las diferentes definiciones y comentarios asociados durante la asistencia en el proceso de definición del modelo estructural y la representación de comportamientos de cada componente.

3.4.2.3.1 Edición de operadores

---

[101] Representa el tiempo máximo que una entidad puede permanecer en dicho estado.

[102] A cada uno de los atributos asociados a la entidad que se le define el estado, se le asigna un valor o una condición que deba cumplir dicho valor. Estos valores son los representativos del estado.

[103] Representa la cantidad máxima y mínima de tiempo que puede transcurrir durante la transición.

Se define un símbolo como un operador, especificando el tipo de operando involucrado o involucrados en dicha definición. Por defecto se definen tres tipos de operandos básicos: booleanos, aritméticos, relacionales.

Booleanos:

and	representado por "and" involucra dos valores de verdad
or	representado por " or " involucra dos valores de verdad
not	representado por " not" involucra un valor de verdad

Relacionales:

=	Igualdad, involucra dos expresiones numéricas o booleanas
<	menor
>	mayor
>=	mayor o igual
<=	menor o igual
!	Diferente

Operadores aritméticos:

+	suma
-	resta
/	división
*	producto

Cualquier otros operador, es definido por el usuario y constan de las siguientes propiedades:

- Nombre representativo y válido.
- Símbolo representativo para la operación.
- Tipos de operando que involucra.
- Tipo de resultado que da como salida.

3.4.2.3.2 Definición de términos

Diccionario de términos que fueron utilizados en los comentarios, durante el transcurso de las definiciones de las propiedades de los diferentes objetos involucrados en el modelado del sistema de información.

Cada definición consta de un término palabra o frase y de un significado descrito en lenguaje natural. Cada término solo puede tener un significado.

3.4.2.3.3 Definición de funciones

Diccionario de funciones que se utilizan en la generación de expresiones dentro de la descripción de cada uno de los objetos involucrados en el modelo.

Una función esta definida mediante un nombre, un dominio[104] y un tipo de datos de salida[105], además de una descripción en lenguaje natural, del proceso que desarrolla.

3.4.2.4 Traducción a VDM++

Contiene el conjunto de reglas que hacen la correspondencia entre los objetos del modelado del sistema de información y la sintaxis textual de VDM++. La traducción es una función cuyo dominio es el conjunto de todos y cada uno de los objetos gráficos que modelan el sistema de información y cuyo codominio es el conjunto de las expresiones sintácticas y semánticas de VDM++.

3.4.2.4.1 Interpretación del diagrama entidad relación extendido

Para mayor comprensión de las reglas de transformación, se presenta el siguiente ejemplo de modelado[106]:

---

[104] Puede ser un tipo de datos básico una entidad o un producto cartesiano de dichos tipos.

[105] O tipo de datos de condominio

[106] Tan solo una pequeña parte de un sistema de información más complejo. Representa algunas entidades y los posibles estados, de una de ellas.

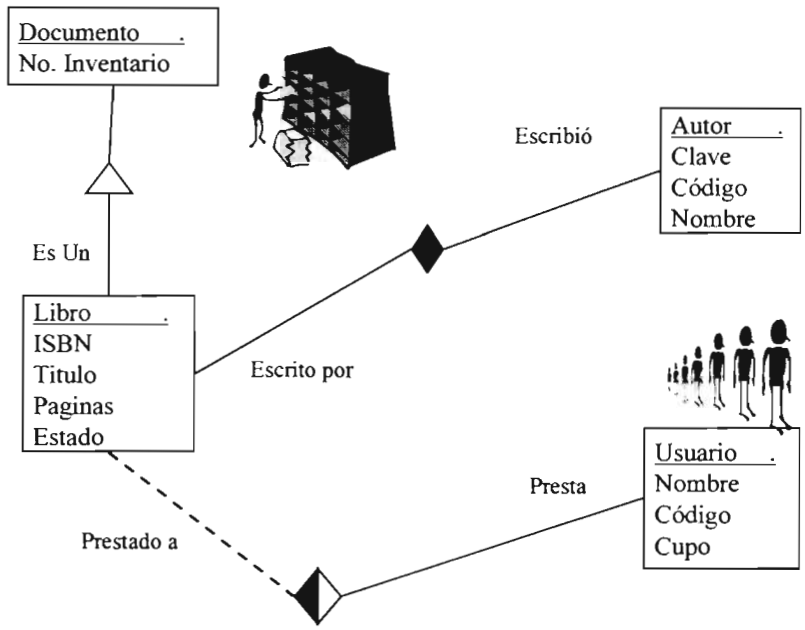

Figura 14 Parte del diagrama Entidad Relación para un SI de una Biblioteca (Ejemplo)

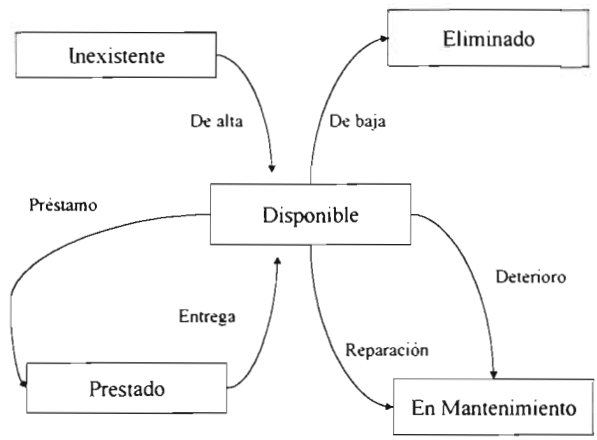

Figura 15 Diagrama de Estados para la entidad LIBRO (Ejemplo)

3.4.2.4.1.1 Generación de la estructura de las clases

Las reglas de conversión del diagrama entidad relación. a la sintaxis de VDM++, generan la estructura de clases. Esta estructura está determinada por:

CLASS *Identificador_de_clase*

IS SUBCLASS OF *Identificador_de_superclase*

INSTANCE VARIABLES

V1: Tipo

V2: Tipo

METHODLIST

M1

M2

INHERIT

.....

END INHERIT

ACCESS TRACES

....

END ACCESS TRACES

END CLASS *Nombre_de_clase*

La descripción de la entidad, solo otorga información para la sección de variables de instancia, para los encabezados de la clase y para el método New de cada clase.

- Cada entidad se transforma en una clase cuyo nombre de clase es el nombre de la entidad, anteponiendo la letra "T"[107].

- Cada atributo de le entidad se transforma en una propiedad de la clase, listada en la sección de variables de instancia.

- Cada relación se traduce de acuerdo a la cardinalidad, así:

    - Si una relación es uno a muchos: la entidad del extremo uno, es agregada como una variable de instancia a la clase generada por la entidad del extremo muchos, especificando el tipo de entidad.

---

[107] Esta convención permite diferenciar las Clases o Tipos de objetos, de otros datos simples. No es una notación propia de VDM++.

BIBLIOTECA UIS

- Si una relación es uno a uno: cada una de las entidades es agregada a la otra entidad, dentro del listado de variables de instancia.

- Si la relación es mucho a muchos se crea una nueva clase cuyo nombre esta formado por los identificadores de los tipos de entidades involucradas. Esta clase se especifica mediante dos variables de instancia correspondientes a referencias de las clases involucradas en la relación.

- Las relaciones obligatorias se traducen en comentarios dentro del método new. El comentario tendrá la siguiente forma: */ la existencia de un objeto del tipo XXX depende de la existencia de un objeto de tipo YYY /*. La clase XXX corresponde a la generada por la entidad en cuestión y la clase YYY es la relacionada con XXX.

- Para una Entidad especializada en una relación de herencia, se convierte en parte de la sentencia IS SUBCLASS OF *Identificador_de_superclase*. Donde *Identificador_de_superclase* es el nombre de la clase generalizada en la relación.

- Las restricciones asociadas a la relación de herencia, se traducen en comentarios, escritos junto a la sentencia IS SUBCLASS OF.

- Las definiciones en el control de herencia de atributos se traducen en los atributos permitidos para la clase especializada.

Las definiciones en el control de herencia de métodos se traduce en parte de la sección INHERIT

Lista de métodos ... END INHERIT.

- Las restricciones de la entidad, se traducen en el modelo de comportamiento, como un prerequisito dentro del método New[108].

- De igual forma que las entidades simples, las entidades de alto nivel se convierten en clases, en las cuales sus atributos son asociaciones de entidades de bajo nivel (simples) y/o atributos básicos.

---

[108] Este método es creado e incluido por defecto, para todas y cada una de las clases en el código VDM++.

- Un espacio de opcional de trabajo, se define como cualquier otra clase, con un único método (New) en el cual se hace el llamado al método new[109] de alguna de las clases definidas.

Por lo tanto la entidad LIBRO genera la siguiente estructura de clase:

CLASS *TLibro*

IS SUBCLASS OF

INSTANCE VARIABLES

No_Inventario*: Numerico;*

Isbn*: Alfanumerico;*

Titulo*:Alfanumerico;*

No_de_Páginas*:Numérico;*

Usuario*: TUsuario;*

METHODLIST

NEW(parámetros especificados mediante los asistentes)

/* Pendientes para el diagrama de estados */

INHERIT

/* Pendientes para el diagrama de estados */

END INHERIT

ACCESS TRACES

/* Pendientes para el diagrama de estados */

END ACCESS TRACES

END CLASS *LIBRO*

CLASS *LIBRO_AUTOR*

IS SUBCLASS OF

---

[109] Normalmente este método corresponde a una clase generada por una entidad de alto nivel.

INSTANCE VARIABLES

Libro: *TLibro;*

*A*utor: *TAutor;*

END CLASS *LIBRO_AUTOR*

3.4.2.4.1.2 Documentación de clases

Cada comentario o definición asociada a la entidad en el diagrama entidad relación, se convierten en comentarios de la clase. Este comentario debe ser anexado al inicio de la definición de la clase, limitado en los extremos por los símbolos de comentarios /**comentario* */.

3.4.2.4.2 Interpretación del Diagrama de estados

Esta componente define las reglas de transformación correspondientes a los diagramas de estados asociado a cada una de las entidades definidas en el modelo estructural.

Para cada entidad, convertida en clase dentro del texto de la especificación, se define los métodos de la siguiente forma:

- Cada transición de estado representa un método de la clase generada por la entidad a la cual se le está asociando el diagrama de estados.
- El nombre del método será el mismo asignado a la transición.
- La condición[110] se convierte en la precondición para el método.
- La acción asociada a cada transición se convierte en comentario dentro de la definición de precondiciones y poscondiciones.
- Las características del estado origen de la transición, se convierten en expresiones adicionales de las precondiciones.
- Las características[111] del estado destino de la transición, se convierten en poscondiciones para el método.

- Los comentarios asociados a cada transición son convertidos en comentarios del método generado, y se definen junto al nombre del método, agrupado en los extremos por los símbolos de comentario /* Comentario */.

- Los nombres de los estados se utilizan como comentarios al inicio de la definición del método así: */* ESTADO1-ESTADO2*/*

- Los comentarios asociados a los estados origen y destino de la transición, se traducen en comentarios para las precondiciones y poscondiciones respectivamente.

- Los datos de entrada y de salida, asociados a la transición, se convierten en parámetros para el método generado.

- Los eventos especiales de las transiciones se convierten en expresiones lógicas adicionadas a las precondiciones del método generado, mediante un conectivo AND.

Por lo tanto, la definición de los métodos, generados por el diagrama de transición de estados para la entidad LIBRO (del ejemplo que nos ocupa), es:

CLASS *TLibro*

IS SUBCLASS OF

INSTANCE VARIABLES

/* Definidas en la interpretación de entidades */

METHODLIST

NEW(Parámetros)

/* método de creación de una instancia */

DE_ALTA (parámetros especificados mediante los asistentes)

/* inexistente - disponible */

Pre-requisitos

---

[110] Expresión booleana asociada a la transición de estados.

[111] Valores de estados más restricciones.

```
                /*expresión booleana que asociada a la condición de la transición */
                Pos-requisitos
                /*expresión booleana que asociada a la acción de la transición */
        PRESTAMO( parametros necesarios)
        /* disponible - prestado */
                Pre-requisitos
                /*expresión */
                Pos-requisitos
                /*expresión */
        ENTREGA( parametros necesarios)
        /* prestado - disponible */
                Pre-requisitos
                /*expresión */
                Pos-requisitos
                /*expresión */
    INHERIT
        /* Generado por los asistentes */
    END INHERIT
    ACCESS TRACES
        T1={ Préstamo. Entrega}
        T2={Reparación. Entrega _ reparación}
        T3={De_alta.T1}
        T4={De_alta.T2}
        T5={De_alta.T1|T2.de_baja}
    END ACCESS TRACES
END CLASS LIBRO
```

BIBLIOTECA UIS

3.4.3 Descripción de control

ESFOSIN es un software dirigidos por eventos que permite crear gráficamente los modelos estructural y de comportamiento, por lo tanto el control debe estar enfocado a el cumplimiento de las reglas de construcción de dichos diagramas.

El control de integridad de los datos es manejado por el motor de base de datos, en el cual se almacenan los metadatos correspondiente a la descripción gráfica. La actualización de la base de datos es en línea con respecto a las modificaciones realizadas en la descripción gráfica.

3.5 DESCRIPCIÓN DEL COMPORTAMIENTO

3.5.1 Estados del sistema

ESFOSIN en determinado momento se encuentra en uno y solo uno de los siguientes estados:

- Creando un modelo Entidad Relación Extendido, dando propiedades a los diferentes objetos que describen este modelo.
- Creando el modelo de estados para una y solo una de las entidades definidas en el modelo estructural.
- Creando el documento textual de especificación formal de acuerdo los diagramas construidos.

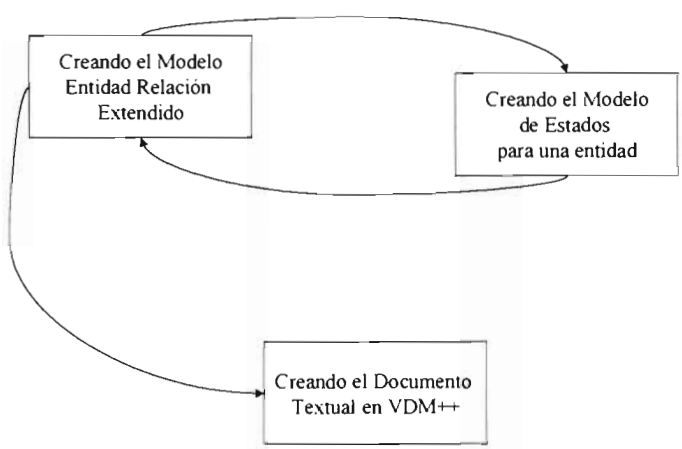

Figura 16 Estados Generales del sistema ESFOSIN

A su vez cada uno de estos estados, se ven como subsistemas con variados procesos.

3.5.2 Sucesos y acciones

Los sucesos relevantes para el sistema están determinados por las transiciones entre los estados descritos anteriormente.

Durante la creación de una entidad a la cual se le han asignado las propiedades correspondientes, se permite la construcción del diagrama de estados asociado a dicha entidad, lo cual implica la transición del estado creando MERE a el estado Creando ME. Esta transición y su inversa se presentan permanentemente durante la creación de cada entidad.

La transición, desde el estado creando MERE, al estado creando documento textual en VDM++ debe controlar el cumplimiento de las reglas de construcción planteado para cada uno de los modelos.

3.6 CRITERIOS DE VALIDACIÓN

3.6.1 Límites de rendimiento

ESFOSIN debe soportar la creación de Modelos Entidad Relación Extendido con por lo menos 70 entidades. Para cada una de las cuales se podrá crear un Modelo de estados con por lo menos 30 estados. Estos límites planteados solo tienen efectos para la validación del software, sin embargo ESFOSIN podrá soportar cualquier cantidad de objetos que permitan el modelado del sistema deseado. Los límites estarán determinados por los recursos Hardware con los que se disponga.

El tiempo de respuesta no es relevante para ESFOSIN, ya que no es un software que requiera el control de procesos en tiempo real, sin embargo un parámetro de comparación válido es: El tiempo de diseño gráfico debe ser menor o igual que el utilizado para representar el mismo modelo en una herramienta genérica de diseño gráfico.

3.6.2 Clases de pruebas

Las pruebas deben estar enfocadas a los aspectos:

- Apariencia gráfica de cada uno de los objetos componentes del modelo planteado.

- Control de las reglas de construcción para cada uno de los modelos involucrados.[112]
- Generación del Documento textual de la Especificación el cual debe estar acorde a la sintaxis y semántica de VDM++[113].
- Facilidad en la descripción de los componentes dentro de los modelos propuestos.

3.6.3 Consideraciones especiales

ESFOSIN requiere ambientes gráficos, y un equipo con las siguientes características mínimas: Computador Pentium de 100MHz, sistema operativo Windows 95, 16 MB en memoria RAM y 10 MB disponibles en Disco Duro.

ESFOSIN tiene una interfaz con un motor de base de datos. El prototipo requiere de ODBC de 32 bits, y los drivers ODBC de Ms Access 97 Database.

En el panel de control, dentro del ODBC 32, se debe configurar en el DSN de usuario y el DSN de archivo, un tipo TAccess, con el camino de la base de datos. c:\Esfosin\BDEsfosin.mdb

---

[112] Tanto para el MERE como para le Modelo de Estados.

[113] Descrita en la sección 2.2.3 de este documento.

4. ESPECIFICACIÓN DEL PROTOTIPO EN ESFOSIN

Esta sección presenta un ejemplo de parte del modelado del prototipo ESFOSIN, construido con la misma herramienta prototipo, la cual generó el código VDM++ que se adjunta a la figura:

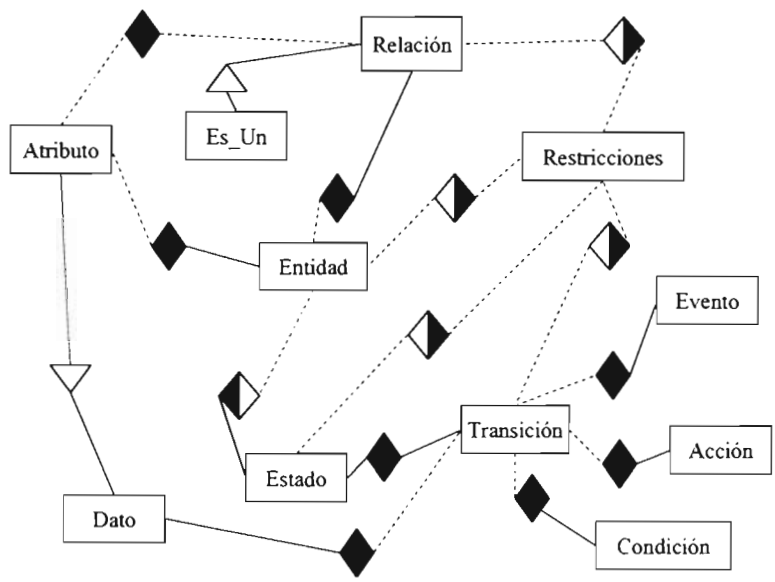

Figura 17 Modelo Entidad Relación Extendido para ESFOSIN

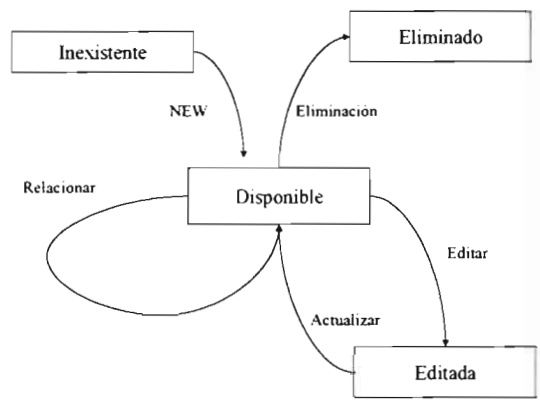

Figura 18 Diagrama de Estados para la entidad ENTIDAD

BIBLIOTECA UIS

Parte del código generado por el prototipo es:

```
CLASS  TEntidad
        IS SUBCLASS OF
        INSTANCE VARIABLES
                Nombre: Cadena */ Identificación de la entidad*/
                Tipo: Cadena  */fuerte o débil/*/
                Cardinaliad_Máxima: Numérico  */númerro máximo de instancias entidad*/
        METHODLIST
                NEW( )
                        */ Inexistente - Dsponible/*
                        Pre-requisitos: Número de instancias < Cardinalidad máxima
                        Pos-requisitos: Nommbre<> NULL
                EDITAR ( PNombre, PLista de Atributos, PListaRestricciones)
                        */ Disponible - Editada /*
                        Pre-requisitos: Evento de edición and Nombre<>NULL
                                */ asociar la lista de atributos a una instancia /*
                                */asociar cada elemento de Plista una instancia/*
                        Pos-requisitos: Nombre=Pnombre
                ACTUALIZAR (PLista de Atributos, PListaRestricciones)
                        */ Editadad - Disponible /*
                        Pre-requisitos: Evento de Actualización and Nombre<>NULL
                                */ asociar la lista de atributos a una instancia /*
                                */asociar cada elemento de Plista una instancia/*
                        Pos-requisitos: Nombre<>PNombre

                ELIMINAR ()
                        */ Disponible-Eliminado*/
                        Pre-requisitos: Evento de eliminación and Nombre<>NULL
                                */ Actualizar  /*
                        Pos-requisitos: Nombre=NULL
                RELACIONAR ( PRelacion)
                        */ Disponible-Disponible*/
                        Pre-requisitos: Evento de conección and Nombre<>NULL
                                */ Generar instancia de la clase Entidad-Relacion /*
                        Pos-requisitos: Nombre<>NULL

        INHERIT

        END INHERIT
```

```
        ACCESS TRACES
                T1={ NEW, EDITAR, ELIMINAR}
        END ACCESS TRACES
END CLASS  TEntidad

CLASS  TRelacion
        IS SUBCLASS OF
        INSTANCE VARIABLES
                Nombre: Cadena
                Descripción: Cadena
        METHODLIST
                NEW( )
                        */ Inexistente - Dsponible/*
                        Pre-requisitos: Número de Intancias de entidad > 0
                        Pos-requisitos: Nommbre<> NULL
                EDITAR ( PNombre, PLista de Atributos, PListaRestricciones)
                        */ Disponible - Disponible /*
                        Pre-requisitos: Evento de edición and Nombre<>NULL
                                */ acción: asociar la lista de atributos a una instancia /*
                                */acción: asociar cada elemento de Plista una instancia/*
                        Pos-requisitos: Nombre=PNombre
                ELIMINAR ()
                        */ Disponible-Eliminado*/
                        Pre-requisitos: Evento de eliminación and Nombre<>NULL
                                */ Actualizar  /*
                        Pos-requisitos: Nombre=NULL
                ASOCIAR ()
                        */ Disponible-Disponible*/
                        Pre-requisitos: Evento de conección and Nombre<>NULL
                                */ Generar instancia de la clase Entidad-Relacion  /*
                        Pos-requisitos: Nombre<>NULL
        INHERIT

        END INHERIT
        ACCESS TRACES
                T1={ NEW, EDITAR, ELIMINAR}
        END ACCESS TRACES
END CLASS  TEntidad
```

BIBLIOTECA UIS

```
CLASS  TEs_Un
        IS SUBCLASS OF TRelacion
        INSTANCE VARIABLES
                Restriciciones_Especiales: Cadena
        METHODLIST
                NEW( )
                        */ Inexistente - Dsponible/*
                        Pre-requisitos: Número de Intancias de entidad > 0
                        Pos-requisitos: Nommbre<> NULL
        INHERIT
                ALL SUPER OF TRelacion
        END INHERIT
        ACCESS TRACES
                T1={ NEW, EDITAR, ELIMINAR}
        END ACCESS TRACES
END CLASS  TEsUn
```

5. CONCLUSIONES Y RECOMENDACIONES

- En la Ingeniería del software las etapas iniciales de concepción, son críticas y por lo tanto la calidad del producto deseado se ve altamente dependiente del éxito que se tenga al abordar estas primeras etapas, por esto, la construcción de Metodologías, Técnicas y Herramientas que den soporte a dichas etapas, se convierte en un aspecto clave para la Ingenierías del Software.

- ESFOSIN cumple con las expectativas planteadas, ya que apoya el proceso de especificaciones formales como fundamento del desarrollo de sistemas de información.

- Las metodologías y técnicas que actualmente se plantean, dentro de la Ingeniería del Software, están enfocadas hacia los modelos "Orientados a Objetos". ESFOSIN hace que la concepción[114] esquemática de los sistemas de información, se convierta en especificaciones orientadas a objetos.

- Es posible obtener un modelo de especificación, fundamentado en la descripción gráfica de los componentes de un sistema de información.

- El proceso de especificación es directamente responsable de la calidad de un producto Software. Por esto se hace indispensable concentrar esfuerzos de investigación y desarrollo dentro de la ingeniería del software, hacia la construcción de métodos, guías, procedimientos, técnicas y herramientas que apoyen al analista en el desarrollo de especificaciones.

- Las especificaciones formales son fundamentales en la evaluación del producto software desarrollado[115].

---

[114] Expresada mediante diagramas entidad relación y máquinas de estado.

[115] El proceso de prueba o evaluación de productos, es precisamente la constatación de que el producto cumple con cada una de las especificaciones. Si las especificaciones son formales se puede demostrar matemáticamente, si un producto cumple o no con la especificación dada.

BIBLIOTECA UIS

- El modelo de especificaciones planteado es didáctico y fácilmente adaptable al ambiente actual de desarrollo de software, dado que los conceptos manejados dentro del modelo, son comúnmente utilizados.

- Los delineamientos metodológicos planteados, no se salen notablemente, de los esquemas tradicionales y se enfocan en obtener los productos necesarios para la construcción del modelo global del sistema.

5.1 CUMPLIMIENTO DE LOS OBJETIVOS

- El modelo de especificaciones planteado, cumple con las expectativas referente a la descripción gráfica de sistemas de información, a partir de las cuales se obtiene especificaciones formales.

- La sección - Modelo propuesto - muestra los delineamientos metodológicos necesarios para la utilización del modelo y la herramienta propuesta.

- El prototipo soporta la descripción gráfica de la estructura y comportamiento de los componentes individuales del sistema de información, asistiendo al analista en la descripción de cada uno de los componentes involucrados en el modelo.

- Las condiciones necesarias y suficientes para la generación de especificaciones formales, a partir de la descripción gráfica de sistemas, se identificaron mediante el estudio de diferentes técnicas de modelado y lenguajes de especificación formal. Dichas condiciones se ven proyectadas en el modelo y la herramienta propuesta.

5.2 APORTES

- El modelo de restricciones para cada uno de los componentes descriptores del sistema de información y el diseño global es flexible, permitiendo su crecimiento escalonado, sin modificar de forma radical el modelo original.

- ESFOSIN se proyecta como un modelo para cerrar la brecha entre lo informal y lo formal.

- Se obtuvo una evaluación internacional, en el evento "Software Engineering", organizado por The International Association of Science and Technology for Development -IASTED, en San Francisco, USA, del 2 al 5 de Noviembre de 1997.

- Una divulgación nacional en la V jornada Iberoamericana de Informática, Organizada por la Agencia Española de Cooperación Internacional -AECI-, en Cartagena de Indias del 9 al 13 de marzo de 1998.

5.3 ESTUDIOS POSTERIORES

- Un modelo de especificaciones, utilizando notación propia de técnicas orientadas a objetos.

- Generar los modelos para demostrar completitud y consistencia en los sistemas especificados, utilizando para ello la especificación formal obtenida a través de una herramienta computarizada.

- Crear una herramienta software en ambientes distribuidos que permita el análisis y diseño compartido por múltiples personas, registrando los cambios en línea[116].

- Diseñar herramientas "inteligentes" que apoyen el proceso de implementación y evaluación de productos software, a partir de su especificación formal.

- Crear una herramienta de simulación del sistema de información, a partir de las descripciones gráficas propuestas en el modelo, de soporte a todo el paradigma de ingeniería del software automatizado, fundamentado en la formalización de cada una de sus etapas.

---

[116] Esto es útil para el desarrollo de grandes proyectos en los cuales interviene un equipo humano numeroso disperso geográficamente.

BIBLIOTECA UIS

BIBLIOGRAFÍA

[1] ANDLEIGH P., GRETZINGER M.. "Distributed Object-Oriented Data System Design", Prentice-Hall. 1992.

[2] ASOCIACIÒN Colombiana de Informática; Universidad de los Andes; Centro latino Americano de Estudios en Informática. "XIII Conferencia latinoamericana de Informática y XI Congreso Colombiano de Cálculo electrónico e investigación operacional". Noviembre 9 al 13 de 1987. Bogotá. Colombia.

[3] BALZER Robert. GOLDMAN Neil y WILE David. "Informality in program especifications". IEEE transaction on software Engineering. Vol 4. páginas 94-103. march 1978.

[4] BORGIDA Alex. MYLOPOULOS Jhon y REITER Raymound. "On the frame problem in procedure specifications". IEEE Transaction on Software Enginneering, vol 21, No 10. octubre de 1995.

[5] BOURDEAU Roger H. y CHENG Betty HC. " A Formal Semantic for Model Diagrams". IEEE Transaction on Software Enginneering. vol 20 No 10. Octubre de 1995

[6] CARDOSO Rodrigo. Verificación y desarrollo de programas. Editorial ECOE ediciones. Ediciones Uniandes. Colombia 1993. 473 Páginas.

[7] CORNELL Gary y STRAIN Troy. Programación en Delphi, traducción de María Dolores del Castillo. Editorial Osborne McGraw-Hill. España 1996.

[8] CORREAL Dario. FRANKY María Consuelo. "Técnicas de Especificación y verificación de sistemas distribuidos". Memo de Investigación No. 145. Universidad de los Andes, Santa Fe de Bogotá. Julio de 1994.

[9] D'ALMEIDA Juliette, ACHUTHAN R., RADHAKRISHNAN T., ALAGAR V.S. "Transformation of a Semi-formal specification to VDM". Deparment of Computer Science, Concordia University, Montreal, Canada, H3G 1 M8

[10] DURR Eugene, KATWIJK Jan van. "VDM ++, a Formal Specification Language For Object-Oriented Designs". IEEE 1992.

[11] FANG f.w Wlater, SO Andrew C., KRELNDLER R. Jordan. "The Visual modeling technique: An introduction and overview", ROAD (Report on Object Analysis and Design). Julio y Agosto de 1996.

[12] FRANKY María Consuelo, CAMACHO Sandra, "SFOAD: Soporte a una metodología de desarrollo de sistemas distribuidos Orientado por Objetos", Memos de investigación, Universidad de los Andes, Santa Fé de Bogotá.

[13] GRASSMAN Winfriend Karl, TEMBLAY Jean-Paul. Matemáticas discretas y lógica, una perspectiva desde la ciencia de la computación, traducción de Rafael García Barrejo, Universidad de Salamanca, Editorial Prentice Hall, España 1996. 706 páginas.

[14] KELLY Dean, Teoría de autómatas y lenguajes formales, Editorial Prentice Hall, 1996, 302 páginas.

[15] KORTH Henry F; SILBERSCHATZ Abraham. "Fundamentos de Bases de Datos", Segunda Edición, McGraw-Hill, 1993, Páginas 25-54.

[16] LANO K., HAUGHTON H., " Standards and Techniques for Object-Oriented Formal Specification". IEEE Transaction 1993, 0-8186-4240-8/93.

[17] LEVESON Nancy G., HEIMDAHL Mats Per Erik, "Requirements Specification for Process-Control System", IEEE transaction on software Engineering, vol 20 No 9 1994.

[18] LLAMOSA Villalba Ricardo, "UIS-ESI-LIS", Análisis y diseño. Bucaramanga, 1995.

[19] LLAMOSA Villalba Ricardo, "Modelos de especificación", 1996.

[20] MARTIN James y ODELL James, Análisis y Diseño Orientado a Objetos, traducción de Oscar Alfredo Palmas Velasco, Editorial Prentice Hall, 1992.

[21] MATS P.E., HEIMDAHL y LEVESON Nancy G. " Completeness and Consistency in Hierarchical State-Based Requirements". IEEE Transaction on Software Enginniering, vol 22, No 6, Junio de 1996.

[22] MONTILVA Jonás. "Desarrollo de sistemas de información", Universidad de los Andes, consejo de publicaciones, Mérida Venezuela, 1992.

[23] MYERS, G. "Composite Structured Design", Van Nostrand, 1978.

[24] OSIER Dan, GROBMAN Steve, BAISON steve. Aprendiendo Delphi 2 en 21 días. Editorial Prentice Hall, México 1996, 704 páginas.

[25] PRESSMAN Roger S, "Ingeniería del Software Un enfoque practico". McGraw-Hill, Segunda Edición, 1988, Páginas 153-402 de 824 páginas.

[26] RUMBAUGH James, BLAHA Michael y otros. Modelado y diseño orientado a objetos, metodología OMT. Editorial Prentice Hall, España 1991, 643 páginas.

BIBLIOTECA UIS

ANEXOS

1. ANEXO: DEFINICIONES DE SISTEMAS DE INFORMACIÓN

" Un sistema de Información es una colección de recursos que está diseñado y construido para adquirir, registrar, procesar, almacenar, recuperar y mostrar información".

D.Tecichroew.

"Un sistema de información es un sistema integrado hombre/máquina, que provee información para apoyar las funciones de: operación, gerencia y toma de decisiones en una organización. El concepto de sistema hombre/máquina implica que algunas tareas las realiza mejor el hombre mientras que otras las hacen mejor las máquinas... El sistema integrado cubre datos y procedimientos. La integración de los datos es ejecutada por la herramienta de gestión de la base de datos... mientras que el procesamiento es realizado por un plan general del sistema".

G. Davis

" El sistema de información dentro de una organización es análogo al del sistema nervioso de un animal. En el sistema están incluidos los componentes que ejecutan funciones tales como las de: percepción, clasificación, transmisión, almacenamiento, recuperación, y transformación. Su propósito primordial es suministrar información para la toma de decisiones y su coordinación ... En el sentido más amplio un sistema de información , incluye componentes para la toma de decisiones , coordinación y advertencia; y son tanto humanos como automáticos"

J. Emery.

"Un sistema de información es: "Un sistema (basado en una máquina de procesos de información) que procesa datos, en forma tal que quien los recibe puede tomar decisiones". ".. es definido como un medio organizado de proporcionar información pasada, presente y futura (proyecciones) relacionada con las operaciones internas y conocimiento externo de la organización".

J. Senn.

" Un sistema de información se define como: Un ensamble formal y sistemático de componentes que ejecutan operaciones de procesamiento de datos para: a) satisfacer los requerimientos y el procesamiento de datos legales y transaccionales, b) proporcionar información a la gerencia para apoyar actividades de planificación, control y toma de decisiones, y c) proporcionar una variedad de reportes, que sean requeridos por entes externos".

J. Burch y F. Strater

"Un sistema de información es un conjunto organizado de hombres, máquinas, programas y procedimientos para llevar a cabo funciones para cumplir objetivos deseados".

W. Harman, H.Matthes y A. Proeme.

"Un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información".

"Un sistema de Información es [22]:

- *Desde el punto de vista Estructural*: Un sistema hombre/máquina, integrado por personas, procedimientos y equipos. Esta característica plantea, que no es posible la existencia de un sistema de información sin el computador; aseveración falsa, pues se sabe que los sistemas de información existen desde el mismo momento en que surgieron las organizaciones. Sin embargo se debe reconocer que el uso de las máquinas facilita la ejecución de las funciones en un sistema de información. Una organización mediana o grande difícilmente puede concebir y utilizar un sistema de información sin el uso de los Computadores que facilitan el procesamiento de los datos. Por lo tanto la relación hombre-máquina en un sistema de información,

depende del grado o nivel de participación de personas y del grado o nivel de utilización de las máquinas, por lo que podemos aceptar como tal a un sistema de información manual o un sistema de información automatizado o computarizado.

- *Desde el punto de vista Funcional:* El objetivo del sistema de información que facilite la ejecución de tareas, operaciones y funciones en una organización. La información requerida por la gerencia es solo una parte de todo el espectro de información que un sistema puede suministrar, Sin embargo la toma de decisiones se realiza en todos los niveles organizacionales desde la ejecución de tareas o actividades básicas hasta la planificación de estrategias; por lo tanto, el sistema de información debe suministrar información en todos los niveles, a las personas autorizadas dentro y entorno de la organización.

- *Desde el punto de vista operacional:* En un sistema de información *l*a operación central está constituida por el procesamiento de datos originados por las transacciones[1] y entidades para producir y diseminar información en la organización y su ambiente."

Jonás Montilva

[1] Una transacción es definida como un evento o acontecimiento que ocurre dentro de los límites de la organización y que la afecta de algún modo.

BIBLIOTECA UIS

2. ANEXO: MODELADO DE SISTEMAS DE INFORMACIÓN

2.1 MODELO ENTIDAD RELACIÓN

2.1.1 Concepto

El modelo Entidad Relación (E-R) es una técnica para definir las necesidades de información de un sistema.

Se basa en una percepción de un conjunto de objetos básicos[2] llamados entidades[3], las propiedades de dichos objetos denominadas atributos y las relaciones entre estos objetos.

2.1.2 Objetivos

- Proporcionar un modelo[4] preciso de las necesidades de información de la organización.
- Proporcionar un modelo independiente, que describa la estructura general de un sistema.

2.1.3 Definiciones

Una entidad es una cosa o un objeto con significado real o imaginario acerca de las necesidades de información que se van a conocer o a mantener.

Un conjunto de entidad, es un conjunto formado por todas las instancias pertenecientes a un mismo tipo[5] o clase de entidad.

Un tipo de entidad está representado por un conjunto de atributos. Formalmente, un atributo es una función que asigna un conjunto de entidades a un dominio. Así cada entidad en un momento

---

[2] Componentes del mundo real que se desea modelar.

[3] Hay que diferenciar una entidad de una instancia de entidad, tal como se aclara en la definición de entidad. Haciendo una analogía con la terminología de la programación orientada a objetos: Clase es a objeto como Tipo de entidad es a entidad.

[4] Este esquema representa la estructura lógica y global del sistema.

[5] El termino "Tipo de entidad" en este documento hace referencia a una clase de objetos y no a ningún objeto en particular, a este lo llamamos "instancia de entidad" o simplemente "entidad".

determinado se describe por medio de un conjunto de pares (atributo, valor del dato), un par para cada atributo de la entidad.

Para cada entidad se define una superclave, que es un conjunto de uno o más atributos que, considerados conjuntamente, nos permiten identificar de forma única una entidad en el conjunto de entidades. Si K es una superclave, entonces también lo será cualquier superconjunto de K. A menudo estamos interesados en superclaves para las cuales ningún subconjunto propio es superclave[6].

Una relación es una asociación entre varias entidades. Un conjunto de relaciones es el conjunto formado por todas las relaciones entre entidades particulares. Formalmente es una relación matemática de n conjuntos[7] de entidades. Si E1, E2,... ,En son conjunto de entidades, entonces un conjunto de relaciones es un subconjunto de:

$\{ (e1,e2,e3,\ldots,en) / e1 \in E1\ e2, \in E2 , \ldots en \in En \}$

donde (e1,e2,e3,....en) es un elemento de la relación.

En el modelo Entidad Relación, siempre es posible sustituir un conjunto de relaciones no binarios por varios conjunto de relaciones binarias[8] distintas.

La función que desempeña una entidad en una relación se llama papel o rol. Los papeles, normalmente son implícitos y no se suelen especificar. Sin embargo, son útiles cuando el significado de una relación necesita ser clarificado.

Dentro de una relación la cardinalidad de asignación expresa el número de entidades con las que se puede asociar otra entidad, en un conjunto de relaciones, para un conjunto de relaciones binarias entre los conjunto de entidades A y B, la cardinalidad de asignaciones debe ser una de las siguientes:

---

[6] Dichas superclaves mínimas son claves candidatas para ser clave primaria de la entidad.

[7] Donde n mayor o igual que dos y posiblemente los conjuntos son no disyuntos.

[8] Una relación es binaria, en el sentido de que es siempre una asociación entre dos entidades.

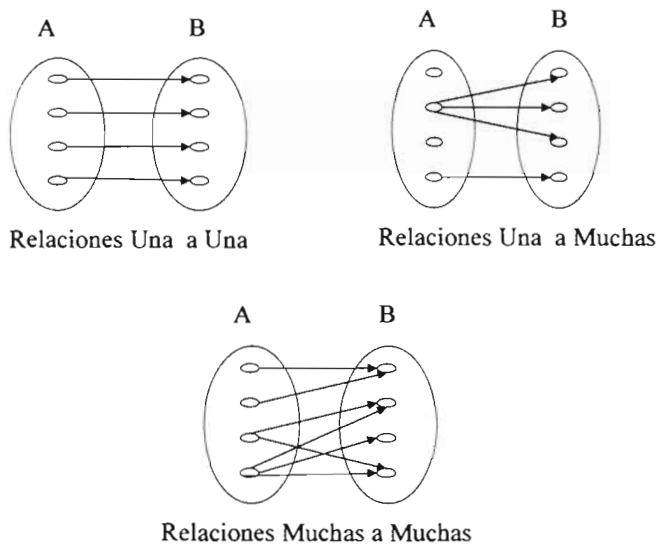

Figura 1 Representación de los tipos de relaciones en el modelo entidad Relación

- Una a Una. Una entidad en A está asociado a lo sumo con una entidad en B y una entidad en B está asociado a lo sumo con una entidad en A.

- Una a muchas. Una entidad en A está asociado con un número cualquiera de entidades en B y una entidad en B está asociado a lo sumo con una entidad en A.

- Muchas a muchas: Una entidad en A está con un número cualquiera de entidades en B, y una entidad en B está asociada con un número cualquiera de entidades en A.

2.1.4 Características especiales del modelo entidad relación extendido

2.1.4.1 Entidades fuertes y débiles

En el modelo Extendido, es posible que un conjunto de entidades no tenga atributos suficientes para formar una clave primaria, por lo tanto se le denomina conjunto de entidades débil[9].

Un miembro de un conjunto de entidades fuerte es, por definición, una entidad dominante, mientras que un miembro de un conjunto de entidades débil es una entidad subordinada.

Aunque un conjunto de entidades débil no tiene una clave primaria, sin embargo debe haber un medio para distinguirla entre todas aquellas entidades en el conjunto de entidades que dependen de

[9] Un conjunto de entidades que tiene una clave primaria, se denomina conjunto de entidades fuerte.

una entidad fuerte determinada. El discriminador de un conjunto de entidades débil es un conjunto de atributos que permiten que se haga esta distinción.

La clave primaria de un conjunto de entidades débil está formada por su discriminador y la clave primaria del conjunto de entidades fuerte de la que depende su existencia.

2.1.4.2 Generalización

Es una relación de inclusión entre un conjunto de entidades de nivel más alto y uno o más conjunto de entidades de nivel más bajo, se usa para resaltar lo parecidos entre tipos de entidades de nivel más bajo y ocultar sus diferencias. La distinción se hace a través de un proceso llamado herencia de atributos. Los atributos del conjunto de entidades de nivel mas alto, son heredados por los conjuntos de entidades de nivel más bajo.

2.1.4.3 Agregación

La agregación es una operación mediante la cual las relaciones se tratan como entidades de nivel más alto.

La agregación se puede ver como una combinación de diferentes clases de objetos para conformar una nueva clase de objetos, con propiedades emergentes[10] y heredadas[11]. Históricamente la agregación es la manera de representar que un objeto complejo tiene otros objetos como componentes.

2.1.5 Uso de las características del modelo entidad relación extendido

Cada una de las características del modelo entidad relación extendido[12], contribuyen a la modularidad del diseño[13]:

- Un conjunto de entidades fuerte y sus conjuntos de entidades débiles dependientes pueden ser considerados como un objeto único en el sistema, ya que las entidades débiles son dependientes por existencia de una entidad fuerte.

---

[10] Propias de la clase agregada.

[11] De las clases que se agregan.

[12] Entidades fuertes y débiles, agregación y generalización

[13] Sin embargo el uso excesivo de estas características puede introducir una complejidad innecesaria en el diseño.

- La agregación agrupa una parte del diagrama entidad relación en un conjunto de entidades únicas. Es posible tratar el conjunto de entidades agregado como una entidad única sin preocuparse de los detalles de su estructura interna.

- La generalización contribuye a la modularidad permitiendo que atributos comunes de conjunto de entidades similares sean representados una sola vez en un diagrama entidad relación.

2.1.6 Representación gráfica del modelo entidad relación

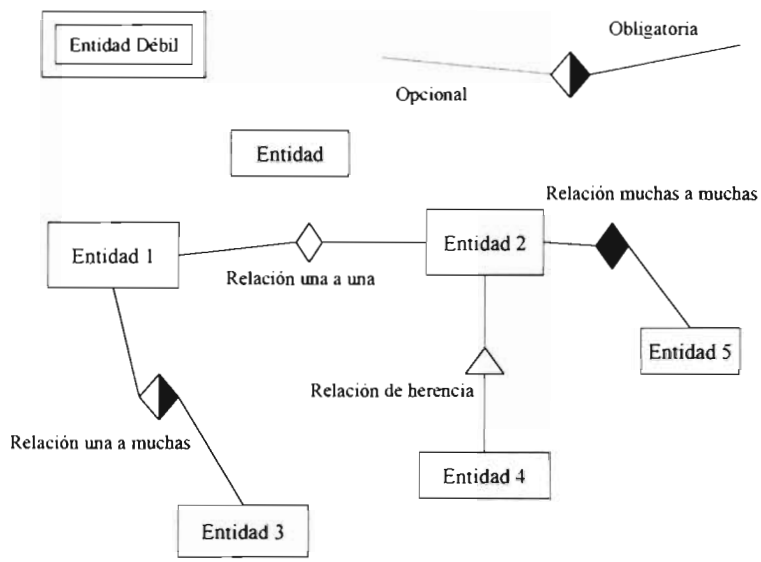

Figura 2 Representación gráfica de los componentes del modelo Entidad Relación

2.1.6.1 Entidades

Una entidad se representa en forma de diagrama con un rectángulo con un nombre en singular y en mayúscula. El recuadro puede tener cualquier tamaño suficiente para contener un nombre sin ambigüedad (sin abreviaturas).

El nombre de la entidad debe ser el que represente una clase o tipo de elemento y no una instancia en particular. Cualquier objeto solo puede ser representado por una entidad. Es decir las entidades son mutuamente exclusivas en todos los casos. Una entidad débil se representa con un rectángulo doble

2.1.6.2 Relaciones

Una relación se representa mediante un rombo unido con líneas a los recuadros de entidades que intervienen en la relación, o recursivamente sobre una misma entidad.

BIBLIOTECA UIS

Cada relación tiene dos extremos, para cada uno de los cuales tiene un nombre o rol, un grado o cardinalidad y una opcionalidad.

La opcionalidad se representa mediante líneas punteadas y la obligatoriedad con líneas continuas.

Para la representación de la generalización[14], se utiliza un triángulo con el título Es Un.

2.2 DIAGRAMAS DE TRANSICIÓN DE ESTADOS (DTE)

Un estado es un conjunto de circunstancias o atributos que caracterizan a una persona o cosa en un tiempo determinado. Cualquier estado observable en el que el sistema puede encontrase solo puede responder a periodos en los que: 1) está esperando que algo ocurra en el ambiente externo o, 2) está esperando a que alguna actividad que se esté dando en ese momento en el ambiente, cambie a otra.

Esto no significa que los sistemas modelados no sean capaces de ejecutar acciones o que no pretendamos mostrarlas, sino solo que las acciones que ocurren instantáneamente en el modelo de tecnología perfecta, no son lo mismo que los estados, que representan condiciones observables en las que el sistema se puede encontrar, esto es que un estado representa algún comportamiento del sistema que es observable y que perdura durante algún periodo finito.

Un sistema que existió en un solo estado, no sería un objeto de estudio muy interesante, sería estático. De hecho los sistemas de información que se modelan normalmente pueden tener muchos estados diferentes y por lo tanto deben existir transiciones[15] entre dichos estados.

En un DTE los cambios de estado se representan con una flecha que indica el estado origen y sucesor[16].

La mayoría de sistemas tienen un estado inicial y un estado final reconocibles. Lo que identifica un estado inicial, en un DTE, es una flecha "desnuda" que no está conectado a ningún estado origen

---

[14] Considerada como una relación especial (de herencia).

[15] Estas transiciones muestran los cambios de estado válidos en el sistema.

[16] Cualquier estado puede llevar un número arbitrario de estados sucesores

y lo que identifica a un estado final, es la ausencia de flechas que salgan de él, en otras palabras, una vez que se ha llegado a un estado como este, no se puede ir a ninguna otra parte.

El sentido común indica que un sistema solo puede tener un estado inicial; sin embargo, puede tener múltiples estados finales. Los diversos estados finales son mutuamente excluyentes, lo cual significa que solo uno de ellos puede ocurrir durante alguna ejecución del sistema.

Para completar un DTE se necesita agregar dos cosas más: Las condiciones que causan un cambio de estado y las acciones que el sistema toma cuando se cambia de estado. En un DTE las condiciones y acciones se muestran junto a la flecha que conecta dos estados relacionados.

Una condición es un acontecimiento en el ambiente externo que el sistema es capaz de detectar; típicamente es una señal, una interrupción o la llegada de un paquete de datos.

Como parte del cambio de estado, normalmente el sistema hará una o más acciones: producirá una salida, desplegará una señal, llevará a cabo un cálculo, etc. Así que las acciones que se muestran en un DTE son respuestas regresadas al ambiente externo o bien cálculos cuyos resultados el sistema recuerda[17] para poder responder a algún acontecimiento futuro.

Para la construcción de un DTE se recomiendan seguirse cualquier de estos dos enfoques:

- Comenzar por identificar todos los posibles estados del sistema y representar cada uno como una caja separada en una hoja de papel. Luego se puede explorar todas las conexiones con significado (es decir los cambios de estado) entre cajas.

Como alternativa se puede comenzar por el estado inicial y luego metódicamente seguir un camino hasta los estados restantes; luego a los estados secundarios, proseguir a los estados terciarios; etc.

---

[17] Típicamente en un almacén de datos que se muestre en un DFD.

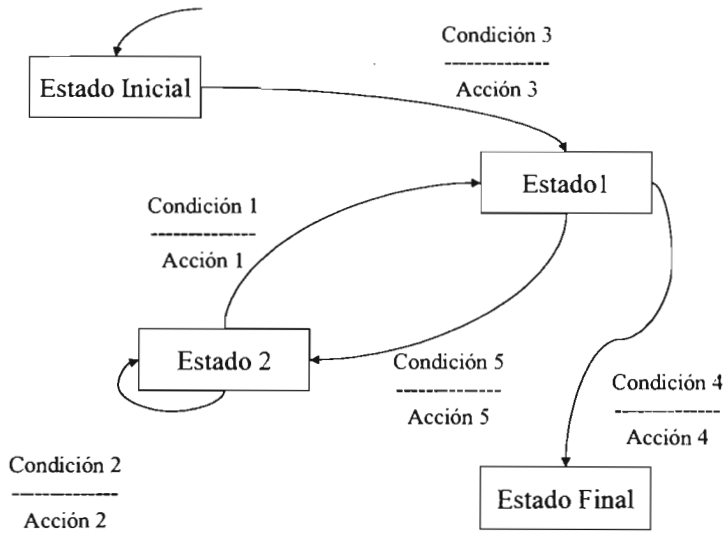

Figura 3 Representación típica de un DTE

Cuando se termine el DTE preliminar, debe seguirse las estas reglas para verificar la consistencia.

- ¿ Se han definido todos los estados?, verificar si existe algún otro comportamiento observable, o alguna otra condición en la que el sistema pudiera estar, aparte de las que se han identificado.

- ¿ Se puede alcanzar todos los estados? ¿Se han definido estados que no tengan caminos que lleguen a ellos?

- ¿ Se puede salir de todos los estados?, todos los estados deben tener por lo menos un sucesor excepto los estados finales.

- En cada estado, ¿ El sistema responde adecuadamente a todas las condiciones posibles?, tanto para condiciones normales como inesperadas.

2.3 DIAGRAMAS DE FLUJO DE DATOS (DFD)

El diagrama de flujo de datos es una de las herramientas más comúnmente usadas y sobre todo por los sistemas operacionales en los cuales las funciones del sistema son de gran importancia y más complejas que los datos que este maneja.

Los DFD se utilizaron por primera vez, en la ingeniería del software, como notación para el diseño de sistemas y continúan siendo utilizados por los ingenieros de software que trabajan en la implementación directa de modelos de requerimientos de los usuarios.

BIBLIOTECA UIS

Los DFD no solo se puede utilizar para modelar sistemas de procesos de información, sino también para modelar organizaciones enteras[18]. Además la notación es sencilla, clara y en cierto sentido intuitivamente obvia, por lo cual prácticamente no requiere explicación.

Es bueno tener en mente que los DFD son tan solo una de las herramientas de modelado disponibles y que únicamente proporcionan un punto de vista del sistema, el orientado a las funciones. Si se está desarrollando un sistema en donde las relaciones entre los datos, son más importantes que las funciones, tal vez se dé menos importancia al DFD, para concentrarse, más bien, en desarrollar un conjunto de diagramas entidad-relación. De otra manera, si en un sistema, el comportamiento dependiente del tiempo, domina cualquier otro factor, tal vez se debe concentrar el esfuerzo en los diagramas de transición de estados.

Los componentes de un diagrama de flujo de datos son: los procesos, los flujos, los almacenes y los terminadores.

El proceso muestra una parte del sistema que transforma entradas en salidas; es decir muestra como una o más entradas se transforman en salidas. Se nombra con una sola palabra, frase u oración sencilla que describe lo que hace, en algunos casos describe quién o qué lo está ejecutando, más que describir el proceso mismo.

Un flujo se representa gráficamente por una flecha que entra o sale de un proceso. Se utiliza para describir el movimiento de bloques o paquetes de información de una parte del sistema a otra, representando así, datos en movimiento. El nombre representa el significado del paquete que se mueve a lo largo del flujo.

El almacén se utiliza para modelar una colección de paquetes de datos en reposo. Se denota por dos líneas paralelas. De modo característico el nombre que se utiliza para identificar el almacén es el plural del que se utiliza para los paquetes que entran y salen del almacén por medio de flujos y por lo tanto, los flujos conectados a un almacén, solo pueden transportar paquetes de información que el almacén sea capaz de guardar.

---

[18] Como una herramienta para la planeación estratégica y de negocios.

Los terminadores representan entidades externas con las cuales el sistema se comunica, pero fuera del control del sistema que se está modelando. Un terminador puede ser otro sistema. Gráficamente se representan con un rectángulo.

Tres cosas importantes se deben recordar sobre los terminadores:

- Son externos al sistema que se está modelando; los flujos que conectan los terminadores a los diversos procesos o almacenes, representan la interfaz entre él y el mundo externo.
- Es evidente que ni el analista, ni el diseñador del sistema están en posibilidad de cambiar los contenidos de un terminador o la manera en que este trabaja[19].
- Las relaciones que existen entre terminadores, no se muestran en un DFD.

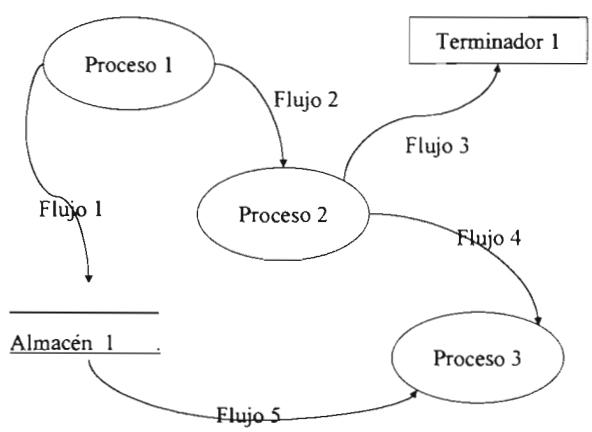

Figura 4 Representación típica para un DFD

Los flujos son simplemente los conductos a lo largo de los cuales viajan los paquetes de datos entre procesos y almacenes. Para una amplia clase de sistemas, sobre todo de negocios, estos tipos de flujo son suficientes, pero para otra clase de sistemas, los de tiempo real, se necesita alguna manera de modelar flujos de control[20], procesos de control[21] y en algunos casos puede ser apropiado incluir almacenes de control o almacenes de eventos.

BIBLIOTECA UIS

Un flujo de control puede imaginarse como un conducto que porta una señal binaria[22], a diferencia de otros flujos, un flujo de control no porta datos con valores.

En proceso de control sus entradas y salidas consisten solo de flujos de control. Los flujos de control salientes del proceso de control, se utilizan para despertar otros procesos; los entrantes indican que un proceso ha terminado su labor o que se ha presentado una situación extraordinaria de la cual necesita informarse el proceso de control. Comúnmente solo hay un proceso de control dentro de un DFD.

Un flujo de control se utiliza para despertar un proceso normal; una vez despierto, el proceso normal procede a llevar a cabo su labor, tal como lo describe la especificación del proceso. Sin embargo el comportamiento interno de un proceso de control es diferente; aquí es donde el comportamiento dependiente del tiempo del sistema se modela con detalle. El interior del proceso de control se modela con un diagrama de transición de estados.

2.4 EL MODELO ORIENTADO A OBJETOS

El paradigma Orientado a Objetos (OO) está jugando un papel cada vez más importante en el diseño e implementación de sistemas de información (SI), puesto que aporta una notación apropiada para el desarrollo de cualquier tipo de software y genera una visión del SI mucho más cercana a la realidad, ya que a la hora del análisis no se separan los datos de las operaciones, sino que ambos se recogen en la definición de los objetos.

El paradigma OO no se definió de forma precisa de entrada, pero debido al éxito del mismo, gran número de equipos de investigación están trabajando desde hace varios años en clarificar los fundamentos teóricos de la orientación a objetos.

Los rasgos más destacados del modelo OO son :

---

[19] El terminador está fuera del dominio de cambio.

[20] Señales o interrupciones

[21] Un proceso cuya única labor es coordinar y sincronizar las actividades de los otros procesos del DFD

[22] Encendido o apagado

- Está basado en los principios básicos de la Ingeniería del Software, que son: *Abstracción*[23] *y Ocultación*[24] de la información.

- Aparece el concepto de objeto como un tipo abstracto de datos (TAD) ampliado, al que se dota de un estado local oculto, que encapsula tanto datos como operaciones. Es el propio concepto de objeto, el que vela por el principio de ocultamiento.

- El sistema se ve globalmente como una sociedad de objetos que se comunican intercambiando mensajes.

Un objeto es una unidad operacional autocontenida, que encapsula aspectos estáticos y dinámicos. Durante su existencia tendrá un comportamiento caracterizado por las acciones que es capaz de realizar y que son invocadas mediante mecanismos de interacción entre objetos.

Las propiedades de los objetos se representan por medio de atributos, de modo que el estado de un objeto, en un instante dado, viene determinado por el conjunto de los valores de sus atributos.

El concepto de cambio de estado es muy importante, hay casos fuertes de cambio de estado (creación y destrucción) y casos débiles[25] . La abstracción del cambio de estado es llamado evento[26].

Cada objeto tiene asociado un conjunto de eventos, que son los que le pueden afectar durante su vida[27]. Un objeto se convierte de esta forma en un proceso observable.

No todas las propiedades de un objeto van a ser relevantes. El conjunto de propiedades[28] seleccionadas, va a depender del analista (observador del sistema), que elegirá aquellas que estime necesarias de acuerdo con el propósito de su modelo.

---

[23] Centrarse en los aspectos esenciales y ocultar aquellos que no lo sean, según la perspectiva del observador

[24] Toda la información sobre un módulo debe ser privada, salvo la que expresamente se declare como pública.

[25] Cambio de estado en un objeto existente, que no implica su destrucción.

[26] Los eventos son discretos, no tienen duración y ocurren en un instante de tiempo puntual.

[27] El estado de un objeto en un instante dado será una función de los eventos que conforman su vida hasta ese momento en el que es observado.

[28] Atributos del objeto

Los atributos de un objeto se clasifican en: atributos constantes[29], variables[30], derivados[31].

Distinguiremos igualmente entre dos tipos de eventos[32]: propios[33] y compartidos[34].

Los eventos son activados por agentes responsables de su ejecución. En sistemas activos, tanto el entorno, como los objetos existentes, pueden actuar como agentes ejecutando eventos. Un evento sólo será relevante en la vida de un objeto si el estado del mismo es el apropiado. Esto se expresa a través de las llamadas precondiciones[35].

Los objetos no están aislados se pueden comunicar de dos formas: compartiendo eventos y mediante relaciones de disparo. Las relaciones de disparo introducen actividad en la Sociedad de Objetos (SO), permitiendo que un objeto actúe como agente cuando se satisfagan en él determinadas condiciones. Existen ciertas reglas que deben seguir los valores de los atributos de los objetos que habitan en una SO. Estas reglas, llamadas restricciones de integridad, se expresan mediante fórmulas en alguna lógica, y deben cumplirse en todos los estados por los que pasen los objetos a lo largo de su vida (restricciones estáticas) o a lo largo de ciertas secuencias de estados (restricciones dinámicas).

Se construyen procesos usando los eventos y transiciones como acciones atómicas, permitiéndonos especificar las posibles vidas correctas de los objetos. Es decir, cuando un objeto está en un estado determinado, cuál es el conjunto de estados potencialmente válidos alcanzables[36].

Por medio de los mecanismos de abstracción, reunimos los objetos[37] con las mismas propiedades en clases. Llamamos tipo, a un conjunto de propiedades que son características de varios objetos, es

---

[29] Su valor no cambia durante la vida del objeto.

[30] Su valor cambia como consecuencia de la ocurrencia de eventos relevantes.

[31] Su valor se da en términos de valores de demás atributos.

[32] Asumimos que evento equivale a suceso, es decir, lo que acontece en un instante de tiempo.

[33] Son los que participan en la vida de los objetos de una única clase (aparecen sólo en la signatura de la clase considerada).

[34] Son los que forman parte de la vida de los objetos de más de una clase.

[35] Fórmulas que se deben satisfacer para que la ocurrencia de un evento tenga éxito.

[36] Relación de accesibilidad y un cierre transitivo: traza

[37] Un objeto individual es una instancia o ejemplar de una clase.

BIBLIOTECA UIS

decir, será un esquema o patrón de objeto y que define entre otras cosas, la estructura de memoria privada y el protocolo de comunicación entre objetos.

Una vez definidas estas clases elementales, a través de un mecanismo constructivo se pueden definir clases complejas a partir de ellas[38], mediante la aplicación de operadores de clases tales como agregación[39], asociación[40], especialización/generalización[41], composición paralela[42].

La especialización define la herencia descendiente o derivación de clases hijas a partir de una clase padre, heredando los descendientes, las propiedades definidas en sus respectivos padres[43]. Hay dos tipos básicos de especialización:

a) Especialización temporal o rol: llamada también especialización dinámica, en ella la clase especializada posee su propio evento de creación de instancias, que debe pertenecer a la colección de eventos de la clase padre. Una vez este evento ocurre, el objeto juega el rol correspondiente a la clase especializada.

b) Especialización permanente: un objeto pertenece a la clase especializada desde el mismo momento de su creación debido a que se cumple una condición de especialización. El evento de creación de la clase especializada coincide en este caso con el de la clase padre.

La generalización es la operación inversa a la anterior y define la herencia ascendente o generación de clases padres recogiendo propiedades comunes de clases definidas con anterioridad[44].

---

[38] O a partir de otras clases complejas definidas de antemano.

[39] Esta operación combina clases componentes, y genera una nueva clase con propiedades emergentes (propias de la clase agregada) y heredadas (de las clases que se agregan).

[40] Permite generar clases complejas cuyas instancias agrupan colecciones de instancias de una clase, la clase agrupada, a la que se denomina también clase base. Es un caso particular de la agregación, que se utiliza en varios modelos semánticos para manipular colecciones de objetos como una nueva clase compleja agregada. Es una agregación dinámica y multivaluada.

[41] Operadores que dan soporte a la noción de herencia como mecanismo para compartir estructura y comportamiento entre clases, que posibilita la reutilización de software y cuya jerarquía se representa en forma de grafo.

[42] Permite definir un sistema organizacional como el resultante de la reunión concurrente de las clases definidas con anterioridad. No es un concepto procedente del área del modelado semántico, sino que proviene de la teoría de procesos y surge del modelo OO utilizado.

[43] Además les está permitido redefinir las propiedades e introducir otras nuevas que enriquezcan la especificación

[44] La clase generalizada es construida sobre un conjunto de clases llamadas hijas, abstrayendo sus propiedades comunes y reuniéndolas en una clase de un nivel mayor.

El operador de composición paralela va a permitir definir el esquema conceptual de un Sistema de Información, como una clase compleja resultante de la composición paralela de las clases definidas previamente.

Los operadores de agregación, especialización y composición paralela son ortogonales, es decir, son independientes unos de otros y se puede elegir cualquier combinación de ellos[45].

2.5 FOAD (FRAME OBJECT ANALYSIS DIAGRAMS)

La metodología FOAD[12], propuesta por Andleigh y Gretzinger [1] para el diseño de sistemas de información orientado por objetos, persigue los siguientes objetivos generales:

- Definir las clases de objetos que componen el sistema en términos de sus atributos y de los servicios que debe ofrecer.
- Determinar jerarquías entre las clases de objetos del sistema con miras a lograr un diseño más rápido mediante la reutilización de objetos.
- Incorporar reglas de integridad al sistema.
- Distribuir los componentes del sistema.

La metodología FOAD propone realizar dos modelajes en etapas sucesivas:

1. *Modelado del sistema de información* en término de funciones y de datos. En esta etapa se especifican dos modelos en paralelo: el Modelo de Entidades en donde se identifican las entidades que componen el sistema de información, sus atributos y relaciones; por otra parte, el modelo de transacciones en donde se identifican las funciones y las interacciones entre entidades que se requieren para poder realizar dichas funciones.

Modelado de Objetos que componen el sistema. Aquí se debe refinar el modelado del sistema de información para definir completamente la semántica de cada clase de objeto del sistema. En esta etapa se especifican nuevamente dos modelos: el modelo estructural en donde se establecen las propiedades estructurales de cada clase de objetos y sus relaciones con otras clases; por otra parte, el modelo de comportamiento en donde se describen las operaciones de cada clase de objeto.

---

[45] Esto permite que el abanico de posibilidades sea muy amplio.

BIBLIOTECA UIS

3. ANEXO: DEFINICIONES BÁSICAS DE LA TEORÍA DE GRAFOS [13]

La teoría de grafos tiene aplicaciones en muchos campos, lo cual ha dado lugar a una gran cantidad diversidad en la terminología[46] utilizada.

Los grafos están formados por nodos que se unen entre sí mediante aristas. Por tanto, una definición matemática de grafo debe basarse en N, el conjunto de nodos, y A el conjunto de aristas. Toda arista está asociada con dos nodos, esto es existe una correspondencia entre las aristas u los pares (ordenados o no) de nodos.

Definición 1: Un grafo G=(N, A , f) consta de un conjunto no vacio N denominado conjunto de nodos (puntos, vértices) del grafo, un conjunto A de aristas del grafo y una correspondencia f del conjunto de aristas A en un conjunto de pares de N. Si una arista se corresponde con un par ordenado, entonces se dice que es una arista se corresponde con un par ordenado, entonces se dice que es una arista dirigida; en caso contrario se denomina arista no dirigida.

Observes e que la definición de grafo implica, que a toda arista del grafo G, se le puede asociar una pareja ordenada o desordenada de nodos del grafo. Si una arista e $\in$ A está asociada de esta forma con un par ordenado (u,v) o con un par desordenado {u,v}, en donde u,v $\in$ N, entonces se dice que la arista e conecta o une los nodos u y v. Los pares de nodos que están conectados por una arista dentro de un grafo, se denominan nodos adyacentes. Se supondrá en todo momento que tanto el conjunto A como el conjunto N, de un grafo, son finitos. Con frecuencia será conveniente escribir los grafos en la forma (N, A), o bien simplemente como G. En el primer caso, cada arista se representa directamente como el par con el cual se corresponde, lo cual obvia la necesidad de especificar f si f es una correspondencia 1 a 1.

---

[46] Los distintos autores, utilizan términos distintos para un mismo concepto y lo que es peor, un mismo término para conceptos diferentes.

Definición 2: Un grafo en el cual toda arista es dirigida se denomina digrafo o bien grafo dirigido. Un grafo en el cual todas las aristas son no dirigidas se denomina grafo no dirigido. Si en un grafo hay aristas dirigidas y aristas no dirigidas se denomina mixto.

Definición 3: En un grafo dirigido, para todo nodo v, el número de aristas que tienen a v como nodo inicial, se denomina grado de entrada del nodo v. El número de aristas que tienen a v como nodo terminal, se denomina grado de salida, y la suma del índice de entrada y el de salida, se le denomina grado total del nodo v.

Definición 4: Sea N(H) el conjunto de nodos del grafo H, y sea N(G) el conjunto de nodos de un grafo G, tal que $N(H) \subseteq N(G)$. Si además toda arista de H es también arista de G, entonces se dice que el grafo H es un subgrafo del grafo G, y esto se expresa en forma $H \subseteq G$.

Definición 5: Sea G=(N,A) un digrafo sencillo. Se dice que una sucesión de aristas es un camino de G si y solo si el nodo terminal de cada arista del camino es el nodo inicial de la próxima arista del camino, si lo hubiere.

Definición 6: El número de aristas que aparecen en la sucesión de un camino, se denomina longitud del camino.

Definición 7: Un camino de un digrafo el cual todas las aristas sean distintas, se denomina camino sencillo. Un camino en el que todos los nodos sean diferentes se denomina camino elemental.

Definición 8: Un camino que comienza y acaba en un mismo nodo es lo que se denomina un ciclo. Un ciclo se denomina sencillo so ninguna arista del ciclo aparece más de una vez en el camino. Un ciclo se denomina elemental si no pasa por ningún nodo más de una vez.

Definición 9: Un digrafo sencillo que no tenga ningún ciclo, se denomina acíclico.

Definición 10: Sea G=(N,A) un digrafo Sencillo. Entonces se define la relación de camino, C de G en la forma C={ (u, v) / existe un camino de u al nodo v.

Definición 11: En un grafo sencillo no dirigido, una sucesión $< V_1, V_2, V_3, ...,V_d >$ forma un camino si para i= 1,2,3.....d existe una arista no dirigida $\{V_{i-1}, V_i\}$. Se dice que la arista $\{V_{i-1}, V_i\}$

se encuentra en el camino. La longitud del camino está dada por en número de aristas que haya en el camino y que es d-1. Si $V_1 = V_d$, entonces el camino forma un ciclo.

Definición 12: Se dice que un grafo es conexo si para cualquier pareja de nodos del grafo, se puede llegar hasta el otro nodo, partiendo de uno de ellos.

Definición 13: Un digrafo sencillo se dice unilateralmente conexo si para toda pareja de nodos del grafo, al menos uno de los nodos de esa pareja se puede alcanzar desde el otro. Si para toda pareja de nodos del grafo los dos nodos de la pareja se pueden alcanzar uno desde el otro entonces se dice que el grafo es fuertemente conexo.

Definición 14: Un subgrafo G_i se denomina maximal con respecto a alguna propiedad, si no hay ningún otro subgrafo que también posea esa propiedad y que incluya a G_i . Para un digrafo sencillo, los subgrafos maximales fuertemente conexos se denominan componentes fuertes. De manera similar, un subgrafo maximal unilateralmente conexo o un subgrafo maximal débilmente conexo se denominan componente unilateralmente o componente débil, respectivamente.

BIBLIOTECA UIS

4. ANEXO: DESCRIPCIÓN DEL ENTORNO PROTOTIPO DE ESFOSIN

Básicamente el prototipo ESFOSIN consta de un Editor de diagramas entidad relación, un editor de diagramas de estados, un generador de especificaciones formales y múltiples asistentes para la descripción de los componentes en particular.

La figura 5 describe el entorno general de la herramienta. Contiene características típicas de un entorno Windows[47]. El menú principal consta de 5 opciones:

Archivo: Consta de las opciones típicas de nuevo proyecto, abrir, guardar, guardar como, imprimir, propiedades generales[48] y salir.

Ver: Permite crear un modelo entidad relación de alto nivel.

Generar: Opciones para la generación de la especificación formal en VDM++, la generación de tablas en un motor de bases de datos[49].

Ventana: Permite administrar y organizar las ventanas[50] .

Ayuda: Información sobre la construcción del prototipo.

La figura 6 describe dos ventanas, dentro del entorno de ESFOSIN, sobre las cuales se están editando diagramas de estado y de entidad relación. La figura 7 muestra la ventana de propiedades de las entidades. Las propiedades generales contienen el nombre, la descripción y el tipo de

---

[47] Ventanas (zonas de edición), menú principal, barra de herramientas, barra de estados y menús emergentes (activados con el click derecho).

[48] Esta ventana permite definir, propiedades generales del sistema a especificar, tales como nombre, objetivos, comentarios generales, criterios de validación y condiciones generales.

[49] Aunque esta opción no está activa en el prototipo de ESFOSIN, se pueden generar sentencias SQL para la creación de las tablas. Esta sentencias utilizarán parte de la especificación formal.

[50] ESFOSIN permite el manejo de múltiples ventanas sobre las cuales se editan diferentes tipos de diagramas entidad relación y/o diagramas de estado.

entidad. La sección de atributos permite asignar características[51] a las entidades y crear nuevos atributos.

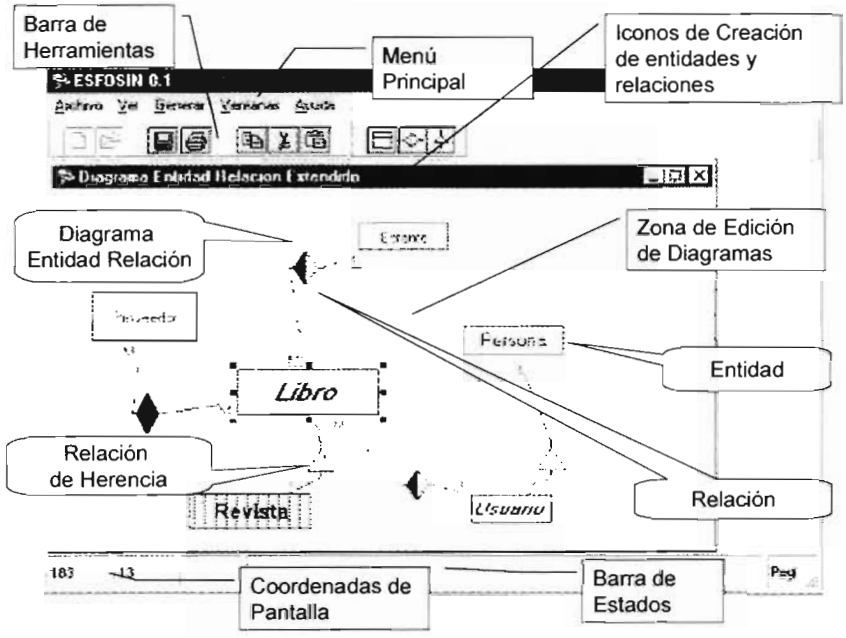

Figura 5 Descripción general del entorno de ESFOSIN

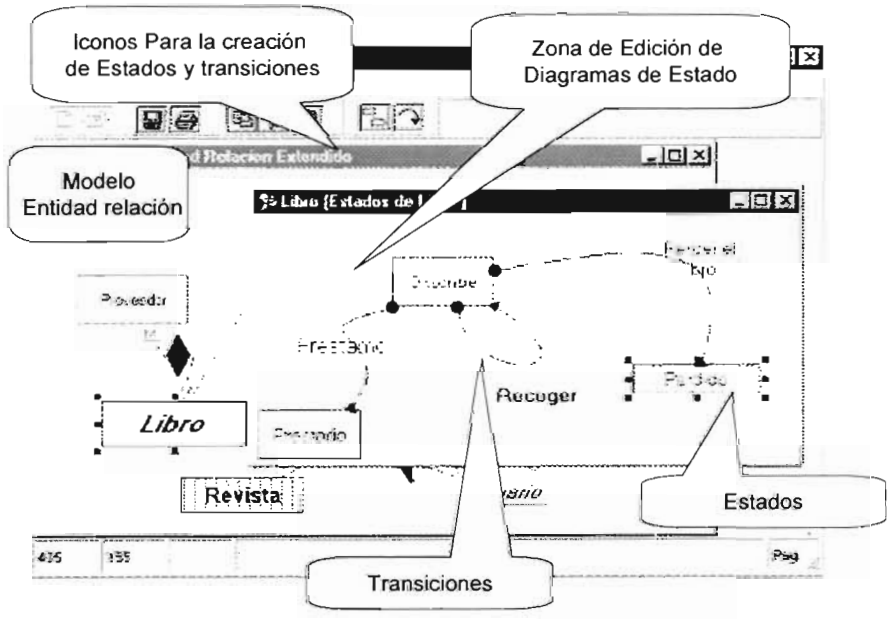

Figura 6 Editor de diagramas

[51] Estas características representan los atributos primarios, secundarios y disponibles o no asignados.

BIBLIOTECA UIS

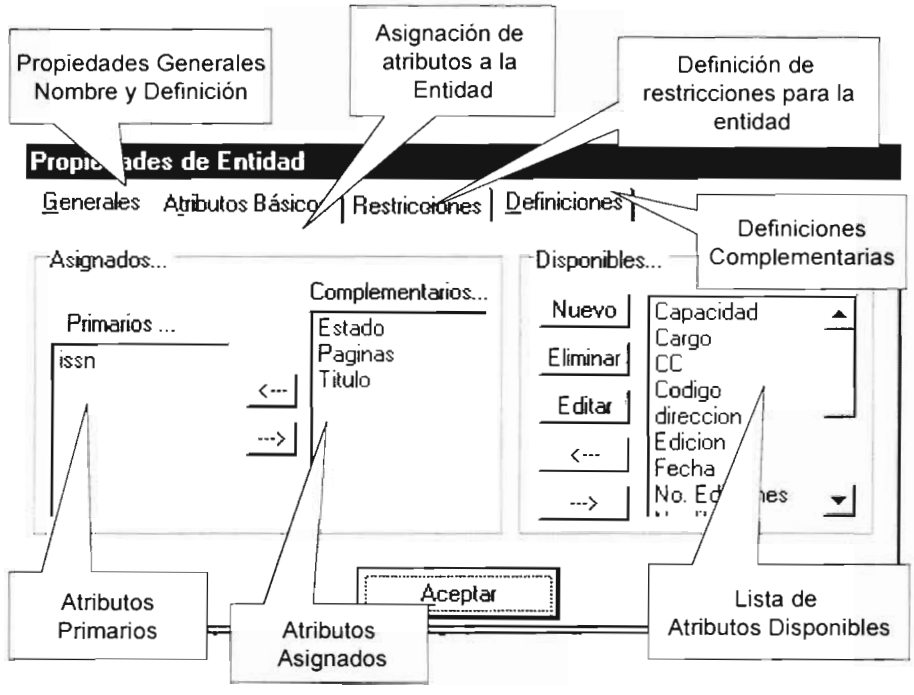

Figura 7 Edición de propiedades de Entidad

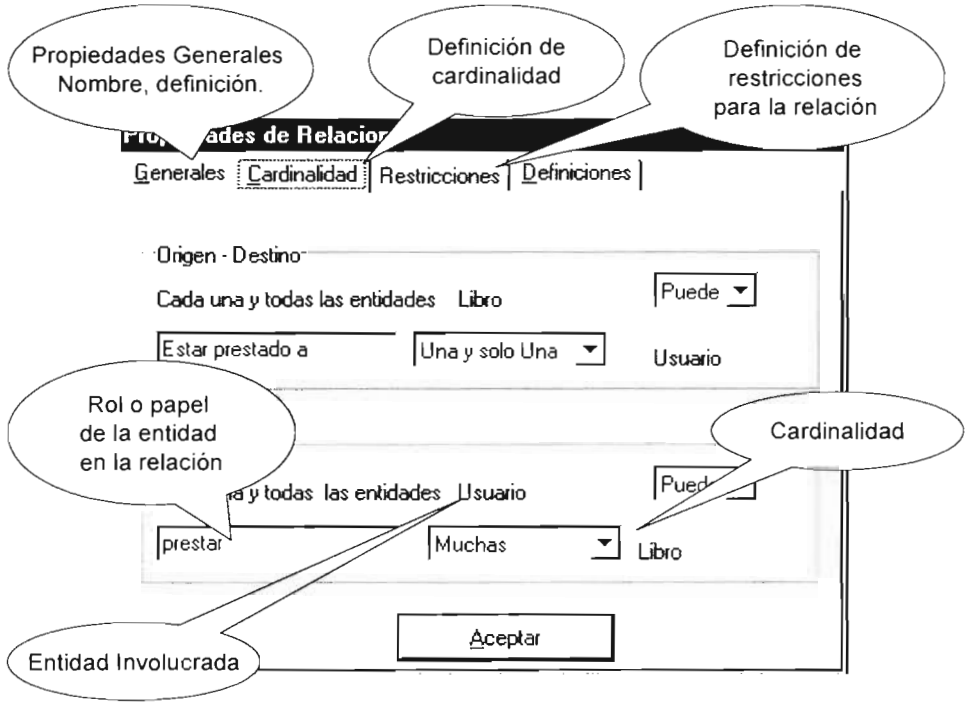

Figura 8 Edición de propiedades de relaciones

La figura 8 muestra la ventana de descripción de las relaciones. Las propiedades generales definen un nombre y una descripción de la relación.

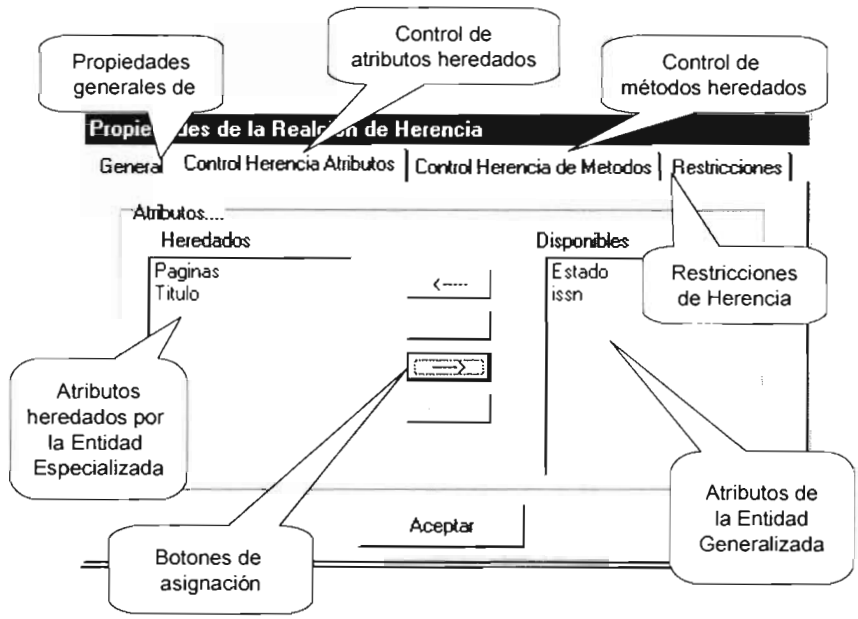

Figura 9 Editor de propiedades de relaciones de herencia

La figura 9 muestra la ventana de descripción de las relaciones de herencia. Las propiedades generales hacen explícita la interpretación de la relación y permite la edición de un comentario. La sección de Control de herencia muestra un listado de atributos y métodos heredados por la entidad especializada

BIBLIOTECA UIS

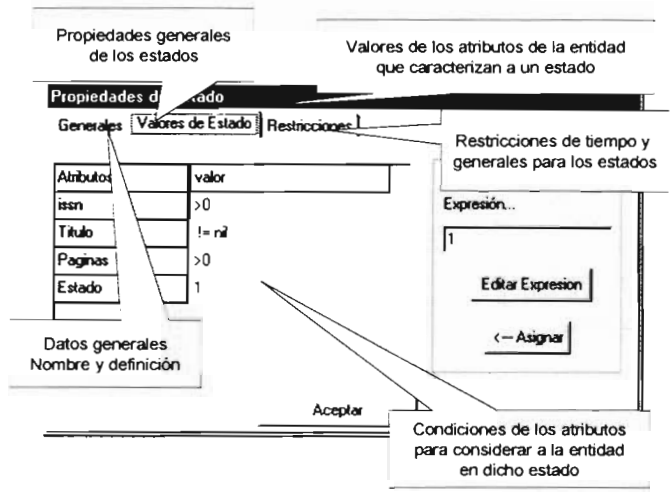

Figura 10 Editor de Propiedades de Estados

La figura 10 muestra la ventana de propiedades de los estados. La sección de valores de estado, contiene un listado de los atributos de la entidad motivo de descripción, con sus valores característicos o con una condición para dicho valor.

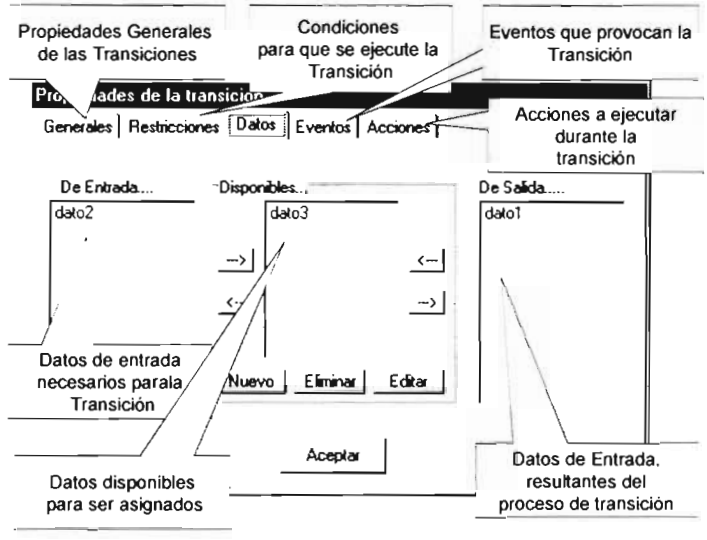

Figura 11 Editor de Propiedades de Transiciones

La figura 11 muestra la ventana de propiedades de las transiciones. La sección de prerequisitos contiene un listado de formulas lógicas[52]. La sección de datos contiene un listado clasificado en

[52] Que son editadas con un asistente. Estas formula están en función de los atributos de la entidad en cuestión.

7791

disponibles, de entrada o de salida. La última sección contiene un listado de formulas lógicas que representan eventos y un listado de expresiones verbales que representan acciones a ejecutar durante la transición.

La figura 33 muestra la ventana de descripción general del sistema a especificar. Contiene la definición del nombre, comentarios, objetivos, recomendaciones y criterios de validación del sistema.

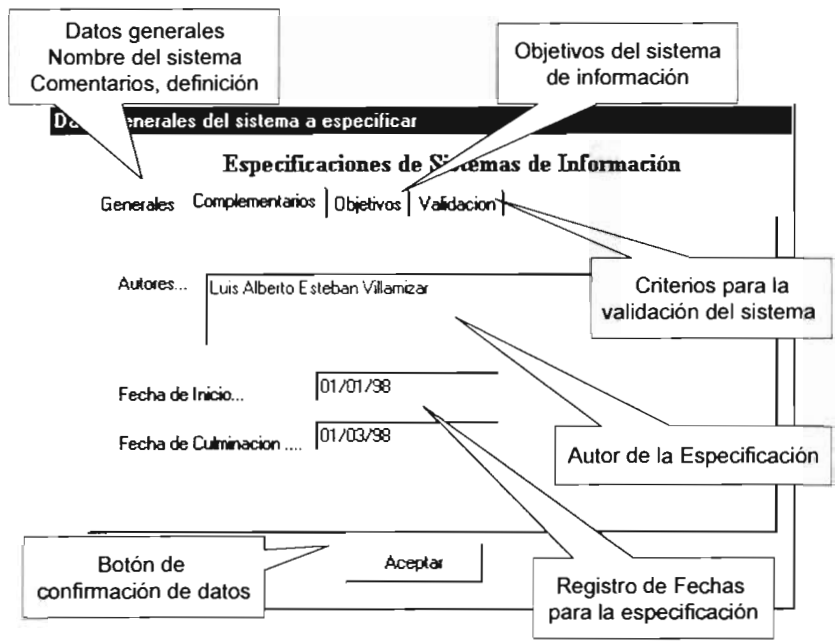

Figura 12 Edición de propiedades generales del sistema

90016