

Instituto Tecnológico de Costa Rica  
Escuela de Ingeniería en Computación  
Programa de Maestría en Computación



Integración efectiva de paradigmas de  
especificación de software

Tesis para optar al grado de Magister Scientiae en Computación

Ing. Luis Fernando Espino Barrios

Cartago, Costa Rica

Noviembre, 2010

# Integración efectiva de paradigmas de especificación de software

## RESUMEN

La especificación como una forma de descripción del comportamiento del software se ha vuelto un tanto subjetiva; esta subjetividad ha conllevado la creación de diferentes paradigmas de especificación, siendo los dos paradigmas más importantes: el paradigma que utiliza modelos diagramáticos y el paradigma que utiliza la descripción formal.

Ambos paradigmas han evolucionado de una manera separada; por un lado, se tiene la facilidad de elaborar diagramas y comunicar visualmente, por otro lado la posibilidad de describir formalmente el software mediante notaciones matemáticas precisas. Debido a esta separación es necesario establecer un marco de referencia que promueva la integración entre estos paradigmas de manera que se pueda utilizar las ventajas de ambos.

Tal marco de referencia requiere conciliar la especificación diagramática y la formal. Para establecer este marco conviene delimitarlo a un área de trabajo que sea común a cualquier esfuerzo de desarrollo de software; la parte más idónea es la base de la arquitectura del software que se refiere al modelado de datos.

El modelado de datos se hace comúnmente por medio del modelo entidad-relación y permite establecer asociaciones entre entidades que tienen atributos. Es importante destacar que dentro del modelo entidad-relación existen diferentes notaciones, siendo las más utilizadas las que manejan cardinalidades máximas y mínimas. La parte estática del modelado de datos corresponde a la definición de datos en lenguaje Z; además, se puede especificar la parte dinámica por medio de esquemas de operaciones en Z y que el usuario pueda extender la especificación en lo que se refiere a invariantes, precondiciones y postcondiciones.

Debido a lo anterior, este trabajo propone un marco de referencia común para el modelo entidad-relación y el lenguaje Z que permite integrar de manera efectiva ambos paradigmas de especificación. Para comprobar dicho marco de referencia se desarrolló una herramienta de software basada en Web que facilita hacer la transformación automática del modelo entidad-relación a su equivalente en lenguaje Z en dos formatos: en PDF para una vista inmediata y en TEX para permitir aumentar la especificación y utilizar algún verificador de tipos como FUZZ, así como su edición con formatos precisos para la publicación.

**Palabras claves:** Métodos de especificación de software, modelo entidad-relación, lenguajes formales, lenguaje Z, integración de paradigmas.

# Integración efectiva de paradigmas de especificación de software

## ABSTRACT

The specification as a way of describing the behavior of software has become somewhat subjective; this subjectivity has led to the creation of different specification paradigms, the two major ones being diagrammatic models and formal description.

Both paradigms have evolved separately; on the one hand, it is easy to draw diagrams and to communicate visually. On the other hand, there is the opportunity to formally describe the software using precise mathematical notations. Because of this separation, it is desirable to establish a framework that promotes the integration of these paradigms in order to use the advantages of both.

Such a framework requires reconciling diagrammatic and formal specifications. To establish this connection, it is convenient to delimit it to an area of software development and the most suitable one is data modeling which is one of the foundations of software architecture.

Data modeling here is done using the entity-relationship model, which comprises associations between entities that have attributes. There are several entity-relationship modeling notations, the most popular being those that let specify maximum and minimum cardinalities in associations. The static part of data modeling corresponds to the definition of data in  $Z$  (with schema types). Furthermore, the functional requirements can be specified through operations in  $Z$ . The user can then extend the specification in regard to invariants (of data types), preconditions and post conditions (of operations).

This document proposes a common framework for entity-relationship visual modeling and formal description in the Z language that allows effectively integrating both paradigms of specification. To validate this framework, we developed a Web-based software tool that facilitates to automatically translate entity-relationship models into their Z-language equivalents, using two formats: PDF for immediate reading and TEX, in order to enable augmenting the specification and using tools such as the FUZZ type-checker and tools for precise editing of documents in formats suitable for publication.

**Keywords:** Software specification methods, entity-relationship model, formal languages, Z notation, paradigm integration.

## APROBACIÓN DE LA TESIS

### “INTEGRACIÓN EFECTIVA DE PARADIGMAS DE ESPECIFICACIÓN DE SOFTWARE”

#### TRIBUNAL EXAMINADOR



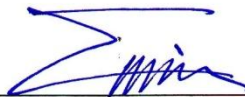
M.Sc. Ignacio Trejos Z.  
Profesor Asesor



Master Alicia Salazar H.  
Profesor Lector



M.Sc. María Lourdes Montes L.  
Profesional Externo



M.Sc. Edwin Aguilar S.  
Coordinador/Programa  
De Maestría

Noviembre, 2010

*A Dios por haberme permitido concluir este trabajo.*  
*A mis padres Augusto, Almita y Elieth por sus consejos y ejemplo.*  
*A mi esposa Lindsay por el amor y apoyo incondicional.*  
*A mis hijas Luisita y Ale por brindar una esperanza en mi existencia.*  
*A Tito que siempre fue mi ejemplo y lo seguirá siendo aunque no esté físicamente.*  
*A mis hermanos para que les sirva de ejemplo a seguir.*  
*A la familia de mi esposa que se ha convertido en mi familia.*  
*A mis amigos centro y sudamericanos por su apoyo moral.*

Agradezco al Servicio Alemán de Intercambio Académico –DAAD– por haberme brindado su confianza y apoyo para estudiar la Maestría en Computación. Al Instituto Tecnológico de Costa Rica –TEC– por haberme proporcionado una casa de estudios de alta calidad. Y a mi alma máter la Universidad de San Carlos de Guatemala –USAC– por haberme dado una base educativa sólida, en especial al Ing. Marlon Pérez y al Ing. Murphy Paiz por haberme apoyado durante la estadía en Costa Rica.

También agradezco a mi mentor M.Sc. Ignacio Trejos por brindarme su ayuda, tiempo y experiencia durante la elaboración de este trabajo de investigación y de sus lecciones que fueron de mucho beneficio. A todos mis profesores del TEC que me formaron académicamente. Y a las personas que intervinieron en la defensa de mi tesis: M.Sc. Alicia Salazar, M.Sc. María Lourdes Montes y al M.Sc. Edwin Aguilar.



# ÍNDICE

1.	INTRODUCCIÓN .....	1
1.1.	Introducción .....	1
1.2.	Definición del problema .....	2
1.3.	Objetivos .....	2
1.3.1.	Objetivo general.....	2
1.3.2.	Objetivos específicos .....	3
1.4.	Justificación .....	3
1.5.	Enfoque metodológico .....	4
1.6.	Alcances y limitaciones .....	5
1.7.	Descripción de los capítulos .....	6
2.	MARCO TEÓRICO .....	8
2.1.	Bases de datos relacionales.....	8
2.1.1.	Introducción a las bases de datos.....	8
2.1.2.	Arquitectura de bases de datos.....	8
2.1.3.	Modelos de bases de datos.....	10
2.1.4.	Utilidad del estándar XML para la integración .....	20
2.2.	Métodos de especificación formal .....	24
2.2.1.	Lenguajes formales de especificación de software.....	25
2.2.2.	Z: Lenguaje formal de especificación.....	26
2.3.	Herramientas para integrar paradigmas .....	27
2.3.1.	Marco de referencia entre UML y Z.....	28
2.3.2.	Integración de modelos estructurados y formales.....	28
2.3.3.	Combinación y refinamiento entre UML y Z .....	29

2.3.4.	Integración de UML y Z por medio de RoZ.....	29
2.3.5.	Biblioteca ER-Z.....	30
3.	INTEGRACIÓN DE PARADIGMAS DE ESPECIFICACIÓN DE SOFTWARE.....	31
3.1.	Definición de datos .....	31
3.2.	Definición de asociaciones .....	34
3.2.1.	Caso 1: Uno a muchos, ambos opcionales.....	35
3.2.2.	Caso 2: Uno a muchos, padre opcional.....	37
3.2.3.	Caso 3: Uno a muchos, hijo opcional .....	39
3.2.4.	Caso 4: Uno a muchos, ambos obligatorios.....	40
3.2.5.	Caso 5: Uno a uno, ambos opcionales .....	42
3.2.6.	Caso 6: Uno a uno, hijo o padre opcional.....	43
3.2.7.	Caso 7: Uno a uno, ambos obligatorios .....	45
3.2.8.	Caso 8: Muchos a muchos, ambos opcionales.....	46
3.2.9.	Caso 9: Muchos a muchos, hijo o padre opcional .....	48
3.2.10.	Caso 10: Asociaciones reflexivas .....	49
3.3.	Descripciones de vaciedad y nulidad.....	50
3.3.1.	Manejo de la vaciedad .....	50
3.3.2.	Manejo de la nulidad.....	51
3.4.	Definición del sistema de base de datos .....	52
3.5.	Definición de operaciones básicas.....	53
3.5.1.	Operación agregar.....	54
3.5.2.	Operación eliminar .....	54
3.5.3.	Operación actualizar .....	55
3.5.4.	Operación consultar .....	56
3.5.5.	Mensajes de respuesta.....	56

3.5.6.	Operaciones completas .....	59
4.	ANÁLISIS Y DISEÑO DE UNA HERRAMIENTA PARA INTEGRAR PARADIGMAS DE ESPECIFICACIÓN DE SOFTWARE.....	61
4.1.	Introducción.....	61
4.1.1.	Propósito .....	61
4.1.2.	Alcance .....	62
4.1.3.	Definiciones, acrónimos y abreviaturas.....	62
4.1.4.	Referencias.....	63
4.1.5.	Descripción del documento .....	63
4.2.	Descripción general .....	63
4.2.1.	Perspectiva del producto.....	63
4.2.2.	Funciones del producto.....	64
4.2.3.	Características de usuario .....	65
4.2.4.	Restricciones.....	65
4.2.5.	Supuestos y dependencias.....	66
4.3.	Requerimientos específicos .....	66
4.3.1.	Requerimientos de interfaz externo .....	66
4.3.2.	Requerimientos funcionales.....	67
4.3.3.	Requerimientos no funcionales.....	70
4.4.	Decisiones de diseño.....	75
4.4.1.	Decisiones respecto de los requerimientos no funcionales.....	75
4.4.2.	Notación del modelo entidad-relación utilizada .....	76
4.4.3.	Descripción del formato de entrada XML .....	78
4.4.4.	Ejemplo de notación ER y formato de entrada .....	80
4.5.	Arquitectura del software.....	82

4.5.1.	Representación arquitectónica .....	82
4.5.2.	Objetivos de arquitectura y limitaciones .....	83
4.5.3.	Vista de casos de uso .....	84
4.5.4.	Vista lógica .....	85
4.5.5.	Vista de proceso.....	85
4.5.6.	Vista de despliegue .....	86
4.5.7.	Vista de implementación .....	87
5.	DESARROLLO DE LA HERRAMIENTA DE SOFTWARE .....	89
5.1.	Enfoque de desarrollo .....	89
5.1.1.	Diagramador entidad-relación .....	90
5.1.2.	Convertidor de entidad-relación a lenguaje Z.....	93
5.2.	Decisiones de implementación .....	95
5.2.1.	Paradigma de programación .....	95
5.2.2.	Lenguaje y plataforma de programación .....	95
5.3.	Utilización de bibliotecas.....	97
5.3.1.	Tutoriales utilizados.....	97
5.3.2.	Biblioteca gráfica 2D para JavaScript .....	97
5.3.3.	Biblioteca FPDF para PHP .....	98
5.4.	Plan y ejecución de pruebas.....	99
5.4.1.	Prueba de contenido.....	100
5.4.2.	Prueba de interfaz .....	101
5.4.3.	Prueba de navegación .....	103
5.4.4.	Prueba de componentes .....	103
5.4.5.	Prueba de configuración .....	104
5.4.6.	Prueba de seguridad .....	105

5.4.7.	Prueba de desempeño.....	106
6.	CASO DE ESTUDIO .....	112
6.1.	Requerimiento.....	112
6.2.	Modelo entidad-relación .....	113
6.3.	Elaboración del diagrama entidad-relación en la herramienta ERZ .....	115
6.4.	Transformación e integración del modelo entidad-relación a lenguaje Z .....	119
6.4.1.	Generación de lenguaje Z en formato PDF .....	119
6.4.2.	Generación de lenguaje Z en formato TEX.....	126
6.4.3.	Verificación de tipos y extensión de la especificación .....	127
7.	ANÁLISIS DE RESULTADOS .....	136
7.1.	Discusión .....	136
7.2.	Dificultades encontradas.....	136
7.3.	Nivel de satisfacción de objetivos .....	137
7.4.	Hallazgos encontrados .....	138
8.	CONCLUSIONES Y RECOMENDACIONES .....	139
8.1.	Conclusiones.....	139
8.2.	Recomendaciones .....	139
9.	REFERENCIAS BIBLIOGRÁFICAS .....	141
10.	APÉNDICE: TIENDA DE VIDEO.....	144
10.1.	Modelo entidad-relación de la tienda de video.....	144
10.2.	Lenguaje Z en PDF generado a partir del modelo entidad-relación .....	144
10.3.	Lenguaje Z en TEX generado a partir del modelo entidad-relación.....	158
10.4.	Documento generado en Latex .....	169

## ÍNDICE DE FIGURAS

FIGURA No. 1.1 Teoría de ajuste cognitivo .....	5
FIGURA No. 2.1 Ejemplo de un diagrama entidad-relación .....	11
FIGURA No. 2.2 Ejemplo de entidad según la notación Barker.....	13
FIGURA No. 2.3 Ejemplo de una asociación según notación Barker.....	13
FIGURA No. 2.4 Ejemplo de entidad según notación IE.....	15
FIGURA No. 2.5 Comparación entre notación IE y Barker.....	15
FIGURA No. 2.6 Casos de asociaciones de uno a muchos con la notación ESB .....	18
FIGURA No. 2.7 Atributos y tuplas de una relación [14].....	19
FIGURA No. 2.8 Generación de especificación en Z en formato Latex con RoZ [27] .....	29
FIGURA No. 3.1 Ejemplo de entidad 1.....	32
FIGURA No. 3.2 Ejemplo de entidad 2.....	33
FIGURA No. 3.3 Caso 1: Asociación 01..0N.....	35
FIGURA No. 3.4 Diagrama de correspondencia del caso 1 .....	36
FIGURA No. 3.5 Diagrama de correspondencia inversa del caso 1 .....	36
FIGURA No. 3.6 Caso 2: Asociación 01..1N.....	38
FIGURA No. 3.7 Diagrama de correspondencia inversa del caso 2 .....	38
FIGURA No. 3.8 Caso 3: Asociación 11..0N.....	39
FIGURA No. 3.9 Diagrama de correspondencia inversa del caso 3 .....	40
FIGURA No. 3.10 Caso 4: Asociación 11..1N.....	41
FIGURA No. 3.11 Diagrama de correspondencia inversa del caso 4 .....	41
FIGURA No. 3.12 Caso 5: Asociación 01..01.....	42
FIGURA No. 3.13 Diagrama de correspondencia del caso 5 .....	42
FIGURA No. 3.14 Caso 6: Asociación 11..01.....	44
FIGURA No. 3.15 Diagrama de correspondencia del caso 6.....	44
FIGURA No. 3.16 Caso 7: Asociación 11..11.....	45
FIGURA No. 3.17 Diagrama de correspondencia del caso 7 .....	45
FIGURA No. 3.18 Caso 8: Asociación 0N..0N.....	46
FIGURA No. 3.19 Diagrama de correspondencia del caso 8 .....	47

FIGURA No. 3.20 Asociación 0N..1N .....	48
FIGURA No. 3.21 Diagrama de correspondencia del caso 9 .....	48
FIGURA No. 3.22 Asociación reflexiva 01..0N.....	49
FIGURA No. 4.1 Diagrama de estados .....	64
FIGURA No. 4.2 Casos de Uso .....	69
FIGURA No. 4.3 Ejemplo de una entidad en el esquema conceptual.....	77
FIGURA No. 4.4 Ejemplo de asociación de uno a muchos opcional.....	78
FIGURA No. 4.5 Modelo entidad-relación de ejemplo.....	81
FIGURA No. 4.6 Vista de casos de uso primarios .....	84
FIGURA No. 4.7 Diagrama de paquetes del software .....	85
FIGURA No. 4.8 Diagrama de actividades del software .....	86
FIGURA No. 4.9 Diagrama de despliegue del software .....	86
FIGURA No. 4.10 Arquitectura de tres capas de la herramienta de software.....	87
FIGURA No. 5.1 Menú contextual de entidad .....	90
FIGURA No. 5.2 Opciones para agregar atributos.....	91
FIGURA No. 5.3 Opciones para agregar cardinalidad a cierta relación .....	91
FIGURA No. 5.4 Ejemplo para mostrar el almacenamiento .....	92
FIGURA No. 5.5 Formato de salida en PDF .....	94
FIGURA No. 5.6 Formato de salida en TEX .....	94
FIGURA No. 5.7 Segmento de diagrama publicado por jsDraw2D [35].....	98
FIGURA No. 5.8 Logo de la biblioteca FPDF [36].....	98
FIGURA No. 5.9 Resultados de la verificación de HTML de la W3C .....	102
FIGURA No. 5.10 Resultados de la verificación de CSS de la W3C .....	102
FIGURA No. 5.11 Resultados de la prueba automática de seguridad del cliente .....	105
FIGURA No. 5.12 Resultados de la prueba automática de seguridad del servidor.....	106
FIGURA No. 6.1 Modelo entidad-relación de la tienda de video .....	113
FIGURA No. 6.2 Segmento usuario-copia_video del ER .....	114
FIGURA No. 6.3 Segmento video-copia_video del ER .....	114
FIGURA No. 6.4 Segmento banda_precio-video del ER.....	115
FIGURA No. 6.5 Opciones principales de la herramienta ERZ.....	115
FIGURA No. 6.6 Cuadro de diálogo para introducir el nombre de una entidad .....	116

FIGURA No. 6.7 Vista temporal de las entidades agregadas.....	116
FIGURA No. 6.8 Menú contextual de una entidad .....	116
FIGURA No. 6.9 Pantalla de ingreso de datos para una atributo .....	117
FIGURA No. 6.10 Ingreso de la cardinalidad de la entidad padre .....	117
FIGURA No. 6.11 Asociación ente usuario y copia de video .....	118
FIGURA No. 6.12 Modelo entidad-relación de la tienda de video .....	118
FIGURA No. 6.13 Cuadro de diálogo al almacenar un modelo en XML .....	119
FIGURA No. 6.14 Opciones de transformación de la herramienta ERZ .....	119
FIGURA No. 6.15 Verificación del caso de estudio con FUZZ.....	128
FIGURA No. 6.16 Inserción de un error en la especificación.....	128
FIGURA No. 6.17 Resultado de la ejecución de FUZZ con videoshop_mod.....	135



## ÍNDICE DE CUADROS

CUADRO No. 1.1 Matriz de relación .....	7
CUADRO No. 5.1 Resultados de los casos de pruebas de contenido .....	100
CUADRO No. 5.2 Resultados de los casos de prueba de componentes .....	103
CUADRO No. 5.3 Tamaños totales de objetos de prueba.....	107
CUADRO No. 5.4 Tiempo de descarga .....	107
CUADRO No. 5.5 Tiempo de carga de usuario .....	109
CUADRO No. 5.6 Tiempo de carga acumulada .....	110
CUADRO No. 5.7 Utilización de ancho de banda .....	110
CUADRO No. 5.8 Comparación de tiempo de carga.....	111

# 1. INTRODUCCIÓN

## 1.1. Introducción

En esta tesis el tema central es la integración de la especificación formal y la especificación por medio de modelos diagramáticos para el desarrollo de software. En esta investigación se tratan específicamente el modelo de datos entidad-relación y el lenguaje formal Z, como base para una integración apoyada por herramientas.

La especificación formal basada en el lenguaje Z tiene relación directa con el modelo de datos, debido a que cada entidad modelada corresponde a un esquema en lenguaje Z, y las asociaciones entre estas pueden ser modeladas mediante relaciones binarias.

La finalidad de la especificación formal es reducir el número y la severidad de los errores que pueden cometerse durante el desarrollo de un determinado artefacto de software, además de proporcionar de una manera formal la especificación para su posterior desarrollo. Sin embargo, se percibe una alta complejidad en la mayoría de lenguajes formales; por eso, la tendencia a utilizar especificaciones orientadas a modelos diagramáticos se ha ido incrementando, lo que explica el éxito de UML.

Por lo anterior, es de mucha importancia realizar una integración de ambas maneras de especificar, obteniendo las ventajas de ambos enfoques si son apoyados con éxito por una herramienta de software apropiada.

Este trabajo de investigación se basa en una integración pragmática del modelo de datos relacional y el modelo formal de especificación. Particularmente trata del modelaje de datos entidad-relación, y del lenguaje formal Z, en su forma original propuesta por Spivey [1].

## **1.2. Definición del problema**

Hay una tendencia a utilizar especificaciones basadas en modelos diagramáticos, por su facilidad de utilización y comprensión. Sin embargo, tales modelos poseen cierto grado de ambigüedad e imprecisión. Dadas las limitaciones que las especificaciones diagramáticas presentan, resultaría provechoso realizar una integración entre las especificaciones basadas en modelos diagramáticos y las especificaciones basadas en lenguajes formales que cuentan con una semántica precisa.

Los métodos formales pretenden mejorar la calidad del software. Por ejemplo, el lenguaje Z está basado en la teoría de conjuntos y en la lógica de predicados de primer orden, siendo así una notación formal que permite describir y modelar muchos aspectos de los sistemas de información.

Se pretende apoyar el desarrollo de especificaciones comprensibles y precisas, combinando aspectos basados en modelos diagramáticos para facilitar su mejor comprensión junto con la especificación formal para la descripción precisa y verificable, y con ello mejorar las probabilidades de éxito en el desarrollo de productos de software. Esta integración puede verse facilitada al seguir estándares y apoyar el trabajo de modelaje con herramientas apropiadas.

## **1.3. Objetivos**

### **1.3.1. Objetivo general**

Establecer un marco de referencia entre el modelo entidad-relación y el lenguaje Z para garantizar la integración efectiva entre ambos paradigmas, además diseñar y construir una herramienta de software que permita elaborar modelos de datos mediante diagramas entidad-relación y generar automáticamente una especificación formal equivalente basada en el lenguaje Z, como base para especificar ulteriormente la funcionalidad.

### **1.3.2. Objetivos específicos**

- Elaborar un resumen documental de los métodos formales de especificación y los de especificación basada en modelos diagramáticos, como base para el desarrollo de una solución integradora.
- Realizar un estudio del estado del arte, para examinar características de otras herramientas de software similares, analizando sus ventajas y desventajas para obtener ventajas como valor agregado y disminuir las desventajas, de esta manera contribuir a aumentar la calidad de la solución.
- Realizar el análisis y diseño de una herramienta de software apropiada, siguiendo un método ágil de desarrollo, con algunos aspectos del Proceso Unificado, tal como los diagramas de casos de uso, por lo que implícitamente se considera la implementación en paralelo al diseño.
- Construir una herramienta que permita, como mínimo: diagramar modelos entidad-relación, generar descripciones en Z correspondientes a entidades y asociaciones del modelo entidad-relación, generar en Z una plantilla para el estado inicial del sistema y las operaciones básicas de manipulación de datos (agregar, modificar, eliminar y consultar).
- Desarrollar un caso de estudio que ilustre las capacidades de la herramienta y sea representativo de su dominio de aplicación.
- Realizar un análisis de resultados del software, considerando aspectos como los requerimientos no funcionales del sistema.

## **1.4. Justificación**

Debido al problema generado por las limitaciones que los enfoques de especificación diagramáticos y formales poseen, se propone una integración de ellos que busca neutralizar sus desventajas y aprovechar al máximo las ventajas de cada uno.

La herramienta de software propuesta posee, como característica innovadora, que un usuario pueda elaborar un diagrama entidad-relación y que, automáticamente, a partir del diagrama podrá generar una descripción básica en lenguaje Z, que formaliza las entidades y relaciones del modelo diagramático, ofreciendo las bases para desarrollar formalmente la especificación funcional.

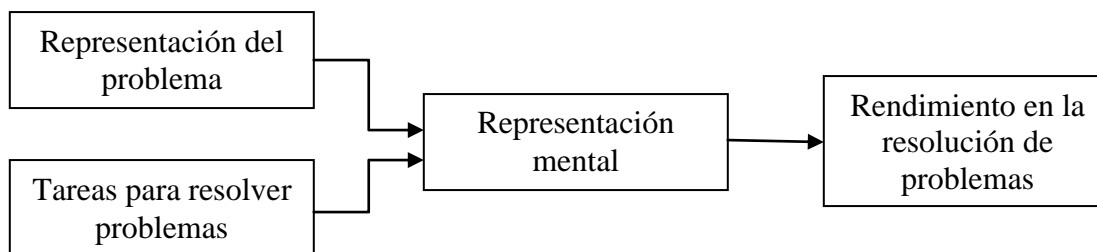
El impacto de la herramienta de software es crucial, debido a que es una herramienta de software de propósito general, es decir, cualquier persona que desea realizar un diagrama entidad-relación podrá generar su equivalente formal sin ser un experto en el tema, y la especificación además de tener modelos diagramáticos también tendrá un carácter formal por la especificación en lenguaje Z, lo que habilitará especificar formalmente las operaciones que constituyen la funcionalidad del software que se está trabajando.

Un aspecto importante es que se buscó diseñar y construir la herramienta de software siguiendo estándares de la Web; con esto se busca que la herramienta pueda ser utilizada desde cualquier navegador Web que siga tales estándares.

La perspectiva de profundidad del software está delimitada por su propio alcance; como se mencionó, se limita el software a diagramar modelos de entidad-relación básicos y el lenguaje Z generado es el descrito por Spivey[1].

## **1.5. Enfoque metodológico**

La teoría que se utilizó como respaldo de la investigación es la llamada “Teoría de ajuste cognitivo” [2]. Esta teoría manifiesta que el rendimiento de una tarea puede ser mejorado cuando hay un ajuste cognitivo entre la información utilizada que describe la representación del problema y la información requerida por el tipo de tarea para solucionar el problema en consideración.



**FIGURA No. 1.1 Teoría de ajuste cognitivo**

Asociando la teoría con la investigación, la representación del problema se refiere a los diferentes paradigmas de especificación, es decir, los modelos diagramáticos y los modelos formales; mientras que las tareas de resolución de problemas se refieren a la sintaxis y semántica de cada paradigma de especificación.

Tal como se ha demostrado en [2] y [3], la representación del problema influye significativamente en la representación mental que la conceptualización de tareas y consecuentemente en el rendimiento de la resolución del problema. De la misma manera, la representación del problema desde el enfoque diagramático propone un mejor entendimiento debido a efectos visuales en la memoria y comprensión.

Por otro lado, para el software se utilizó un método de desarrollo ágil, complementado con algunos aspectos del Proceso Unificado.

## **1.6. Alcances y limitaciones**

Se definieron dos entregables principales: el marco de referencia y una herramienta de software que complemente dicho marco. Con respecto del modelo entidad-relación, solamente se diagramarán asociaciones binarias. La herramienta de software tiene carácter de prototipo funcional, ya que la intención es mostrar la integración de ambos paradigmas de especificación de software.

Tanto el paradigma como el lenguaje de programación se definieron en la fase de diseño de la herramienta de software. Tentativamente los lenguajes que se consideraron fueron PHP, Java, JavaScript y Flash.

El modelado es de tipo gráfico para facilitar la tarea de diagramar basado en la notación ESB [4] que muestra explícitamente la cardinalidad de las asociaciones. El lenguaje generado por la solución –datos de salida– está basado en un lenguaje de marcas como XML y/o Latex.

Además el lenguaje generado se limitó a producir: el diccionario de datos; los esquemas de entidades, instancias y asociaciones; el estado inicial del sistema; las operaciones básicas sobre los datos como agregar, eliminar, modificar y consultar.

## **1.7. Descripción de los capítulos**

Con base en la teoría utilizada se ha realizado un análisis de variables independientes y dependientes o factores que influyen en el rendimiento de resolución de problemas. Esta investigación trata de la integración efectiva de los diferentes paradigmas de especificación. Para exponer sus bases, desarrollo y resultados se presentan los capítulos siguientes:

- Marco teórico: Aquí se presenta toda la teoría en que se basará el trabajo posterior.
- Integración de paradigmas de especificación: En este capítulo se propone el marco de referencia.
- Análisis y Diseño de una herramienta de software: Se muestran los principales requerimientos.
- Desarrollo de la herramienta de software: Aquí se presentan decisiones de diseño y construcción de la herramienta, así como las pruebas realizadas.
- Caso de estudio: Este es un capítulo importante ya que muestra la funcionalidad tanto del marco de referencia como de la herramienta.

- Análisis de resultados: En este capítulo se discute el tema de la integración y el grado de cumplimiento de los objetivos.

**CUADRO No. 1.1 Matriz de relación**

	<b>Marco teórico</b>	<b>Integración de paradigmas de especificación</b>	<b>Diseño de una herramienta de software para integración</b>	<b>Desarrollo de la herramienta de software</b>	<b>Caso de estudio</b>	<b>Análisis de Resultados</b>
<i>Representación del problema</i>	<ul style="list-style-type: none"> <li>- Especificación diagramática (ER)</li> <li>- Especificación formal (Z)</li> </ul>					
<i>Tareas para resolver problemas</i>	<ul style="list-style-type: none"> <li>- Herramientas de integración, analizando sintaxis y semántica</li> </ul>					
<i>Representación mental</i>		<ul style="list-style-type: none"> <li>- Propuesta de integración de paradigmas de especificación</li> </ul>	<ul style="list-style-type: none"> <li>- Diseño de una herramienta para integrar paradigmas</li> </ul>	<ul style="list-style-type: none"> <li>- Desarrollo de la herramienta de integración</li> </ul>		
<i>Rendimiento en la resolución de problemas</i>					<ul style="list-style-type: none"> <li>- Realización de un caso de estudio probando los diferentes factores</li> </ul>	<ul style="list-style-type: none"> <li>- Presentación de resultados en cuanto al rendimiento proporcionado por la representación mental del paradigma</li> </ul>



## **2. MARCO TEÓRICO**

### **2.1. Bases de datos relacionales**

#### **2.1.1. Introducción a las bases de datos**

Se identifican tres conceptos importantes, las bases de datos, los sistemas de bases de datos y los sistemas administradores de bases de datos.

Una base de datos es un conjunto o colección de datos relacionados y almacenados para su posterior utilización; dicho conjunto de datos debe ser relevante para la organización.

Un sistema de base de datos es un sistema computarizado para archivar datos en una computadora [5], con el objetivo de mantener la información y disponer de ella cuando se necesite.

Un sistema gestor de bases de datos SGBD o DBMS consiste en una colección de datos interrelacionados y un conjunto de programas que maneja el acceso a dichos datos.

#### **2.1.2. Arquitectura de bases de datos**

Hay diferentes enfoques de arquitectura, en ANSI/SPARC [6] se definen tres niveles:

- El nivel interno: Siendo el más cercano al almacenamiento físico, es decir, es el que se encarga de la forma en que se guardan los datos en cualquier medio de almacenamiento físico, como los discos duros.

- El nivel externo: Es el más cercano a los usuarios, es decir, es la forma en los usuarios perciben la idea de la base de datos.
- El nivel conceptual: Es el mediador entre ambos niveles, que corresponde a la representación de la información de la base de datos tanto desde una vista externa como interna.

Halpin y Morgan [7] definen cuatro niveles:

- Nivel conceptual: Siendo el nivel fundamental, traslada conceptos del dominio de negocio hacia un nivel de conceptos humanos, es llamado el “esquema conceptual”, describe la estructura o gramática del dominio de negocio.
- Nivel externo: Especifica el diseño del dominio de negocio y las operaciones accesibles para un usuario particular o grupo de usuarios particulares, proveyendo un nivel de jerarquía de privilegios para modificar la base de datos.
- Nivel lógico: Para una aplicación dada se necesita de un modelo lógico apropiado, al seleccionarlo se mapea del esquema conceptual al esquema lógico expresado en términos de estructuras de abstracción para los datos y operaciones soportadas en el modelo de datos.
- Nivel físico: El último nivel es el físico en donde el esquema lógico es implementado en un DBMS específico, tal sería el caso de SQL Server, Oracle, etc. Este incluye detalles acerca del almacenamiento físico y el acceso a estructuras utilizadas en el sistema.

En lo que respecta a la investigación y el desarrollo de la herramienta de software, se definen tres niveles:

- Nivel conceptual
- Nivel lógico
- Nivel físico

### **2.1.3. Modelos de bases de datos**

#### **2.1.3.1. Modelo entidad-relación**

La idea básica del modelo entidad-relación propuesto por Chen [8] es conceptualizar la percepción de la cosas por medio de objetos. Esto se hace a partir de una representación de un modelo de datos que incorpora información semántica importante acerca del mundo real. Este modelo es la unificación de las ventajas los siguientes modelos: modelo de red, modelo relacional y el modelo de conjunto de entidades.

Barker menciona en [9] que el modelado entidad-relación es una técnica que define las necesidades de información de una organización. Provee una base firme para entregar sistemas apropiados y de alta calidad. El modelar implica la identificación de objetos importantes representados como entidades, las propiedades de esos objetos representados como atributos y describir cómo los objetos se relacionan unos con otros representados como asociaciones.

Chen [8] mencionó en su artículo las cardinalidades de las asociaciones; describió la restricción de utilizar relaciones binarias “1:N” que describe la existencia de “N” instancias de la entidad en un lado de la asociación que dependen de la existencia de “1” entidad en el otro lado de la relación desde un punto de vista de participación.

En el libro de modelado de Barker [9] se mencionan 10 puntos importantes que ayudan a definir la información necesitada:

1. Datos: Datos como recurso, últimamente los datos han ganado mucha importancia para el éxito de organizaciones.
2. Compromiso de gestión: El tiempo de gestión debe ser adquirido para confirmar los requerimientos de la información.
3. Convenciones: Convenciones, estándares y guías deben ser aplicados todo el tiempo, incluyendo los conceptos de normalización de datos.

4. Definición mínima: Cualquier concepto debe ser definido y modelado en una y solamente una manera.
5. Independencia de datos: Los requerimientos de información deben ser definidos de tal manera que sea independiente de cualquier almacenamiento eventual o método de acceso.
6. Patrones genéricos: Recurrir a patrones de datos para habilitar la mejor manera de procesar datos, que sea efectivo respecto del costo y que las estructuras sean flexibles.
7. Calidad y postura: Los modeladores deben aprender a aplicar convenciones sin sacrificar el rigor, ayudando a mejorar la precisión de los modelos.
8. Comunicación: La comunicación con usuarios finales debe estar en términos en que ellos entiendan.
9. Relevancia: Los requerimientos de información pueden ser de valor si se apoyan las necesidades funcionales de la organización.
10. Un medio, no un fin: Mientras que el modelado sea poderoso se otorga visibilidad dentro de la organización y actúa como un marco para el diseño de datos solo como una técnica intermedia.

El siguiente ejemplo basado en la figura 10 de [8] muestra un diagrama entidad-relación simple, indica que muchos empleados pueden participar en muchos proyectos y a su vez que en muchos proyectos pueden participar muchos empleados. Hay que notar que en ese momento la cardinalidad se encuentra definida pero no la opcionalidad, es decir, la cardinalidad mínima; en las próximas secciones se aclarará el tema.



**FIGURA No. 2.1 Ejemplo de un diagrama entidad-relación**

Es importante destacar la noción de un identificador[8], el cual identifica una única instancia o tupla –término que se utilizará posteriormente en la sección del modelo relacional– dentro de una entidad. Este identificador comúnmente se le llama “llave primaria” o “clave primaria”.

A veces es necesario más de un atributo para identificar una tupla, por tanto una llave primaria es un grupo de atributos que identifican una tupla única y determina funcionalmente a los demás atributos. En la teoría de las bases de datos se dice que los atributos de una entidad dependen funcionalmente de la llave primaria.

El concepto de identificador también sirve para identificar los pares ordenados que se generarán cuando se crea una asociación entre entidades. A nivel lógico la entidad hija adopta el identificador de la entidad padre para referenciar dicho objeto; a este identificador se le llama “llave foránea” o “clave foránea”. Esta llave foránea [10] surge por la necesidad de referenciar un objeto particular.

#### **2.1.3.2. Notaciones para modelos entidad-relación**

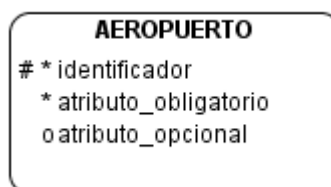
Existe un buen número de notaciones de modelos entidad-relación; sin embargo, en este trabajo solo se mostrarán tres notaciones más, que sus asociaciones se particularizan por referenciar al opuesto; es decir, la cardinalidad se lee al contrario que la notación de Chen; y porque están relacionadas con el trabajo expuesto más adelante.

En las notaciones para modelos entidad-relación es común utilizar los términos “padre” e “hijo”. Estos términos aparecieron en trabajos de Codd [10] y [11], indicando una jerarquía entre entidades. Esta jerarquía indica que las entidades padre proveen descripción a las entidades hijas formado un árbol de asociaciones.

### 2.1.3.2.1. Notación de Barker

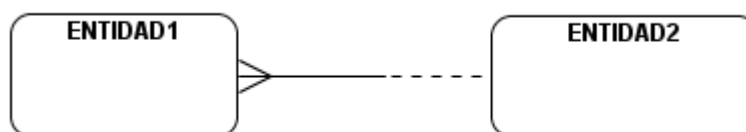
La notación de Barker [9] se basa en que una entidad es un objeto de importancia que contiene la información que se necesita saber; una entidad es representada de manera diagramática por un rectángulo con las esquinas redondeadas con nombre; el nombre debe ser singular y en mayúscula.

Cada entidad debe ser única y cada instancia también debe ser identificable de manera única, los atributos de cada entidad deben describirse con un símbolo “#” para un identificador, un símbolo “\*” para atributos obligatorios y un símbolo “o” para un atributo opcional.



**FIGURA No. 2.2 Ejemplo de entidad según la notación Barker**

Las asociaciones tienen dos terminales, cada una debe tener la cardinalidad que se refiere a cuántos y la opcionalidad que se refiere si una instancia es opcional u obligatoria. La terminación similar a una pata de gallo representa muchos, la terminación lineal representa uno, la opcionalidad continua significa que es obligatoria y la opcionalidad discontinua significa que es opcional.



**FIGURA No. 2.3 Ejemplo de una asociación según notación Barker**

La manera de leer el anterior diagrama es la siguiente:

- Cada instancia de la entidad1 DEBE estar asociada con UNA Y SOLAMENTE UNA instancia de la entidad2.
- Cada instancia de la entidad2 PUEDE estar asociada con UNA O MÁS instancias de la entidad1.

Cabe destacar que la notación de Barker se ha malentendido en cuanto a la interpretación de la cardinalidad y opcionalidad. La interpretación correcta según [9] es la mostrada anteriormente; esto significa que en una notación de cardinalidad mínima y máxima para una entidad la opcionalidad se interpreta al contrario como se muestra a continuación:

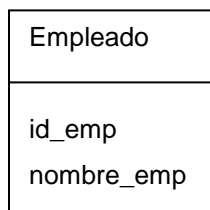
- Cada instancia de la entidad1 DEBE (1) estar asociada con UNA Y SOLAMENTE UNA (1) instancia de la entidad2. Esto significa que la cardinalidad explícita de la entidad2 debe interpretarse como (1..1).
- Cada instancia de la entidad2 PUEDE (0) estar asociada con UNA O MÁS (N) instancias de la entidad1. Esto significa que la cardinalidad explícita de la entidad1 debe interpretarse como (0..N).

Por tanto, la interpretación de esta notación debe hacerse, por un lado, leyendo de izquierda a derecha la opcionalidad y cardinalidad respectivamente para interpretar la cardinalidad explícita de la entidad2, y por otro lado, se debe leer de derecha a izquierda la opcionalidad y cardinalidad respectivamente para interpretar la cardinalidad explícita de la entidad1. En conclusión la interpretación cruzada de la cardinalidad y de la opcionalidad de esta notación presenta una inconsistencia que plantea una barrera cognitiva e induce a error.

#### 2.1.3.2.2. Notación IE

La notación IE [12] o *Information Engineering* es un enfoque que inicia con el trabajo de Clive Finkelstein en Australia y la CACI del Reino Unido; luego fue adoptada por James Martin. La principal desventaja es que existen diferentes versiones o no hay un estándar.

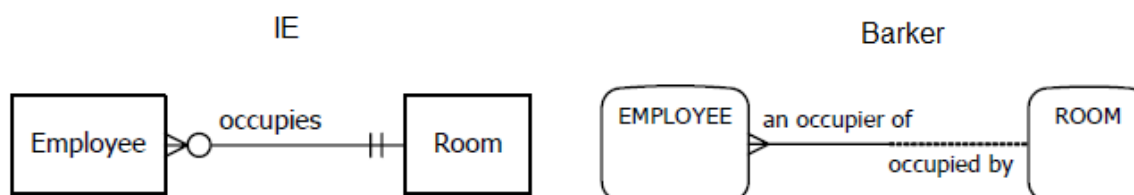
Las entidades se colocan en un rectángulo y separados con una línea se colocan los atributos.



**FIGURA No. 2.4 Ejemplo de entidad según notación IE**

Las asociaciones se representan como las cardinalidades explícitas pero siguiendo el patrón de pata de gallo, al contrario de la notación de Barker las cardinalidades se escriben del lado de la entidad, es decir, cada extremo de la asociación debe tener dos símbolos. Los símbolos son una línea corta perpendicular para indicar “uno”, un círculo para indicar “cero” y la pata de gallo para indicar “muchos”.

Para entender la interpretación de la notación IE y de la notación Barker se muestra una comparación extraída de [7]:



**FIGURA No. 2.5 Comparación entre notación IE y Barker**

Comparando las cardinalidades del diagrama anterior, es fácil notar que la opcionalidad mostrada en Barker del lado de la entidad “ROOM” es la misma opcionalidad mostrada en IE del lado de la entidad “Employee”.



Y como preámbulo para la siguiente notación cabe destacar que la cardinalidad explícita es la misma que en IE pero descrita con números o con la letra N como en la notación de Chen. La cardinalidad explícita de la entidad “Employee” es (0..N) y la cardinalidad explícita de la entidad “Room” es (1..1).

### 2.1.3.2.3. Notación ESB

La notación ESB [4] es la unificación de varios aspectos de otras notaciones; la necesidad de utilizar una notación que permite agregar semántica al modelo relacional ha llevado a la propuesta de esta notación, la idea principal es unificar las mejores prácticas de diferentes notaciones presentadas anteriormente.

La cardinalidad de las relaciones se pueden expresar a partir de la cardinalidad mínima y máxima que representa la cardinalidad y la opcionalidad respectivamente; esto similar a la notación UML y MIN-MAX. La cardinalidad mínima puede tener dos valores 0 ó 1, y la cardinalidad máxima también puede tener dos valores 1 ó N. Dado que en una relación binaria intervienen dos entidades, existen cuatro valores variables:

$$\text{CardMin1..CardMax1} - \text{CardMin2..CardMax2}$$

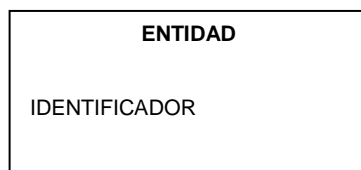
Esto produce 16 diferentes combinaciones posibles. Inicialmente la propuesta no permitía diseñar diagramas con asociaciones de muchos a muchos, porque esto provoca crear diagramas que no estén normalizados; pues la normalización es muy importante porque disminuye la redundancia, evita problemas de actualización y garantiza la integridad de datos. Sin embargo, para no plantear una barrera en el diseño las asociaciones de muchos a muchos es permitido actualmente.

Por un lado, los casos repetidos de cardinalidad y opcionalidad provocan una disminución de los 16 casos, y por otro lado, Barker [9] plantea un conjunto de asociaciones inválidas que también disminuyen los 16 casos; reduciéndolos de 16 a 10.

Los 10 casos que se podrán diagramar son los siguientes:

- Caso-1: De uno a muchos (ambos opcionales) 01..0N
- Caso-2: De uno a muchos (uno obligatorio) 11..0N
- Caso-3: De uno a muchos (muchos obligatorio) 01..1N
- Caso-4: De uno a muchos (ambos obligatorios) 11..1N
- Caso-5: De uno a uno (ambos opcionales) 01..01
- Caso-6: De uno a uno (alguno opcional) 11..01
- Caso-7: De uno a uno (ambos obligatorios) 11..11
- Caso-8: De muchos a muchos (ambos opcionales) 0N..0N
- Caso-9: De muchos a muchos (alguno opcional) 1N..0N
- Caso-10: Casos reflexivos (ambos opcionales) 01..0N ó 01..01

El esquema conceptual especifica la estructura para todos los estados permitidos y transiciones de la base de datos [12]. Siendo así una manera expresiva de representar un sistema del mundo real, hay que notar que en este esquema aún no se mencionan términos relacionales.



**FIGURA No. 2.1 Ejemplo de una entidad en el esquema conceptual.**

Entre algunos componentes del esquema conceptual están:

- Entidad: Es cualquier objeto distinguible. (Se representa como un rectángulo con nombre)
- Atributos: Son propiedades que corresponden a la entidad.
- Relación: Representa la asociación entre entidades.

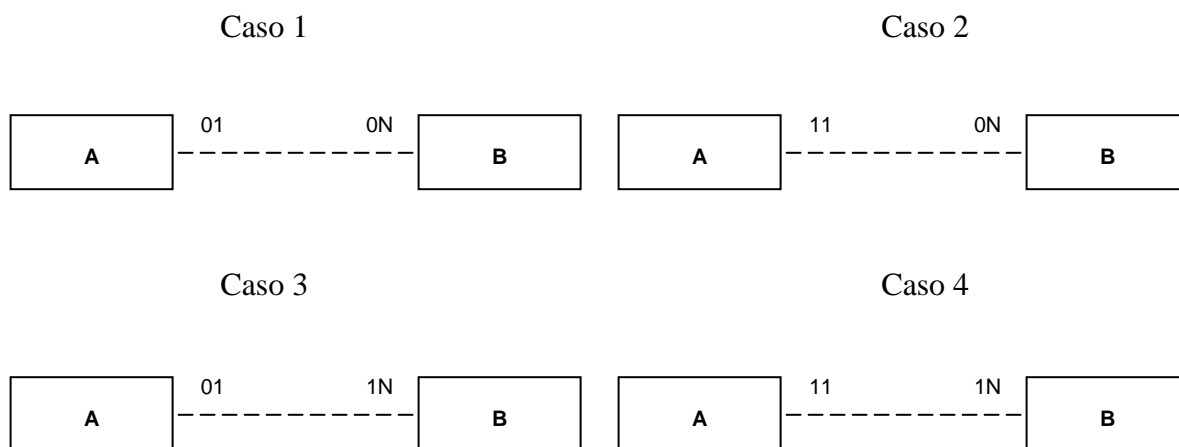
Un concepto importante es la dependencia de existencia. Aquí la definición de identificadores es importante, hay dos casos a considerar: un caso mencionado en [13] es cuando la entidad depende existencialmente de otra o llamada entidad débil, a este se le denominará “identificador dependiente”; y el segundo caso es cuando no depende, a este se le denominará “identificador independiente”, ambos casos se representan así:

Identificador Dependiente: \_\_\_\_\_

Identificador Independiente: - - - - -

Cabe mencionar que desde el punto de vista lógico del modelo se utiliza la denominación padre e hijo para entidades participantes en cuanto a la asociación se refiere, debido por un parte a la facilidad de identificación de entidades para ayudar al mapeo conceptual a lógico, y por otra parte a la jerarquía y dependencia de existencia que hay entre entidades.

A continuación se muestran algunas asociaciones de esta notación:



**FIGURA No. 2.6 Casos de asociaciones de uno a muchos con la notación ESB**

### 2.1.3.3. Modelo relacional

El modelo relacional representa la base de datos como un conjunto de relaciones; es el modelo que conceptualiza y semeja los datos y sus relaciones por medio de tablas. Cada tabla está compuesta de columnas y filas. Las columnas tienen nombres únicos y las filas se pueden identificar con una llave única.

Este es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), llegó a consolidarse como un nuevo paradigma en los modelos de base de datos [11]. Su idea fundamental es el uso de "relaciones".

En la terminología del modelo relacional se encuentran las tuplas que representan las filas, atributos o cabecera de columna y relación para una tabla; además de términos como dominio que es conjunto de valores atómicos, donde el grado de una relación es el número de atributos y la cardinalidad el número de tuplas.

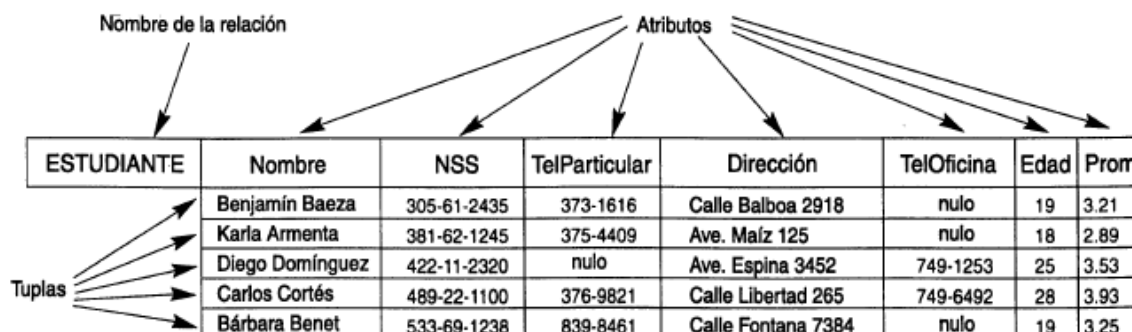


FIGURA No. 2.7 Atributos y tuplas de una relación [14]

#### **2.1.3.4. Mapeos entre niveles**

- Mapeo Lógico: Es construir un modelo lógico apropiado mapeándolo desde el esquema conceptual, expresándolo en términos de estructuras de abstracción para los datos y operaciones soportadas en el modelo de datos.  
Las relaciones se mapearán a tablas y las restricciones serán expresadas en términos de declaraciones de llaves primarias y foráneas.
- Mapeo Físico: Es construir una serie de instrucciones apropiadas mapeándolas desde el esquema lógico, es decir, se toman las tablas y restricciones y se transforman en instrucciones para un DBMS, generalmente este sería el script de la base de datos, en el cual el DBMS se encargará de almacenar nuestro modelo de manera física.

#### **2.1.4. Utilidad del estándar XML para la integración**

XML significa Lenguaje de Marcas Extensible, del inglés eXtensible Markup Language, es un conjunto de reglas para definir etiquetas semánticas que dividen un documento en partes e identifica las diferentes partes del documento [15].

XML está derivado de un lenguaje para estructurar documentos [16], conocido como lenguaje estándar generalizado de marcas, del inglés Standard Generalized Markup Language, SGML que también fue la base de HTML.

A continuación se describen las partes de un archivo XML específico con fin de intercambio basado en [17] que servirá posteriormente en el desarrollo de una herramienta integradora de paradigmas de especificación de software.

### 2.1.4.1. Elementos para la construcción de un archivo XML

Debido a que se utiliza el concepto relacional, los elementos tendrán nombres de la teoría relacional, a continuación se mencionan a detalle.

#### 2.1.4.1.1. Cabecera de XML

Como cualquier archivo XML necesita una cabecera, está se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8" ?>
```

#### 2.1.4.1.2. Relación

El primer nivel del XML es lo que corresponde a una relación. El nombre del elemento a utilizar es "relation", es lo que corresponde en un nivel físico a una base de datos.

Por simplicidad un archivo contendrá solamente un esquema de base de datos, los atributos que debe utilizar son:

- name: nombre de la relación.
- version: versión de esta definición, por defecto es "1.0".

A continuación se muestra un ejemplo del elemento que representa la relación.

```
<relation name="relation_name" version="relation_version">  
</relation>
```

#### 2.1.4.1.3. Entidades

Dentro de una relación se puede tener entidades y relaciones, para las entidades se utilizará el elemento "entities":

```
<entities>
</entities>
```

En entities se pueden definir cada entidad con el elemento "entity", los atributos por utilizar son:

- id: identificador de la entidad.
- name: nombre de la entidad.
- x1: posición horizontal inicial.
- y1: posición vertical inicial.
- x2: posición horizontal final.
- y2: posición vertical final.

```
<entity id="entity_id" name="entity_name" x1="100" y1="100" x2="200"
y2="200">
</entity>
```

#### 2.1.4.1.4. Atributos

De la misma manera en que se definen las entidades, asimismo se definen los atributos, el elemento contenedor es "attributes":

Dentro del elemento "attributes" se define cada atributo con el elemento "attribute", el cual, tiene los siguientes atributos:

- id: identificador del atributo.
- name: nombre del atributo.
- type: tipo de dato del atributo.
- key: define un atributo como llave, se representa con 0 si no es y con 1 si es llave.
- null: define un atributo nulo, se representa con 0 si no es nulo y con 1 si puede ser nulo.
- increment: si el atributo es de tipo entero puede definirse como auto incremental, se representa con 0 si no es y con 1 si lo es.

```
<attribute id="attribute_id" name="attribute_name" type="varchar" key="0"
null="0" increment="0"/>
```

#### 2.1.4.1.5. Relaciones

Dentro del elemento principal "relation" se definen las diferentes relaciones (con el sentido de "relationship") y se define con el elemento "relationships":

```
<relationships>
</relationships>
```

Y dentro de las relationships se definen las diferentes relaciones entre entidades con el elemento relationship y los siguientes atributos:

- id: identificador de la relación.
- id\_parent: identificador de la entidad que representa el padre o de quien se extrae información de referencia.
- id\_child: identificador de la entidad que representa el hijo o quien extraerá información de su padre.
- legend\_parent: leyenda que tiene la relación del lado del padre, puede ser opcional.
- legend\_child: leyenda que tiene la relación del lado del hijo, puede ser opcional.
- min\_parent: cardinalidad mínima del padre.
- max\_parent: cardinalidad máxima del padre.
- min\_child: cardinalidad mínima del hijo.
- max\_child: cardinalidad máxima del hijo.

```
<relationship id="relationship_id" id_parent="entity_parent_id"
id_child="entity_child_id" legend_parent="legend" legend_child="legend"
min_parent="0" max_parent="1" min_child="0" max_child="*" />
```



## 2.2. Métodos de especificación formal

Una especificación formal de software [18] es una descripción que se expresa mediante un lenguaje cuyo vocabulario, sintaxis y semántica están formalmente definidos. Los métodos formales se utilizan para referirse a cualquier actividad relacionada con representaciones matemáticas del software, incluyendo la especificación formal de sistemas, el análisis, y la demostración rigurosa de propiedades de la especificación.

Sommerville menciona que en las disciplinas de ingeniería es normal el análisis matemático dentro del proceso de desarrollo y validación del diseño de un producto; sin embargo, en la ingeniería de software no ha sucedido así, la utilización ha sido limitada debido a los costos que conlleva y la falta de claridad respecto de su rentabilidad.

En [18] se mencionan dos aproximaciones para redactar especificaciones formales para sistemas de software industrial:

- Aproximación algebraica, que se describe en función de las operaciones y relaciones, se mencionan Larch, Lotos, entre otros.
- Aproximación basada en modelos, que construye un modelo utilizando construcciones matemáticas como conjuntos y sucesiones, operaciones y estado del sistema, se mencionan Z, VDM, B, Redes de Petri, entre otros.

En los lenguajes formales, los conjuntos [19] son colecciones de objetos o elementos, son utilizados como parte fundamental de los métodos formales y sus componentes son únicos. Debido a lo anterior, los métodos formales permiten una especificación más completa, consistente y precisa, en comparación con los métodos diagramáticos.

Las deficiencias de los enfoques no formales [19] se pueden describir de la siguiente manera:

- Contradicciones, son conjuntos de planteamientos que divergen unos con otros.
- Ambigüedades, son planteamientos que se interpretan de diferentes formas.
- Vaguedades, ocurren por especificaciones muy voluminosas.

- Incompleto, es uno de los problemas que ocurren con mayor frecuencia con las especificaciones del sistema.
- Grados mixtos de abstracción, ocurren cuando planteamientos muy abstractos se mezclan al azar con planteamientos que tienen un grado mucho mayor de detalle.

Pero, ¿Por qué utilizar lenguajes formales?; según [20] las representaciones formales proveen un marco sistemático para construir y validar la sintaxis de sistemas a partir de la construcción de enfoques estándar de representación. Reducen el tiempo y esfuerzo requerido para comunicar y administrar los sistemas o procesos. Los lenguajes formales implican definiciones absolutamente precisas y exactas de los sistemas, que se puede utilizar para validar y puede ser utilizado como la base para la resolución de problemas con ayuda de computadoras

### **2.2.1. Lenguajes formales de especificación de software**

A continuación se resumen algunos enfoques de especificación formal de software [21]:

- Z: Es un lenguaje de especificación formal basado en la teoría de conjuntos y la lógica de predicados de primer orden. Está soportado por una biblioteca matemática, que contienen ciertas leyes algebraicas que ayudan al razonamiento acerca de las descripciones en Z. El principal elemento de este lenguaje es el esquema, que ayuda a la estructuración de la especificación matemática.
- SAZ: Es un método integrado de desarrollo de la Universidad de York, en el cual se genera una especificación basada en diagramas y texto, y desarrolla una descripción formal en Z basada en sus estados y operaciones. Originalmente fue desarrollado para usarlo con SSDAM, que es un método estructurado de análisis y diseño popular en el Reino Unido.
- B: Es un lenguaje que soporta un gran segmento de desarrollo del ciclo de vida, de la especificación a la implementación, tiene una base formal y tiene una semántica axiomática basada en el cálculo de precondiciones e inspirado en el trabajo de Dijkstra. Este lenguaje está muy relacionado con los lenguajes Z y VDM.

El elemento principal es una máquina abstracta que encapsula variables de estado que satisfacen cierto invariante. Una diferencia entre B y otros lenguajes son las pruebas obligatorias que se realizan automáticamente con el apoyo de una herramienta.

- ASM: Es un método basado en una máquina de estados abstractos que guía el desarrollo de software y sistemas embebidos desde la captura de requerimientos hasta la construcción. Soporta e integra uniformemente las actividades del ciclo de desarrollo para sistemas complejos.
- Alloy: Es un lenguaje [22] basado en Z. Describe las estructuras con una biblioteca matemática, pero es más simple que Z, está influido por las notaciones de modelado de objetos. Uno de los beneficios importantes de Alloy es que posee un analizador para chequear el modelo baso en tecnología de satisfacción booleana.
- OCL: Es un lenguaje de restricciones de UML. Busca precisar la semántica más que otros enfoques, pero para lograrlo se introdujo mucha complejidad. Esta complejidad ya no es posible extraerla del lenguaje debido a que se volvió estándar. Ha servido más como un aspecto textual de notación de UML.

### **2.2.2. Z: Lenguaje formal de especificación**

Usualmente cualquier lenguaje formal de especificación [19] debe contener tres componentes principales:

- Sintaxis
- Semántica
- Conjunto de relaciones

Los métodos formales [23] están basados en matemática discreta elemental, utilizados para producir documentación precisa y no ambigua, en la cual la información está estructurada y presentada en un adecuado nivel de abstracción, que ayuda mucho al proceso de diseño de sistemas de información y es una guía para su desarrollo, pruebas y mantenimiento.

Los métodos formales necesitan un sólido lenguaje de especificación formal, tal como el lenguaje Z, siendo este uno de los lenguajes más extensos y que tiene muchos aspectos positivos, proveyendo una especificación clara.

Z es un lenguaje de especificación que ha evolucionado [19]. Entre sus componentes se encuentran los conjuntos tipificados, las relaciones y las funciones dentro del contexto de predicados lógicos de primer orden, que se incorporan en esquemas como medio de estructuración.

Una especificación en Z está conformada por un conjunto de esquemas, siendo estos [19] en la especificación formal a lo que correspondería un componente en un lenguaje de programación. Los esquemas describen datos y restricciones sobre ellos.

La teoría de conjuntos utilizada en Z incluye operadores estándares de conjuntos, productos cartesianos y conjunto potencia. La lógica matemática es el cálculo de predicados de primer orden, y conjuntamente hacen del lenguaje más fácil de comprender y aprender, además de la abstracción y la ayuda en lo que se refiere a las pruebas.

En [1] y [23] se puede ampliar el conocimiento y conocer la mayoría de elementos del lenguaje Z.

### **2.3. Herramientas para integrar paradigmas**

Aquí se presentan los principales antecedentes de este trabajo, resumiendo las ventajas y desventajas que cada enfoque tiene; de esta manera, se construyó un marco de referencia robusto entre el modelo entidad-relación y Z.

### **2.3.1. Marco de referencia entre UML y Z**

Varios han sido los intentos por tratar de integrar los enfoques de especificación diagramáticos con los formales. Cabe destacar un marco de correspondencias de relaciones entre Z y UML presentadas en [24].

Dicho marco de correspondencias es muy importante; en él se muestra que una asociación de muchos a muchos corresponde a una relación, una asociación de uno a muchos es una función parcial, una asociación de uno a uno es una inyección, etc.

De la misma manera realiza correspondencias entre elementos de modelado, por ejemplo, un objeto o un caso de uso de UML es un esquema en lenguaje Z, un atributo en UML es una variable en lenguaje Z y las asociaciones se modelan con relaciones o funciones.

### **2.3.2. Integración de modelos estructurados y formales**

De manera similar al marco de referencia entre UML y Z fue tratado también en [25] por Jessica Cordero varios años antes, desde un aspecto de modelo de datos con el modelo entidad-relación.

Este marco está orientado a usuarios con experiencia que desean encapsular esquemas y reutilizarlos como una biblioteca. Como el lenguaje Z tiende a parecer un lenguaje difícil de aprender, las bibliotecas no ayudan a esa comprensión; por tanto, se requiere un marco de referencia, que aunque genere especificaciones más grandes en tamaño, sean más claras y fáciles de interpretar para usuarios no experimentados con el lenguaje Z. Además, el usar modelos diagramáticos simplifica la producción sistemática de modelos formales cuya estructura es estándar.

### 2.3.3. Combinación y refinamiento entre UML y Z

Un aspecto muy importante es tratado en [26], donde se realizan refinamientos con el lenguaje Z, haciéndolo más comprensivo y completo.

Este trabajo incluye una verificación preliminar comprensiva; la integración de ambos enfoques proporciona muchas ventajas, a la vez que neutraliza las desventajas.

### 2.3.4. Integración de UML y Z por medio de RoZ

Una de las herramientas de software desarrolladas para la integración de UML y el lenguaje Z es RoZ, desarrollada por Y. Ledru y S. Dupuy. RoZ es un ambiente que integra UML y Z, siendo una extensión de Rational Rose que hace la transición entre UML y Z [27].

RoZ tiene tres funcionalidades: la generación de especificación en Z, la generación de operaciones básicas y la generación de teoremas para validar operaciones. Aunque se debe notar la dependencia que tiene con la herramienta Rational Rose para la creación de diagramas de UML.

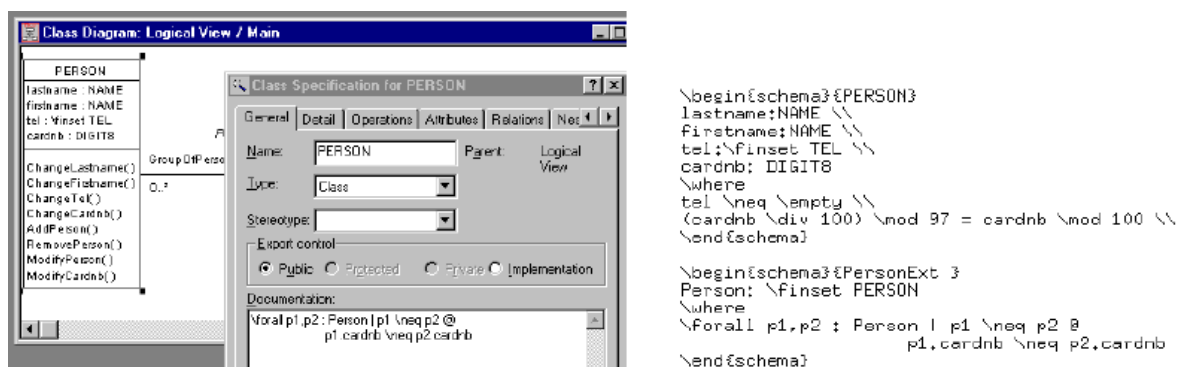


FIGURA No. 2.8 Generación de especificación en Z en formato Latex con RoZ [27]

### **2.3.5. Biblioteca ER-Z**

También existe una biblioteca de entidad-relación desarrollada para el lenguaje Z, es un conjunto de esquemas que permiten establecer una correspondencia uno-a-uno entre un diagrama de entidad-relación y un esquema en lenguaje Z. Esta fue desarrollada en el Instituto Tecnológico de Costa Rica por estudiantes del profesor Ignacio Trejos, y se tomará como base para la construcción de esquemas en Z con ciertas variaciones de sintaxis, sin usar la biblioteca ER-Z directamente sino su idea conceptual.

### 3. INTEGRACIÓN DE PARADIGMAS DE ESPECIFICACIÓN DE SOFTWARE

Este capítulo representa el aporte científico de la presente tesis. La integración entre paradigmas de especificación de software se puede realizar por medio de un conjunto de equivalencias, esto es, mediante un marco de referencia que establece correspondencias entre el modelo entidad-relación y el lenguaje Z, siendo estos los representantes de cada paradigma de especificación de software que se desea integrar en este trabajo.

El marco de referencia consta de varias partes, desde la definición de datos hasta las operaciones totales del sistema, tal como se mencionó en el alcance de este trabajo. En lo referente a las asociaciones el marco se basa en las asociaciones válidas según el apéndice B de Barker en [9] y la notación es la descrita en [4].

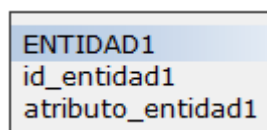
Es importante mencionar que a diferencia de lo propuesto por Jessica Cordero en [25], este marco de referencia pretende ayudar al usuario a entender mejor la especificación formal en Z; posteriormente esta especificación –como valor agregado a este trabajo– se generará automáticamente por una herramienta que parte de un diagrama en que se modelan los datos mediante el enfoque entidad-relación y la notación ESB. Para usuarios principiantes el lenguaje Z es difícil de comprender; la descripción de instancias y asociaciones por medio de una biblioteca se torna confusa para este tipo de usuarios aunque esta biblioteca conlleva otros beneficios.

#### 3.1. Definición de datos

Dentro de la definición de datos se deben tomar en cuenta las propiedades que caracterizan a las entidades que participan en el sistema.



En el modelo entidad-relación las propiedades se definen como atributos de la entidad dentro de un rectángulo –tal como se muestra en la siguiente figura– y en el lenguaje Z se definen como tipos básicos dentro de los símbolos “[..]” –tal como se muestra debajo de la siguiente figura–; para simplificar la especificación de la entidad solamente se muestra un identificador y un atributo.



**FIGURA No. 3.1 Ejemplo de entidad 1**

$[ID\_ENTIDAD1, ATRIBUTO\_ENTIDAD1]$

Estos tipos básicos permiten definir dominios abstractos para los atributos que aparecen dentro de las entidades.

A continuación se debe encapsular por un lado los identificadores (si aplica el caso, de lo contrario el identificador queda definido como un tipo) y por otro lado las propiedades de la entidad.

```

    INFO_ENTIDAD1
    atributo_entidad1: ATRIBUTO_ENTIDAD1
  
```

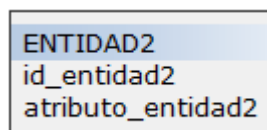
Hasta el momento ya se tienen dos conjuntos de datos: los identificadores y la información asociada; falta relacionarlos de cierta manera para cumplir con el hecho de tener tuplas únicas, esto se puede hacer por medio de una función parcial, donde el dominio de la función es el identificador y el rango es la información.

A continuación se muestra el esquema que relaciona los pares ordenados (identificador, información) por medio de la función parcial “info\_entidad1”; la única restricción es que el conjunto de identificadores declarados es igual al dominio de dicha función, a efectos de asegurar la integridad de los datos como un invariante.

<i>INSTA_ENTIDAD1</i>
<i>id_entidad1</i> : $\mathbb{P}$ <i>ID_ENTIDAD1</i>
<i>info_entidad1</i> : <i>ID_ENTIDAD1</i> $\rightarrow$ <i>INFO_ENTIDAD1</i>
<i>id_entidad1</i> = dom <i>info_entidad1</i>

De esta manera, se especifican 4 partes: el conjunto de datos, los identificadores, la información y la instancia. Estos constituyen la población de la base de datos. Las primeras tres partes definen la intensión o estructura informacional intrínseca a cada instancia de una entidad. La cuarta parte corresponde a la extensión o conjunto de instancia que pueblan la entidad en un instante en su vida, producto de una dinámica de operaciones de creación, modificación o eliminación de instancias.

Esto se debe hacer con toda entidad que participe en el sistema. A manera de ejemplo, cuando se requiere describir asociaciones binarias, se debe especificar otra entidad para los siguientes apartados de este marco de referencia, tal como se muestra en la siguiente figura:



**FIGURA No. 3.2 Ejemplo de entidad 2**

Y con base en la entidad 2 se produce de manera análoga a la entidad 1 el siguiente segmento de la especificación:

[*ID\_ENTIDAD2*, *ATRIBUTO\_ENTIDAD2*]

$INFO\_ENTIDAD1$ $atributo\_entidad2: ATRIBUTO\_ENTIDAD2$
--------------------------------------------------------------

$INSTA\_ENTIDAD1$ $id\_entidad2: \mathbb{P} ID\_ENTIDAD2$ $info\_entidad2: ID\_ENTIDAD2 \leftrightarrow INFO\_ENTIDAD2$
$id\_entidad2 = \text{dom } info\_entidad2$

### 3.2. Definición de asociaciones

Existen dos maneras de definir una asociación: por una parte, la asociación por participación, utilizada principalmente en el modelo entidad-relación de Peter Chen [8]; y por otra parte, la asociación con referencia al opuesto, utilizada por muchas notaciones entidad-relación, donde destacan la notación de Barker [9] e Information Engineering [12].

Este marco de referencia se basa en la asociación con referencia al opuesto, debido a que comúnmente se utiliza este tipo de asociación en diferentes herramientas de software para elaborar diagramas entidad-relación.

Cabe mencionar que desde el punto de vista lógico del modelo se utiliza la denominación “padre” e “hijo” para entidades participantes en cuanto a la asociación se refiere, debido por un parte a la facilidad de identificación de entidades para ayudar a hacer el mapeo de conceptual a lógico, y por otra parte a la dependencia de existencia que hay entre entidades.

Inicialmente se trató de simplificar el número de casos posibles para un mejor entendimiento, pero como se muestra en [4] los 16 casos posibles de asociaciones se reducen a 10 casos principales, que muestran el marco de referencia desde un punto de vista lógico-conceptual porque se permitió utilizar asociaciones de muchos a muchos para no limitar conceptualmente el marco de referencia.

En los esquemas del lenguaje Z se expresan distintos invariantes; por ejemplo, la cardinalidad expresa la restricción de la relación binaria ya sea con una función parcial o una función parcial inyectiva –según sea el caso–; y para la opcionalidad pueden haber de dos clases: cuando corresponde al concepto de opcionalidad representada como un subconjunto impropio ( $\subseteq$ ) y cuando corresponde al concepto de obligatoriedad representada con el símbolo ( $=$ ).

Las distintas cardinalidades y las opcionalidades brindan distintos casos de asociaciones; cada caso consta de cardinalidades mínimas (primer elemento) que corresponden a la opcionalidad y máximas (segundo elemento) que corresponden a la cardinalidad, tanto para el padre como al hijo, dichos casos se muestran a continuación:

### 3.2.1. Caso 1: Uno a muchos, ambos opcionales

Este caso es representado por la cardinalidad 01..0N; esta asociación es utilizada ocasionalmente; las cardinalidades máximas (1 y N) representan que un padre tiene asociados a muchos hijos, y las cardinalidades mínimas (0 y 0) representan que para ambas entidades pueden existir elementos que no estén asociados.

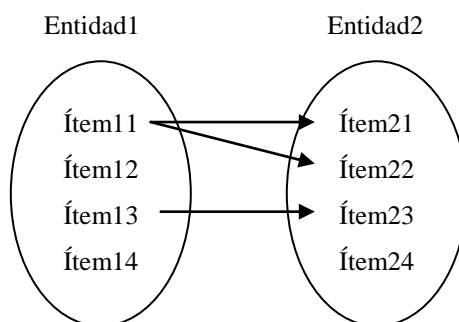
Para ilustrar el caso se crea una asociación entre la entidad 1 y la entidad 2 con las cardinalidades mencionadas anteriormente en el siguiente diagrama entidad-relación con la notación antes mencionada:



FIGURA No. 3.3 Caso 1: Asociación 01..0N

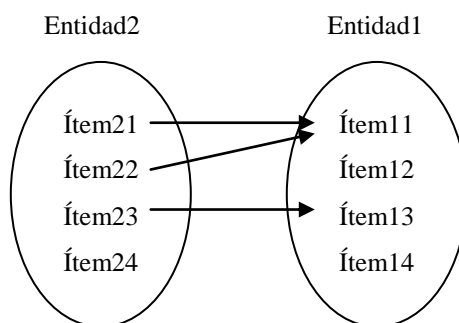
Todas las asociaciones de este marco de referencia se describen por medio de un esquema que tiene como declaración una relación binaria ( $\leftrightarrow$ ); esto evitará cualquier tipo de inconsistencia en cuanto a la restricción que por un lado proporciona la cardinalidad y por otro lado la opcionalidad.

La cardinalidad de esta asociación se puede describir en lenguaje Z por medio de un invariante que restringe la relación binaria utilizando una función parcial; para ilustrar de mejor manera dicha función se muestra el siguiente diagrama de correspondencia:



**FIGURA No. 3.4 Diagrama de correspondencia del caso 1**

La definición matemática de función no permite que un elemento del dominio tenga más de una imagen en el rango, debido a esto es apropiado cambiar el sentido de la función por medio de la correspondencia inversa, tal como se muestra en la siguiente figura:



**FIGURA No. 3.5 Diagrama de correspondencia inversa del caso 1**

De este modo, es posible representar la asociación por medio de una función parcial, que relaciona distintos identificadores, esto se realiza por medio de un esquema que contendrá los conjuntos de identificadores de ambas entidades, la relación binaria que involucra los identificadores de cada entidad y la restricción de la cardinalidad y opcionalidad.

La restricción de la opcionalidad mostrada en el invariante del esquema describe la contención de los elementos ya sea en el dominio o en el rango. En el caso del dominio de la relación se observa que es el conjunto de identificadores de la entidad 1, sin embargo, existen ciertos elementos que no están asociados, es decir, los identificadores pueden ser un subconjunto impropio del dominio de la función, es por ello que se utiliza el símbolo “ $\subseteq$ ”.

De la misma manera, en el rango también puede haber elementos sin asociación y se convierte en un subconjunto impropio. En el caso más general ambos podrían ser subconjuntos impropios, debido a que el dominio puede ser igual al conjunto de identificadores de la entidad 1 y el rango puede ser igual al conjunto de identificadores de la entidad 2.

*ASOCIA1\_ENTIDAD1\_ENTIDAD2*

*id\_entidad1:  $\mathbb{P}$  ID\_ENTIDAD1*

*id\_entidad2:  $\mathbb{P}$  ID\_ENTIDAD2*

*asocia1\_entidad1\_entidad2 : ID\_ENTIDAD1  $\leftrightarrow$  ID\_ENTIDAD2*

*dom *asocia1\_entidad1\_entidad2*  $\subseteq$  *id\_entidad1**

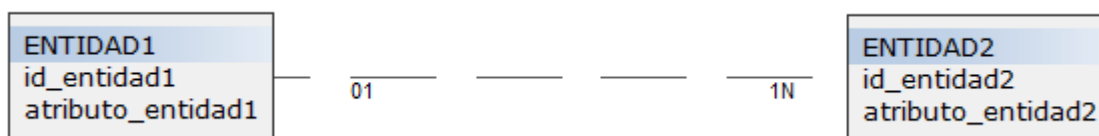
*ran *asocia1\_entidad1\_entidad2*  $\subseteq$  *id\_entidad2**

*asocia1\_entidad1\_entidad2  $\sim \in$  ID\_ENTIDAD2  $\rightarrow$  ID\_ENTIDAD1*

### 3.2.2. Caso 2: Uno a muchos, padre opcional

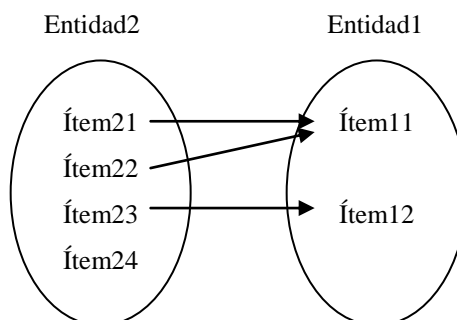
Este caso es representado por la cardinalidad 01..1N; esta asociación es muy rara, debido a que introduce un problema porque pueden existir elementos en la entidad hijo sin padre.

En la teoría de bases de datos se incluyó el término “nulo” en las llaves foráneas, pero conceptualmente hablando no tiene cabida esta situación, simplemente la asociación no existe; las cardinalidades máximas (1 y N) representan que un padre tiene asociados a muchos hijos, y las cardinalidades mínimas (0 y 1) representan que pueden existir ciertos elementos de la entidad 2 que no estén asociados con elementos de la entidad 1, es decir, elementos hijos que no tienen padre.



**FIGURA No. 3.6 Caso 2: Asociación 01..1N**

En el caso 1 se explicó la problemática de la correspondencia de la función, en este caso ya se utiliza la correspondencia inversa definida a partir del siguiente diagrama:



**FIGURA No. 3.7 Diagrama de correspondencia inversa del caso 2**

En este caso se tiene un elemento de la entidad 2 sin imagen, debido a que tiene padre opcional, la restricción de la opcionalidad mostrada en el invariante es que el dominio es igual al conjunto de identificadores de la entidad 1 (utilizando el símbolo “=”), este es igual debido a no hay elementos en la entidad 1 que no estén asociados y el rango es subconjunto impropio del conjunto de identificadores de la entidad 2 (similar al caso 1).

ASOCIA2\_ENTIDAD1\_ENTIDAD2

$id\_entidad1: \mathbb{P} ID\_ENTIDAD1$

$id\_entidad2: \mathbb{P} ID\_ENTIDAD2$

$asocia2\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$dom\ asocia2\_entidad1\_entidad2 = id\_entidad1$

$ran\ asocia2\_entidad1\_entidad2 \subseteq id\_entidad2$

$asocia2\_entidad1\_entidad2 \sim \in ID\_ENTIDAD2 \rightarrow ID\_ENTIDAD1$

El último invariante indica la cardinalidad de la relación con base en la correspondencia inversa mostrada anteriormente.

### 3.2.3. Caso 3: Uno a muchos, hijo opcional

Este caso es representado por la cardinalidad 11..0N; esta asociación es la más común forma de asociación ya que la dependencia de existencia corresponde correctamente a los términos de padre e hijo, es decir, primero se crea al padre y luego se crea al hijo; las cardinalidades máximas (1 y N) representan que un padre tiene asociados a muchos hijos, y las cardinalidades mínimas (1 y 0) representan que pueden existir elementos de la entidad 1 que no estén asociados con elementos de la entidad 2, es decir, elementos padre que no tienen hijos.

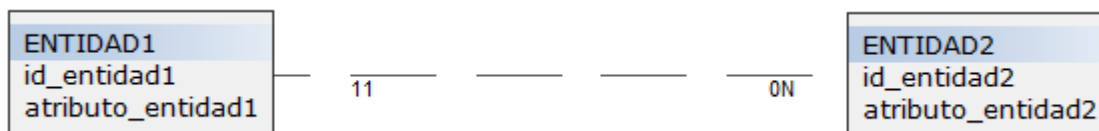
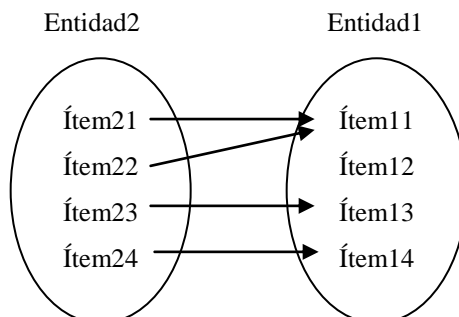


FIGURA No. 3.8 Caso 3: Asociación 11..0N

El diagrama de correspondencia inversa de este caso es el siguiente:





**FIGURA No. 3.9 Diagrama de correspondencia inversa del caso 3**

En este caso se tiene un elemento de la entidad 1 que no es imagen de nadie, debido a que es opcional tener hijos, la restricción de la cardinalidad mostrada en el invariante es que el dominio es subconjunto impropio ( $\subsetneq$ ) del conjunto de identificadores de la entidad 1 y el rango es igual ( $=$ ) al conjunto de identificadores de la entidad 2.

ASOCIA3\_ENTIDAD1\_ENTIDAD2

$id\_entidad1: \mathbb{P} ID\_ENTIDAD1$

$id\_entidad2: \mathbb{P} ID\_ENTIDAD2$

$asocia3\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$dom\ associa3\_entidad1\_entidad2 \subsetneq id\_entidad1$

$ran\ associa3\_entidad1\_entidad2 = id\_entidad2$

$asocia3\_entidad1\_entidad2 \sim \in ID\_ENTIDAD2 \rightarrow ID\_ENTIDAD1$

### 3.2.4. Caso 4: Uno a muchos, ambos obligatorios

Este caso es representado por la cardinalidad 11..1N; esta asociación es una construcción poderosa ya que necesita operaciones simultáneas para crear las instancias, desde el punto de vista relacional las tuplas de cada entidad deben crearse de manera atómica; las cardinalidades máximas (1 y N) representan que un padre tiene asociados a muchos hijos, y las cardinalidades mínimas (1 y 1) representan que cada elemento de cualquier entidad debe estar asociado.

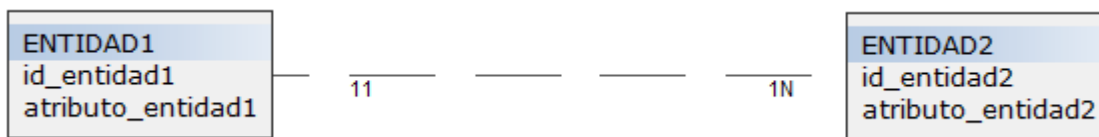


FIGURA No. 3.10 Caso 4: Asociación 11..1N

A continuación se muestra el diagrama de correspondencia inversa y su equivalencia en lenguaje Z, en este caso cada elemento de cualquier entidad debe estar asociado, la restricción de la opcionalidad mostrada en el invariante del esquema es que el dominio es igual ( $=$ ) al conjunto de identificadores de la entidad 1 y el rango es igual ( $=$ ) al conjunto de identificadores de la entidad 2 y.

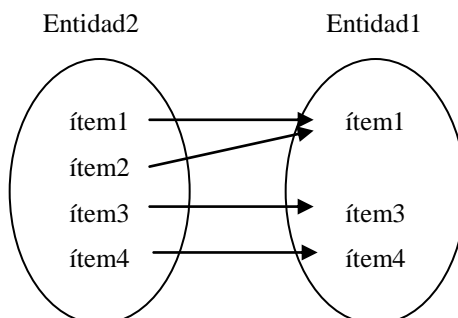


FIGURA No. 3.11 Diagrama de correspondencia inversa del caso 4

ASOCIA4\_ENTIDAD1\_ENTIDAD2

$id\_entidad1: \mathbb{P} ID\_ENTIDAD1$

$id\_entidad2: \mathbb{P} ID\_ENTIDAD2$

$asocia4\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$dom\ asocia4\_entidad1\_entidad2 = id\_entidad1$

$ran\ asocia4\_entidad1\_entidad2 = id\_entidad2$

$asocia4\_entidad1\_entidad2 \sim \in ID\_ENTIDAD2 \rightarrow ID\_ENTIDAD1$

Hasta el momento se han descrito los cuatro casos que se refieren a las asociaciones con cardinalidades de uno a muchos con sus diferentes opcionalidades. A continuación se mostrarán otras variaciones.

### 3.2.5. Caso 5: Uno a uno, ambos opcionales

Este caso es representado por la cardinalidad 01..01; las asociaciones de uno a uno son por naturaleza raras por el hecho de su obligatoriedad implícita; las cardinalidades máximas (1 y 1) representan que un padre tiene asociado a un hijo, y las cardinalidades mínimas (0 y 0) representan que pueden haber ciertos elementos de ambas entidades sin estar asociados, es decir, hijos sin padre o padres sin hijos.

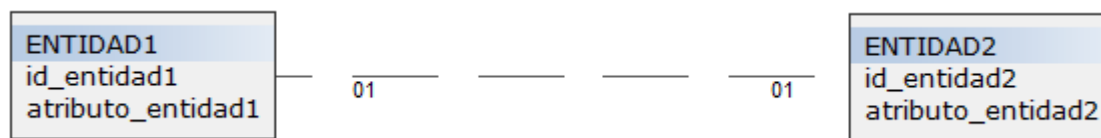


FIGURA No. 3.12 Caso 5: Asociación 01..01

Hay que notar que el diagrama de correspondencia en las asociaciones de uno a uno, cada elemento tiene solamente una imagen que varía un poco del concepto de función parcial. Debido a esto, no se necesita invertir el diagrama de correspondencia y por tanto el invariante del esquema que se refiere al tipo de función no tendrá el operador de inversa:

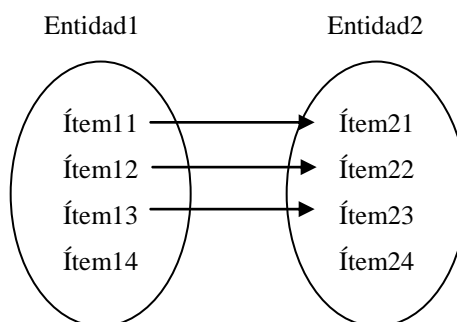


FIGURA No. 3.13 Diagrama de correspondencia del caso 5

En este caso pueden haber elementos de la entidad 2 que no son imagen de nadie y elementos de la entidad 1 que no tienen imagen, la restricción de la opcionalidad mostrada en el invariante del esquema es que el dominio es subconjunto impropio ( $\subsetneq$ ) del conjunto de identificadores de la entidad 2 y el rango es subconjunto impropio ( $\subsetneq$ ) del conjunto de identificadores de la entidad 1.

Además, es importante notar que la asociación es representada como un invariante por medio de una función parcial inyectiva, definiendo de esta manera, elementos del dominio con una sola imagen.

*ASOCIA5\_ENTIDAD1\_ENTIDAD2*

*id\_entidad1*:  $\mathbb{P} ID\_ENTIDAD1$

*id\_entidad2*:  $\mathbb{P} ID\_ENTIDAD2$

*asocia5\_entidad1\_entidad2* :  $ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$\text{dom } \textit{asocia5\_entidad1\_entidad2} \subseteq \textit{id\_entidad1}$

$\text{ran } \textit{asocia5\_entidad1\_entidad2} \subseteq \textit{id\_entidad2}$

$\textit{asocia5\_entidad1\_entidad2} \in ID\_ENTIDAD1 \succ\rightarrow ID\_ENTIDAD2$

### 3.2.6. Caso 6: Uno a uno, hijo o padre opcional

Este caso es representado por las cardinalidades 01..11 o 11..01 que en la práctica es el mismo caso; igual que el caso 5 este caso es una construcción rara; las cardinalidades máximas (1 y 1) representan que un padre tiene asociado a un hijo, y las cardinalidades mínimas (0 y 1) o (1 y 0) representan que pueden haber ciertos elementos de una de las entidades sin estar asociados, es decir, ya sea hijos sin padre para un caso o padres sin hijos para el otro caso.

A fin de ilustrar la correspondencia se utilizará solamente la cardinalidad (11..01) o dicho de otro modo hijo opcional.

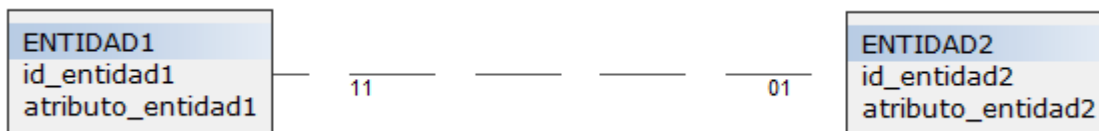


FIGURA No. 3.14 Caso 6: Asociación 11..01

El diagrama de correspondencia de este caso es el siguiente:

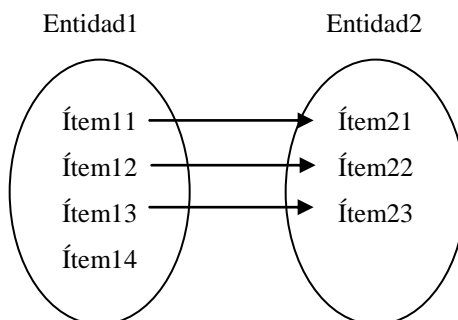


FIGURA No. 3.15 Diagrama de correspondencia del caso 6

En este caso puede existir elementos de la entidad 1 que no tienen imagen, debido a que tiene hijos opcionales, la restricción de la cardinalidad es que el dominio es subconjunto impropio ( $\subseteq$ ) del conjunto de identificadores de la entidad 1 y el rango es igual ( $=$ ) al conjunto de identificadores de la entidad 2, y definida con una función parcial inyectiva. El otro caso implícito simplemente sería de intercambiar la posición de las entidades, pero para fines prácticos es la misma situación.

ASOCIA6\_ENTIDAD1\_ENTIDAD2

$id\_entidad1: \mathbb{P} ID\_ENTIDAD1$

$id\_entidad2: \mathbb{P} ID\_ENTIDAD2$

$asocia6\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$dom\ asocia6\_entidad1\_entidad2 \subseteq id\_entidad1$

$ran\ asocia6\_entidad1\_entidad2 = id\_entidad2$

$asocia6\_entidad1\_entidad2 \sim \in ID\_ENTIDAD1 \rightsquigarrow ID\_ENTIDAD2$

### 3.2.7. Caso 7: Uno a uno, ambos obligatorios

Este caso es representado por la cardinalidad 11..11; esta asociación es muy rara debido a que sus instancias también deben crearse de manera simultánea, es utilizada solamente para sobrecarga o subtipo ortogonal; las cardinalidades máximas (1 y 1) representan que un padre tiene asociado a un hijo, y las cardinalidades mínimas (1 y 1) representan que cada elemento de ambas entidades debe estar asociado.

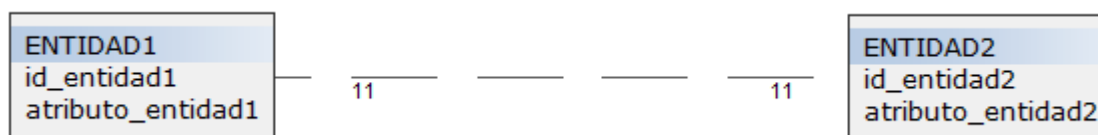


FIGURA No. 3.16 Caso 7: Asociación 11..11

El diagrama de correspondencia de este caso es el siguiente:

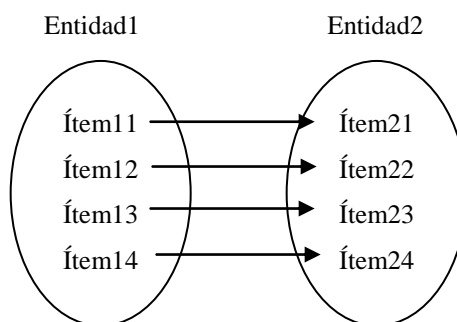


FIGURA No. 3.17 Diagrama de correspondencia del caso 7

En este caso todo elemento de la entidad 1 tiene imagen y todo elemento de la entidad 2 es imagen, la restricción de la cardinalidad es que el dominio es igual ( $=$ ) al conjunto de identificadores de la entidad 1 y el rango es igual ( $=$ ) al conjunto de identificadores de la entidad 2, y la cardinalidad se describe con una función parcial inyectiva en el invariante.

---

**ASOCIA7\_ENTIDAD1\_ENTIDAD2**


---

 $id\_entidad1: \mathbb{P} ID\_ENTIDAD1$ 
 $id\_entidad2: \mathbb{P} ID\_ENTIDAD2$ 
 $asocia7\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$ 


---

 $dom\ asocia7\_entidad1\_entidad2 = id\_entidad1$ 
 $ran\ asocia7\_entidad1\_entidad2 = id\_entidad2$ 
 $asocia7\_entidad1\_entidad2 \sim \in ID\_ENTIDAD2 \rightsquigarrow ID\_ENTIDAD1$ 


---

Con estos tres últimos casos termina la descripción de las asociaciones de uno a uno, a continuación se continúa con las asociaciones de muchos a muchos, las cuales solo figuran dos casos, porque el caso donde ambos son obligatorios es una asociación imposible según Barker en [9].

### 3.2.8. Caso 8: Muchos a muchos, ambos opcionales

Este caso es representado por la cardinalidad 0N..0N; esta asociación es muy común desde un punto de vista conceptual representa una asociación colectiva; las cardinalidades máximas (N y N) representan que varios padres tiene asociados a varios hijos, y las cardinalidades mínimas (0 y 0) representan que pueden existir elementos en ambas entidades que no están asociados, o que para un padre no es obligatorio tener hijos y viceversa.

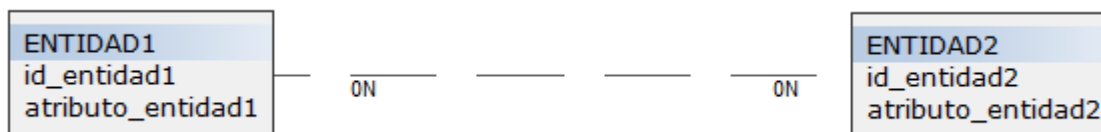
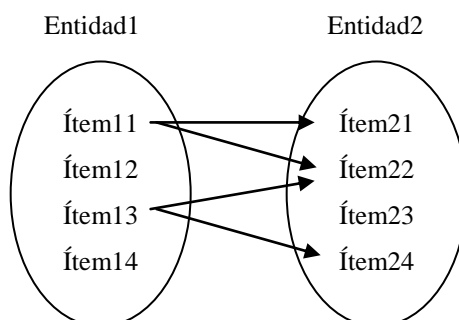


FIGURA No. 3.18 Caso 8: Asociación 0N..0N

De la misma manera que en las asociaciones de uno a uno no es necesario describir la relación con una función, por lo que no se necesita ni de la correspondencia inversa en el diagrama ni figurará el invariante de la función parcial; el diagrama de correspondencia para este caso es el siguiente:



**FIGURA No. 3.19 Diagrama de correspondencia del caso 8**

Como se mencionó, las asociaciones de muchos a muchos no pueden describirse con una función parcial; basta con la declaración de relación binaria donde se crea un conjunto de pares ordenados que corresponden a la imagen o las imágenes de cada elemento de la entidad 1.

Además por ser una asociación opcional para ambas entidades pueden existir elementos que no están asociados y esto corresponde a describir una restricción de opcionalidad como invariante del esquema con un subconjunto impropio ( $\subseteq$ ) tanto del dominio como en el rango respecto de los diferentes conjuntos de identificadores.

ASOCIA8\_ENTIDAD1\_ENTIDAD2

$id\_entidad1: \mathbb{P} ID\_ENTIDAD1$

$id\_entidad2: \mathbb{P} ID\_ENTIDAD2$

$asocia8\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$

$dom\ associa8\_entidad1\_entidad2 \subseteq id\_entidad1$

$ran\ associa8\_entidad1\_entidad2 \subseteq id\_entidad2$



### 3.2.9. Caso 9: Muchos a muchos, hijo o padre opcional

Este caso es representado por las cardinalidades 0N..1N o 1N..0N; esta asociación es rara y su comportamiento debe describirse a detalle para evitar problemas; las cardinalidades máximas (N y N) representan que varios padres tiene asociados a varios hijos, y las cardinalidades mínimas (0 y 1) o (1 y 0) representan que pueden existir elementos en una de la entidades participantes que no están asociados, o dicho de otra forma que hay hijos sin padres o padres sin hijos.

En el siguiente diagrama entidad-relación se utilizó la cardinalidad 0N..1N, es decir, donde el padre es opcional.

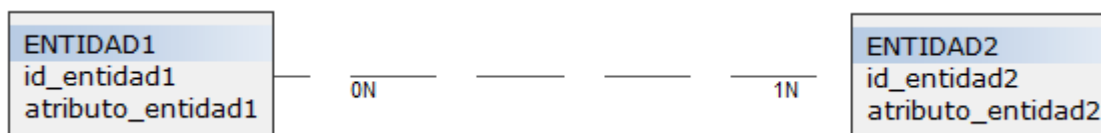


FIGURA No. 3.20 Asociación 0N..1N

Como el padre es opcional, el diagrama de correspondencia es similar al caso 8 con la diferencia que la entidad 2 tiene elementos sin padre. El diagrama de correspondencia para este caso es el siguiente:

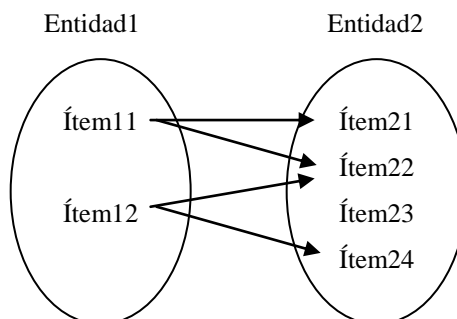


FIGURA No. 3.21 Diagrama de correspondencia del caso 9

Igual que el caso anterior se utiliza una relación binaria y en la restricción de la opcionalidad en el invariante del esquema por un lado el dominio de la relación es igual ( $=$ ) al conjunto de identificadores de la entidad1, es decir, no hay padres sin hijos; y por otro lado el rango de la relación es un subconjunto impropio ( $\subseteq$ ) del conjunto de identificadores de la entidad 2, es decir, hay hijos sin padre.

$ASOCIA9\_ENTIDAD1\_ENTIDAD2$ $id\_entidad1: \mathbb{P} ID\_ENTIDAD1$ $id\_entidad2: \mathbb{P} ID\_ENTIDAD2$ $asocia9\_entidad1\_entidad2 : ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD2$ <hr/> $dom\ asocia9\_entidad1\_entidad2 = id\_entidad1$ $ran\ asocia9\_entidad1\_entidad2 \subseteq id\_entidad2$
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 3.2.10. Caso 10: Asociaciones reflexivas

Según Barker en [9] la únicas dos asociaciones reflexivas que son validas son las de cardinalidad 01..0N y la de 01..01, siendo la última un construcción rara. La de uno a muchos describe un comportamiento jerárquico o de clasificación y la de uno a uno describe una alternativa.

Por ser casos similares solamente se diagramó la asociación de uno a muchos y se muestra en la siguiente figura.

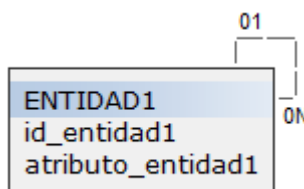


FIGURA No. 3.22 Asociación reflexiva 01..0N

Las descripciones de cada caso son similares, lo que cambia es el tipo de función a utilizar en el invariante. La asociación de uno a muchos utiliza una función parcial ( $\rightarrow$ ) y la asociación de uno a uno utiliza una función parcial inyectiva ( $\rightarrow$ ). Las restricciones de opcionalidades son idénticas para ambos casos, tanto el dominio como el rango de la asociación son un subconjunto impropio ( $\subseteq$ ) del conjunto de identificadores de la entidad en cuestión.

*ASOCIA10\_ENTIDAD1\_ENTIDAD2*

*id\_entidad1*:  $\mathbb{P} ID\_ENTIDAD1$

*asocia10\_entidad1\_entidad1* :  $ID\_ENTIDAD1 \leftrightarrow ID\_ENTIDAD1$

*dom* *asocia10\_entidad1\_entidad1*  $\subseteq id\_entidad1$

*ran* *asocia10\_entidad1\_entidad1*  $\subseteq id\_entidad1$

*asocia10\_entidad1\_entidad1*  $\sim \in ID\_ENTIDAD2 \rightarrow ID\_ENTIDAD1$

### 3.3. Descripciones de vaciedad y nulidad

En este apartado se muestran las decisiones que se tomaron respecto de situaciones particulares; por un lado, cuando en una entidad no hay información o atributos propios; y por otro lado, cuando en el esquema lógico se utilizan nulos en las llaves foráneas.

#### 3.3.1. Manejo de la vaciedad

Hay circunstancias que hacen que las entidades no tengan atributos propios; por ejemplo, cuando se desea modificar una asociación de muchos a muchos hacia una asociación de uno a muchos se debe crear una nueva entidad en medio de ambas que tiene por identificador el conjunto de identificadores de las entidades antes asociadas y por ende no tiene otra información.

Para estos casos se define un tipo libre en lenguaje Z que describe esa falta de información y este tipo se utiliza para crear un atributo imaginario dentro de la entidad. Esto se hace por uniformidad al estilo de la descripción de la información de entidades.

Tanto el tipo libre como el esquema se describen a continuación:

$$NULL\_ENTIDAD1 ::= no\_information$$

$INFO\_ENTIDAD1$ $att\_entidad1: NULL\_ENTIDAD1$
--------------------------------------------------

### 3.3.2. Manejo de la nulidad

En las bases de datos para manejar la opcionalidad en alguna asociación se propuso la utilización de los nulos en las llaves foráneas, pero es bien sabido que desde un punto de vista conceptual el nulo provoca problemas, y en el lenguaje Z no hay excepción, sin embargo, en [23] se propone manejar los nulos por medio de una definición de tipo libre de Z, aunque esta solución en el modelo entidad-relación significa que se crea un tipo libre para cada identificador de las asociaciones con esta condición.

Esta condición solamente se da cuando la opcionalidad de la asociación es por parte de la entidad padre, es decir, existen tuplas de la entidad hija que no están asociadas a ninguna tupla de la entidad padre, es decir, una llave foránea nula. La solución se muestra a continuación:

$$NULL\_ID\_ENTIDAD ::= null\_id\_entidad \mid id\_entidad \langle\langle ID\_ENTIDAD \rangle\rangle$$

De esta manera, se especifica que en cierto momento el tipo *NULL\_ID\_ENTIDAD* puede tomar el valor en efecto del identificador de la entidad en cuestión o por otro lado puede ser nulo. Esta descripción se utiliza posteriormente en las operaciones donde el identificador esté relacionado para simular la nulidad.

### 3.4. Definición del sistema de base de datos

Un sistema de base de datos es aquel compuesto por el conjunto de entidades, atributos y asociaciones; de la misma manera en  $Z$  se declara un estado del sistema con esos mismos datos de la siguiente manera:

<i>SYSTEM_STATE</i>
<i>INSTA_ENTIDAD1</i>
<i>INSTA_ENTIDAD2</i>
<i>ASOCIA_ENTIDAD2_ENTIDAD1</i>

También es necesario declarar el sistema cuando es inicializado y que no tiene datos, en este caso se especifica asociando el conjunto vacío con el conjunto de identificadores y asociaciones de la siguiente manera:

<i>INITIAL_SYSTEM_STATE</i>
<i>SYSTEM_STATE</i> '
<i>id_entidad1</i> ' = $\emptyset$
<i>id_entidad2</i> ' = $\emptyset$
<i>asocia_entidad2_entidad1</i> ' = $\emptyset$

Es fácil notar que el sistema de base de datos se definió como un conjunto, en el cual se realizarán ciertas operaciones básicas y para cada tipo de operación entrarán a participar ciertas entidades o asociaciones.

Por tanto es necesario definir un esquema de operación para cada entidad utilizando los esquemas Delta y Xi ( $\Delta$  y  $\Xi$ ) que significan que algo cambia o que nada cambia respectivamente, según las convenciones que se acostumbra al realizar especificaciones en Z [1] y [23].

Por ejemplo, las operaciones que afectan a la entidad 1 -la cual es padre- solo utilizan el conjunto de identificadores propios y no afectan a los demás; se dice que todo el sistema cambia excepto la entidad hijo y la asociación entre ellas; por lo que se utiliza el operador Delta ( $\Delta$ ) para el estado del sistema y el operador Xi ( $\Xi$ ) para lo demás:

<i>ENTIDAD1_OPERATION</i> $\Delta$ SYSTEM_STATE $\Xi$ INSTA_ENTIDAD2 $\Xi$ ASOCIA_ENTIDAD2_ENTIDAD1
--------------------------------------------------------------------------------------------------------------

Este esquema debe elaborarse por cada entidad del sistema de bases de datos para determinar qué conjunto de identificadores y asociaciones afecta; por otra parte también se utiliza para hacer más simplificados los esquemas de operaciones básicas, evitando así declaraciones repetitivas en los esquemas posteriores.

### 3.5. Definición de operaciones básicas

El alcance de este trabajo fue definir las operaciones básicas sobre la base de datos, entre las cuales están: agregar, eliminar, actualizar y consultar instancias de una entidad.

Para describir las operaciones se necesitan tres cosas:

- Operaciones parciales, que describen las operaciones exitosas.
- Mensajes de error y éxito, que describen mensajes al usuario.
- Operaciones completas, que describen las operaciones en sus diferentes alternativas de flujo.

### 3.5.1. Operación agregar

La operación agregar tiene dos objetivos: almacenar por un lado el identificador en el conjunto de identificadores y por otro lado el par (identificador, información). El esquema se inicia con la declaración del esquema de operación de la entidad, el identificador y la información de entrada, los últimos dos son parte del conjunto de identificadores y de información. En el predicado se verifica que el identificador de entrada no pertenezca al conjunto de identificadores de dicha entidad de lo contrario no puede agregarse porque ya existe; luego se procede a modificar el conjunto de identificadores uniendo el nuevo identificador con el operador de unión ( $\cup$ ); por último, también se procede a modificar la información uniendo el nuevo par ordenado (identificador, información) con el operador unión. A continuación se presenta un ejemplo del esquema explicado:

*ENTIDAD1\_ADD\_OK*

*ENTIDAD1\_OPERATION*

*id\_entidad1\_in? : ID\_ENTIDAD1*

*info\_entidad1\_in? : INFO\_ENTIDAD1*

*id\_entidad1\_in?  $\notin$  id\_entidad1*

*id\_entidad1' = id\_entidad1  $\cup$  {id\_entidad1\_in?}*

*info\_entidad1' = info\_entidad  $\cup$  {id\_entidad\_in?  $\mapsto$  info\_entidad1\_in?}*

### 3.5.2. Operación eliminar

Para realizar esta operación es necesario quitar el par ordenado (identificador, información) de la información y el identificador del conjunto de identificadores. Similar al esquema de agregar, en el de eliminar se declara el esquema de la operación y solamente el identificador como entrada, ya que no se necesita la información asociada.

En el predicado se verifica si el identificador pertenece al conjunto de identificadores -al contrario de la operación agregar que no debe pertenecer-, además el identificador no debe pertenecer al rango de ninguna asociación, si estas precondiciones se cumplen se elimina el par ordenado restringiendo el dominio ( $\Leftarrow$ ) y aplicando la operación de diferencia ( $\setminus$ ) al conjunto de identificadores, a continuación se muestra un ejemplo del esquema explicado:

<p><i>ENTIDAD1_DELETE_OK</i></p> <hr/> <p><i>ENTIDAD1_OPERATION</i></p> <p><i>id_entidad1_in? : ID_ENTIDAD1</i></p> <hr/> <p><i>id_entidad1_in? ∈ id_entidad1</i></p> <p><i>id_entidad1_in? ∉ dom asocia1_entidad1_entidad2</i></p> <p><i>info_entidad1' = {id_entidad1_in?} ⇐ info_entidad1</i></p> <p><i>id_entidad1' = id_entidad1 \ { id_entidad1_in?}</i></p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 3.5.3. Operación actualizar

En esta operación no cambian los conjuntos solamente la información del par ordenado (identificador, información). La operación actualizar es similar a la de agregar con respecto de la declaración del esquema, la diferencia es en el predicado; se verifica si el identificador pertenece al conjunto de identificadores y si es así se procede a actualizar por medio del operador de sobrecarga relacional ( $\oplus$ ), en otras palabras este operador se utiliza para sobrescribir el contenido del par ordenado (identificador, información), a continuación se muestra un ejemplo del esquema explicado:

<p><i>ENTIDAD1_UPDATE_OK</i></p> <hr/> <p><i>USUARIO_OPERATION</i></p> <p><i>id_entidad1_in? : ID_ENTIDAD1</i></p> <p><i>info_entidad1_in? : INFO_ENTIDAD1</i></p> <hr/> <p><i>id_entidad1_in? ∈ id_entidad1</i></p> <p><i>info_entidad1' = info_entidad1 ⊕ {id_entidad1_in? ↦ info_entidad_in?}</i></p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



### 3.5.4. Operación consultar

A diferencia con los esquemas anteriores, el de consultar utilizará el estado del sistema sin modificación, es decir, se utiliza el esquema Xi para incluirlo; también hay dos variables, una de entrada (?) y otra de salida (!), la de entrada es el identificador y la de salida es la información que corresponde a dicho identificador.

En el predicado se debe verificar que el identificador de entrada pertenece al conjunto de identificadores y la salida será la información por medio de la aplicación funcional de “info\_entidad1”, a continuación se muestra el esquema explicado:

*ENTIDAD1\_QUERY\_OK*

$\exists$ SYSTEM\_STATE

*id\_entidad1\_in?* : ID\_ENTIDAD1

*info\_entidad1\_out!* : INFO\_ENTIDAD1

$id\_entidad1\_in? \in id\_entidad1$

$info\_entidad1\_out! = info\_entidad1(id\_entidad1\_in?)$

Hasta el momento se han mostrado las cuatro operaciones parciales básicas, a continuación se mostrarán los mensajes de error y de éxito.

### 3.5.5. Mensajes de respuesta

Todas las operaciones mostradas anteriormente son las de resultado exitoso, sin embargo, siempre hay que tomar en cuenta errores que los usuarios pueden introducir al sistema y captarlos para mostrar ciertos mensajes de respuesta.

Para ello se debe definir un tipo libre con las respuestas, que a su vez se tendrá varios tipos: OK, EXISTS, NOT\_FOUND, HAS\_RELATIONSHIP.

*RESPONSE ::= OK / EXISTS / NOT\_FOUND / HAS\_RELATIONSHIPS*

La mejor alternativa es que la operación haya sido exitosa, por lo que define un esquema en común que usarán todas las operaciones exitosas y que muestra un mensaje de OK.

*SUCCESS\_OP*

*response! : RESPONSE*

*response! = OK*

Con respecto de las demás alternativas se elaboran esquemas de error que tienen componentes repetitivos tal como el estado del sistema sin cambios ( $\exists$ ), la entrada del identificador (?) y como salida la respuesta (!); por lo que se crea un esquema en común.

*ENTIDAD1\_ERROR*

*$\exists$ SYSTEM\_STATE*

*id\_entidad1\_in? ID\_ENTIDAD1*

*response! : RESPONSE*

Y por último las diferentes alternativas erróneas para operaciones básicas son:

- Que el identificador ya exista (EXISTS), para cuando se agrega.
- Que el identificador no exista (NOT\_FOUND), para cuando se elimine o se actualice.
- Que el identificador existe dentro de una asociación (HAS\_RELATIONSHIPS), cuando se agrega o se quiera eliminar.

Cuando el identificador ya existe se declara el esquema común de error y en el predicado se verifica que el identificador pertenece al conjunto de identificadores y se muestra un error.

*ENTIDAD1\_EXISTS*

*USUARIO\_ERROR*

$id\_entidad1\_in? \in id\_entidad1$

$response! = EXISTS$

Cuando el identificador no existe se declara el esquema común de error y en el predicado se verifica que el identificador no pertenece al conjunto de identificadores y se muestra un error.

*ENTIDAD1\_NOT\_FOUND*

*USUARIO\_ERROR*

$id\_entidad1\_in? \notin id\_entidad1$

$response! = NOT\_FOUND$

Cuando el identificador tiene asociaciones se declara el esquema común de error y en el predicado se verifica que el identificador pertenece al rango de una asociación y se muestra un error.

*ENTIDAD1\_HAS\_RELATIONSHIPS*

*USUARIO\_ERROR*

$id\_entidad1\_in? \in ran\ asocia\_entidad1\_entidad2$

$response! = HAS\_RELATIONSHIPS$

### 3.5.6. Operaciones completas

Al definir las operaciones exitosas y los posibles mensajes se puede construir con estas la descripción de las operaciones completas. Para construir estas operaciones de manera simplificada se definen los esquemas de forma horizontal por medio del constructor “ $\cong$ ” y se utilizan expresiones de esquema.

Lo que define el esquema completo es el flujo de la operación, es decir, los diferentes caminos alternos que la operación puede tomar; por ejemplo, la operación agregar tiene dos caminos:

- Ejecutar la operación de manera exitosa y
- Mostrar un mensaje de error acerca de que ya existe el identificador que se quiere crear con la operación.

La manera de describir estos caminos alternos es por medio del “o lógico” ( $\vee$ ) y entre la operación exitosa y su mensaje por medio de un “y lógico” ( $\wedge$ ), tal como se muestra a continuación:

$$ENTIDAD1\_ADD \cong (ENTIDAD1\_ADD\_OK \wedge SUCCESS\_OP) \\ \vee ENTIDAD1\_EXISTS$$

De la misma manera se construyen las operaciones completas para cada tipo: agregar, eliminar, actualizar y consultar. Es libre agregar diferentes caminos alternos para cada operación, solamente es de agregar un “o lógico” por cada camino alternativo. Sin embargo, hay que notar que los mensajes definidos son: identificador existe, identificador no existe e identificador está asociado.

A continuación se muestran las cuatro operaciones completas básicas con sus respectivos mensajes de error:

$$\begin{aligned} ENTIDAD1\_ADD &\equiv (ENTIDAD1\_ADD\_OK \wedge SUCCESS\_OP) \\ &\vee ENTIDAD1\_EXISTS \end{aligned}$$

$$\begin{aligned} ENTIDAD1\_DELETE &\equiv (ENTIDAD1\_DELETE\_OK \wedge SUCCESS\_OP) \\ &\vee ENTIDAD1\_NOT\_FOUND \\ &\vee ENTIDAD1\_HAS\_RELATIONSHIPS \end{aligned}$$

$$\begin{aligned} ENTIDAD1\_UPDATE &\equiv (ENTIDAD1\_UPDATE\_OK \wedge SUCCESS\_OP) \\ &\vee ENTIDAD1\_NOT\_FOUND \end{aligned}$$

$$\begin{aligned} ENTIDAD1\_QUERY &\equiv (ENTIDAD1\_QUERY\_OK \wedge SUCCESS\_OP) \\ &\vee ENTIDAD1\_NOT\_FOUND \end{aligned}$$

## **4. ANÁLISIS Y DISEÑO DE UNA HERRAMIENTA PARA INTEGRAR PARADIGMAS DE ESPECIFICACIÓN DE SOFTWARE**

Para comprobar que es posible la integración pragmática entre paradigmas de especificación de software –específicamente con base en el marco de referencia entre el modelo entidad–relación y el lenguaje Z- se propuso la construcción de una herramienta de software que utilice el marco de referencia propuesto en el capítulo anterior con el fin de realizar transformaciones automatizadas de un modelo ER a uno con semántica expresada en Z para comprobar la efectividad de la integración.

El presente capítulo contiene la especificación de requerimientos de software bajo el estándar IEEE 830[28] y según la metodología, el alcance y las limitaciones propuestos para dicha herramienta.

### **4.1. Introducción**

#### **4.1.1. Propósito**

Elaborar modelos de datos mediante diagramas entidad-relación y generar automáticamente una especificación formal equivalente basada en lenguaje Z.

Este producto de software está destinado tanto a audiencias académicas como a profesionales que desean realizar una integración de los modelos diagramáticos con los formales para garantizar la calidad del software. No es necesario que la audiencia tenga conocimiento avanzado del lenguaje Z, la idea es facilitar la generación automática de dicho lenguaje a partir de la construcción de un diagrama entidad-relación.

### **4.1.2. Alcance**

El nombre del producto es ERZ, siendo un diagramador de modelos entidad-relación que mapea diferentes niveles de esquemas relacionales y complementa el modelo de datos con la especificación formal en el lenguaje Z definido por Spivey [1], con el fin de integrar características propias de cada paradigma de especificación de software.

Con respecto de los alcances y las limitaciones, la herramienta de software se desempeña como un prototipo funcional que está disponible a mejoras posteriores.

Acerca del modelo entidad-relación se utiliza la notación ESB [4] que está basada en diferentes notaciones y no se pretende que maneje cuestiones como: dependencia de existencia, generación de scripts, relaciones no binarias, y el límite de entidades que soporta está limitado por el espacio visible de la interfaz.

Con lo que se refiere al lenguaje Z se genera dos tipos de formatos: PDF y TEX; ambos formatos tienen la misma especificación basada en: el diccionario de datos, definición de identificadores, información, instancias, asociaciones, estado del sistema y las operaciones básicas de agregar, eliminar, actualizar y consultar.

### **4.1.3. Definiciones, acrónimos y abreviaturas**

- ER: Modelo entidad-relación que sirve para representar conceptualmente el mundo real.
- ESB: Notación para diagramar modelos entidad-relación basada en cardinalidades mínimas y máximas.
- ERZ: Producto de software para la integración de modelos de datos con especificaciones formales y mapeo entre esquemas relacionales.
- Z: Lenguaje de especificación formal para describir y modelar sistemas.

#### **4.1.4. Referencias**

Existen dos publicaciones realizadas por el investigador que se refieren a formatos de interfaces externas y a la notación seleccionada para elaborar los diagramas entidad-relación.

- Definición de Esquemas Relacionales en XML para Propósitos de Integración. 2009. Luis Espino. Instituto Tecnológico de Costa Rica.
- Notación ESB: Modelado Semántico de Datos basado en Diagramas Entidad-Relación Normalizados. 2009. Luis Espino. Instituto Tecnológico de Costa Rica.

#### **4.1.5. Descripción del documento**

A continuación se detalla por una parte la descripción general del producto, que contiene las características, funciones y restricciones; por otra parte los requerimientos específicos, que contiene los requerimientos de interfaz externa, requerimientos funcionales y no funcionales.

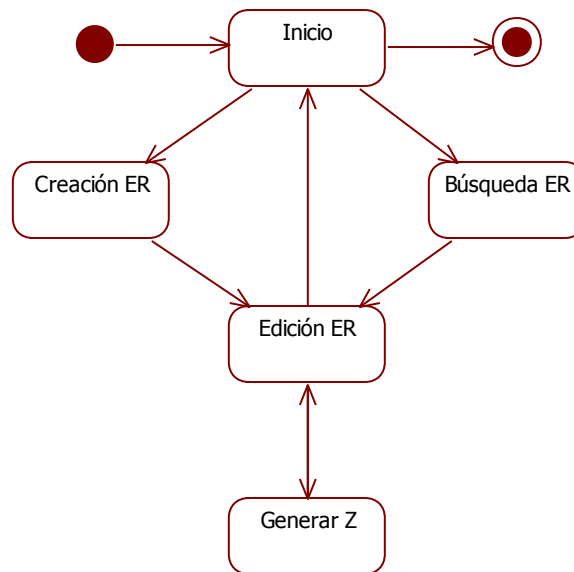
### **4.2. Descripción general**

#### **4.2.1. Perspectiva del producto**

ERZ es un producto de software autónomo, es decir, no es componente de ningún otro producto de software. Está basado en Web, los lenguajes de programación considerados inicialmente fueron PHP y JavaScript, con ayuda de AJAX. En la parte visual se estudió utilizar ciertas aplicaciones de Flash, que fueron descartadas.



El comportamiento, las operaciones y las diferentes interfaces entre los componentes del producto de software están definidos en el siguiente diagrama de estados:



**FIGURA No. 4.1 Diagrama de estados**

#### 4.2.2. Funciones del producto

Las principales funciones del producto son:

- Diagramar ER: Por medio gráfico la función principal del producto es permitir elaborar fácilmente diagramas entidad-relación basado en la notación ESB [4].
- Importar/Exportar XML: Para permitir un intercambio entre modelos.
- Generar Z: A partir de algoritmos de traducción se pueden generar dos archivos en formato de marcas y de impresión con la especificación equivalente en lenguaje Z.

### 4.2.3. Características de usuario

Para lograr el objetivo de la integración entre modelos diagramáticos y formales, se necesita que los usuarios que hagan uso del producto tengan conocimientos en la construcción de diagramas entidad-relación, luego la herramienta será la encargada de presentar una especificación equivalente en lenguaje Z. Idealmente, se espera que los usuarios tengan conocimientos suficientes de Z para completar la especificación formal.

Sin embargo, se considera que la herramienta puede servir como punto de aprendizaje del lenguaje Z; el modelo entidad-relación por tener características diagramáticas es muy fácil de construir y no se requiere ser un experto en bases de datos para interpretarlo.

Por otro lado, la herramienta tiene la capacidad de producir una especificación con base en marcas (TEX) con el fin de expandir la especificación, para esto el usuario debe tener conocimientos avanzados del lenguaje y del verificador de tipos FUZZ. Este verificador puede ser descargado del sitio [29] y existe un manual [30] que es de mucha ayuda.

### 4.2.4. Restricciones

Bajo el concepto de “Software as a Service”, el producto fue publicado en un sitio Web y tiene habilitado todo el funcionamiento para que los usuarios puedan ejecutar las funciones mencionadas anteriormente.

Debido a que el objetivo es generar un archivo con la especificación en Z, queda como interfaz manual para el usuario trasladar y compilar el archivo generado con cualquier herramienta de Latex para generar un posterior documento de especificación formal con una presentación adecuada de los segmentos que contienen notación matemática. Si el usuario no puede producir este documento se habilitó en la herramienta un generador en formato PDF para una vista inmediata de la especificación.

#### **4.2.5. Supuestos y dependencias**

La herramienta está desarrollada en PHP y JavaScript, por lo que es necesario que el usuario haga uso de la herramienta a partir de un navegador de internet, se sugiere Internet Explorer o Firefox, ya que la herramienta fue probada en esos navegadores. Con respecto de JavaScript, es necesario que el navegador tenga habilitada la ejecución de este lenguaje, de lo contrario la funcionalidad de la herramienta se verá afectada.

Se espera que el usuario tenga instaladas las herramientas necesarias para la ulterior manipulación del archivo generado por ERZ. Se sugiere tener instalado el verificador de tipos FUZZ y cualquier herramienta Latex.

La herramienta no depende de ningún otro software, pero si tiene vinculadas dos bibliotecas: una para dibujar asociaciones y otra para generar archivos PDF –más adelante se detallan las características-. Estas bibliotecas están incluidas con la herramienta, por lo que no se necesita ninguna instalación y se cuenta con el permiso para utilizarlas.

### **4.3. Requerimientos específicos**

#### **4.3.1. Requerimientos de interfaz externo**

El formato de los archivos de entrada se toma como un requerimiento externo, está definido en [17] para más detalle. Básicamente es la especificación del formato del archivo de intercambio a utilizar, el formato es XML.

## 4.3.2. Requerimientos funcionales

### 4.3.2.1. Crear ER

Actores: Usuario (iniciador)

Tipo: Primario

Descripción: Un usuario desea crear un diagrama entidad-relación, puede agregar, eliminar y modificar elementos, también puede guardar el diagrama en formato XML, al tener el diagrama en formato XML puede generar el lenguaje Z equivalente.

### 4.3.2.2. Abrir XML

Actores: Usuario (iniciador)

Tipo: Primario

Descripción: Un usuario desea abrir un archivo en formato XML, puede generar el lenguaje Z equivalente, también puede mostrar el diagrama entidad-relación asociado, puede agregar, eliminar y modificar elementos, también puede guardar el diagrama en formato XML.

### 4.3.2.3. Diagramar ER

Actores: Usuario y Módulo ER

Tipo: Primario

Descripción: Un usuario al crear un diagrama entidad-relación o al abrir un archivo XML puede mostrar el diagrama entidad-relación asociado, puede agregar, eliminar y modificar elementos, también puede guardar el diagrama en formato XML, al tener el diagrama en formato XML puede generar el lenguaje Z equivalente.

**4.3.2.4. Agregar elemento**

Actores: Usuario y Módulo ER

Tipo: Opcional

Descripción: Un usuario al diagramar el modelo entidad-relación puede agregar cualquier elemento que pertenece a la notación seleccionada.

**4.3.2.5. Eliminar elemento**

Actores: Usuario y Módulo ER

Tipo: Opcional

Descripción: Un usuario al diagramar el modelo entidad-relación puede eliminar cualquier elemento asociado al diagrama.

**4.3.2.6. Modificar elemento**

Actores: Usuario y Módulo ER

Tipo: Opcional

Descripción: Un usuario al diagramar el modelo entidad-relación puede modificar cualquier elemento asociado al diagrama.

**4.3.2.7. Guardar XML**

Actores: Usuario y Módulo ER

Tipo: Opcional

Descripción: Un usuario al tener un diagrama entidad-relación lo puede almacenar en un archivo con el formato XML seleccionado.

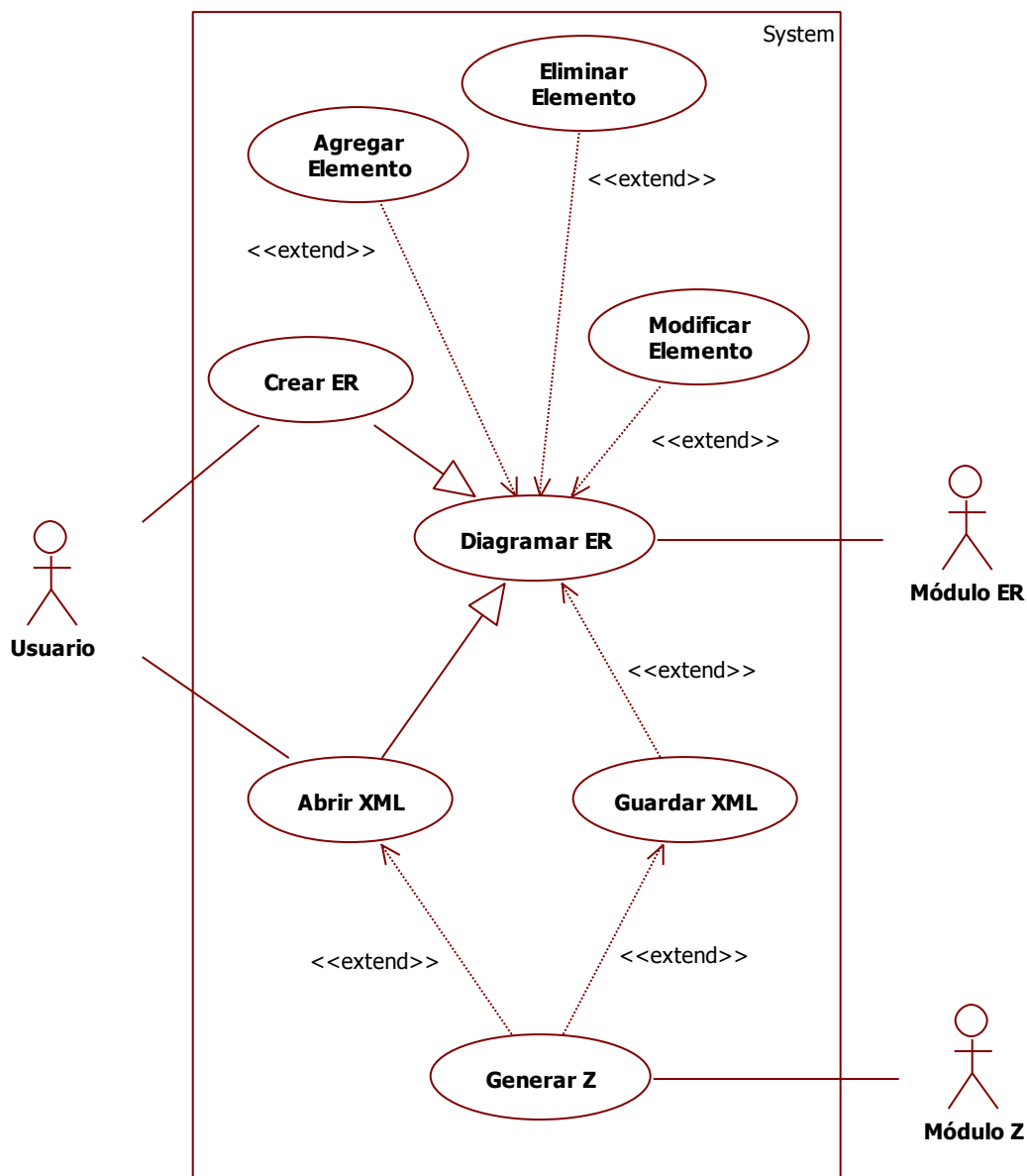


FIGURA No. 4.2 Casos de Uso

#### 4.3.2.8. Generar Z

Actores: Usuario y Módulo ER

Tipo: Opcional

Descripción: Un usuario al tener un diagrama entidad-relación puede generar la descripción correspondiente en Z.

### **4.3.3. Requerimientos no funcionales**

Se estudiaron varias clasificaciones de factores de calidad, las dos más cercanas al desarrollo son: por un lado la basada en el estándar ISO 9126-1 [31] y por otro lado el modelo de calidad de McCall [32].

Por el alcance de este trabajo no se considera necesario seguir el estándar ISO 9126, ya que dicho estándar además de presentar el modelo de calidad en la parte 1 tiene una amplia descripción de métricas en las siguientes partes; mientras que el modelo de McCall se acopla mejor con el desarrollo por la subjetividad que tiene asociada.

McCall se concentra en tres aspectos del producto de software: las características operativas, la capacidad de experimentar cambios y la capacidad para adaptarse a nuevos entornos.

Los requerimientos asociados con sus características operativas (operación del producto) son: corrección, confiabilidad, eficiencia, integridad y facilidad de uso; los cuales se muestran a continuación:

#### **4.3.3.1. Corrección**

Es el grado en que el programa cumple con su especificación y satisface los objetivos propuestos bajo las siguientes dimensiones:

- Misión de salida: La herramienta de software deberá generar una especificación equivalente en lenguaje Z a partir de un modelo entidad-relación.
- Exactitud y completitud: Siempre y cuando el modelo entidad-relación esté bien construido la salida generada en formato TEX podrá ser verificada con un chequeador de tipos como FUZZ sin generar errores.

- Disponibilidad de la información: El tiempo de respuesta deberá ser menor de 2 segundos, exceptuando la generación de la salida que depende directamente del tamaño del modelo entidad-relación.

#### **4.3.3.2. Confiabilidad**

Es el grado en que se esperaría que un programa desempeñe su función con la precisión requerida. Las dimensiones son las siguientes:

- Confiabilidad operativa: Según el sitio Web<sup>1</sup> -donde está alojada la herramienta- la frecuencia de falla es menor al 1%.
- Tolerancia a fallas: Después de ocurrir una falla y el hosting se restablezca la herramienta por estar orientada a Web se restablecerá inmediatamente.
- Recuperación: La herramienta por ser un prototipo solamente está provista de operaciones de almacenamiento manual, es responsabilidad del usuario tener una copia actualizada de su trabajo.

#### **4.3.3.3. Eficiencia**

Es la cantidad de recursos necesarios para que el programa realice su función. Este requerimiento está definido bajo las siguientes dimensiones:

- Capacidad de procesamiento: Al utilizar arquitecturas cliente-servidor, dependerá del equipo del cliente que tanta capacidad de procesamiento tenga, del lado del servidor se garantiza el procesamiento.
- Capacidad de almacenamiento: El sitio cuenta con un espacio de almacenamiento de 1500 MB, con lo cual se considera suficiente para la tarea de la herramienta.
- Capacidad de comunicación: El sitio garantiza una tasa de transferencia al mes de 100 GB, lo cual es suficiente para transferir “muchas” especificaciones que no sobrepasan los 100 KB.

---

<sup>1</sup> <http://www.000webhost.com>



#### **4.3.3.4. Integridad**

Es el grado de control sobre el acceso al software y datos por parte de personas no autorizadas. Las dimensiones son las siguientes:

- Control de acceso: La utilización está disponible para cualquier usuario, porque esto no daña ni la herramienta ni los datos de otros usuarios.
- Seguridad: Se debe proveer seguridad para evitar que dos o más usuarios utilicen el mismo archivo del modelo entidad-relación para garantizar la integridad de la información.
- Confidencialidad: Se debe proveer de un mecanismo que brinde seguridad y confidencialidad de los datos almacenados en cada modelo entidad-relación.

#### **4.3.3.5. Facilidad de uso**

Es el esfuerzo necesario para operar los datos de entrada de un programa e interpretar la salida. Este atributo tiene las diferentes dimensiones:

- Facilidad de operación: Se debe proveer una interfaz amigable que permita la fácil construcción del modelo entidad-relación y el manejo de operaciones relacionadas.
- Facilidad de comunicación: Se debe definir un formato sencillo de comprender para almacenar el modelo entidad-relación como entrada del sistema, y la salida debe tener un formato asimilable.
- Facilidad de aprendizaje: Se debe proveer un tutorial para que el usuario se pueda familiarizar con el software.

Los requerimientos asociados con la capacidad de experimentar cambios (revisión del producto) son: la facilidad de mantenimiento, la flexibilidad y la facilidad de prueba; se muestran a continuación:

#### **4.3.3.6. Facilidad de mantenimiento:**

Es el esfuerzo necesario para localizar y corregir un error en un programa, se describe por las siguientes dimensiones:

- El tamaño de cada archivo que representa un módulo principal debe tener alrededor de 300 líneas de código.
- La programación se debe realizar en algún IDE para manejar el proyecto de manera estandarizada y ordenada.

#### **4.3.3.7. Flexibilidad**

Es el esfuerzo necesario para modificar un programa en operación, se muestran las siguientes dimensiones:

- La herramienta debe ser capaz de manejar diferentes notaciones de asociaciones (asociación por participación y referencia al opuesto) para no limitar el funcionamiento.
- El formato de entrada debe ser simple y auto descriptible para que pueda entenderse sin necesidad de utilizar la herramienta y el formato de salida también debe ser simple y capaz de ser extendido para complementar la especificación formal.

#### **4.3.3.8. Facilidad de prueba**

Es el esfuerzo que demanda probar un programa con el fin de asegurar que realice su función.

- El software debe tener funciones precisas de modo que puedan ejecutarse de manera independiente para probar su funcionalidad, el nivel de dependencia con cualquier otro software o biblioteca debe ser mínimo.

Los requerimientos asociados con la capacidad de adaptarse a nuevos entorno (transición del producto) son: la portabilidad, la facilidad de reutilización y la interoperabilidad; se muestran a continuación:

#### **4.3.3.9. Portabilidad**

Es el esfuerzo necesario para transferir el programa de un entorno de hardware o software a otro, las dimensiones son las siguientes:

- Se requiere que la herramienta pueda ser ejecutada independiente de cualquier sistema operativo, por ello se basa en Web. (ej. Windows o Linux)
- Además de ser independiente del sistema operativo también debe ser independiente del navegador de internet. (ej. IE o Firefox)
- Tanto el formato de entrada como el de salida de datos debe establecerse un formato portátil y simplificado.

#### **4.3.3.10. Facilidad de reutilización**

Es el grado en que un programa puede reutilizarse en otras aplicaciones. Las dimensiones definidas son:

- Debe tratarse de desarrollar módulos atómicos que puedan adaptarse fácilmente a otros programas; un módulo de PHP puede ser extraído para ejecutarse de manera independiente, por ejemplo el módulo de generación debe recibir los parámetros y ejecutarse no debe ser dependiente de otro módulo.
- Además del código, la entrada y salida se pretende que no sea de formato exclusivo de la herramienta sino que sea de fácil intercambio.

#### **4.3.3.11. Interoperabilidad**

Es el esfuerzo necesario para acoplar un sistema con otro. Las dimensiones son:

- Los datos de entrada y de salida de la herramienta deben elaborarse bajo estándares comunes para habilitar un posible intercambio futuro con otras herramientas.

### **4.4. Decisiones de diseño**

En esta parte se describen diferentes decisiones respecto del diseño de la herramienta; las decisiones son respecto del cumplimiento de los requerimientos no funcionales, la notación del modelo entidad-relación y el formato de almacenamiento.

#### **4.4.1. Decisiones respecto de los requerimientos no funcionales**

Respecto de cómo garantizar el cumplimiento de los requerimientos no funcionales se tomaron decisiones que coadyuven en la calidad del software. Estas decisiones están clasificadas según cada requerimiento propuesto anteriormente:

- Integración: Para manejar un formato común de intercambio entre modelos se definió un tipo de documento XML para representar el esquema relacional del modelo. También el formato del archivo generado con la especificación de Z está definido con un lenguaje de marcas para exportarlo a cualquier herramienta de Latex.
- Portabilidad: Se decidió desarrollar el producto bajo un ambiente Web, para permitir la portabilidad de la herramienta a cualquier plataforma, también el uso desde cualquier cliente HTTP.

- Usabilidad: El principal objetivo del producto es que por medio de un diagrama entidad-relación -que es ampliamente utilizado y relativamente más fácil- se pueda generar una especificación equivalente en lenguaje Z.
- Funcionalidad: Cumpliendo todos los requerimientos funcionales se espera brindar un producto altamente funcional en lo que respecta a especificar modelos de datos.
- Rendimiento: El producto utiliza Apache como servidor Web, utilizando PHP como lenguaje de programación, este provee un alto rendimiento en cuanto a tiempo de respuesta.
- Disponibilidad: El producto se publicó en un sitio Web<sup>2</sup> para que la disponibilidad del mismo sea 24/7.

#### **4.4.2. Notación del modelo entidad-relación utilizada**

La notación del modelo entidad-relación utilizada es la notación ESB [4], se seleccionó esta notación porque utiliza aspectos importantes de las cardinalidades explícitas para mejorar la facilidad de uso al usuario; esta notación también pretende utilizar las ventajas de varios modelos para aprovecharlo en el sentido cognitivo para diagramar.

La cardinalidad de las relaciones se pueden expresar a partir de la cardinalidad mínima y máxima. La cardinalidad mínima puede tener dos valores 0 ó 1, y la cardinalidad máxima también puede tener dos valores 1 ó N. Dado que una relación intervienen dos entidades, existen cuatro valores variables:

$$\text{CardMin..CardMax} - \text{CardMin..CardMax}$$

Esto produce 16 diferentes combinaciones posibles.

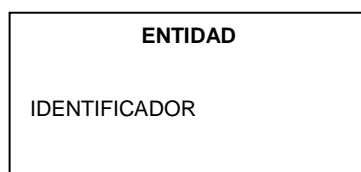
---

<sup>2</sup> <http://erz.comli.com>

El esquema conceptual especifica la estructura para todos los estados permitidos y transiciones de la base de datos [12]. Siendo así una manera expresiva de representar un sistema del mundo real, hay que notar que en este esquema aún no se mencionan términos relacionales.

Entre algunos componentes del esquema conceptual están:

- Entidad: Es cualquier objeto distinguible. (Se representa como un rectángulo con nombre)
- Atributos: Son propiedades que corresponden a la entidad.
- Relación: Representa la asociación entre entidades.



**FIGURA No. 4.3 Ejemplo de una entidad en el esquema conceptual.**

Con respecto de la dependencia de existencia, la definición de identificadores es importante. Hay dos casos por considerar: un caso mencionado en [13] es cuando la entidad depende existencialmente de otra o llamada entidad débil, a este se le denominará “identificador dependiente”; y el segundo caso es cuando no depende, a este se le denominará “identificador independiente”, ambos casos se representan así:

Identificador Dependiente:

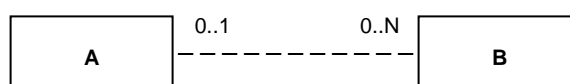
\_\_\_\_\_

Identificador Independiente:

-----

En este trabajo, a diferencia de la propuesta inicial de ESB -que solo pretende utilizar siete casos de asociaciones-, se utilizarán 10 casos de asociación que incluyen las asociaciones de muchos a muchos y reflexivas descritas en el capítulo 3.

Un ejemplo del caso 1 es la asociación de uno a muchos donde la cardinalidad mínima de ambas entidad es opcional que se muestra a continuación:



**FIGURA No. 4.4 Ejemplo de asociación de uno a muchos opcional**

#### **4.4.3. Descripción del formato de entrada XML**

Existe un punto intermedio para lograr una integración efectiva entre diferentes paradigmas de especificación de software, en esta tesis se propone utilizar un archivo de intercambio basado en XML para lograr una transformación automática, debido a la facilidad de manipulación de dicho formato.

La razón por la cual se seleccionó XML para almacenar el esquema de la base de datos fue porque dicho esquema se compone de metadatos, lo cual, se acopla muy bien con XML, además de ser un formato portátil. Se propone una definición estándar de esquemas relacionales en XML, la finalidad de definir estándares es algo muy común, y resulta necesario para realizar transformaciones entre los diferentes esquemas definidos, por ejemplo, de ANSI/SPARC [6] y también con especificaciones formales, para propósitos de integración.

El esquema XML es un lenguaje que describe la estructura y restricciones de los documentos XML, que mejoró los problemas que tenía la definición de tipo de documentos (DTD). En otro sentido representa los metadatos para un documento XML asociado o clases de documentos XML[33].

La definición anterior de los elementos se pueden expresar por medio de un esquema XML de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wmh="http://www.wmhhelp.com/2003/eGenerator"
elementFormDefault="qualified">
  <xs:element name="relation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="entities"/>
        <xs:element ref="relationships"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string"/>
      <xs:attribute name="version" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="entities">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="entity" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="entity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="attributes"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="x1" type="xs:string" use="required"/>
      <xs:attribute name="y1" type="xs:string" use="required"/>
      <xs:attribute name="x2" type="xs:string" use="required"/>
      <xs:attribute name="y2" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="attributes">
    <xs:complexType>
      <xs:sequence>
```



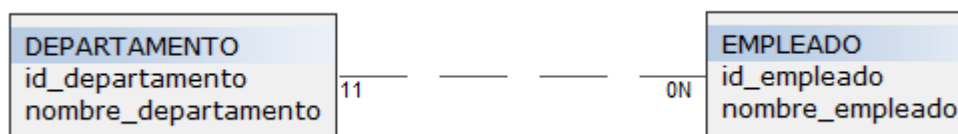
```

        <xs:element ref="attribute" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="attribute">
    <xs:complexType>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="type" type="xs:string" use="required"/>
        <xs:attribute name="key" type="xs:string" use="required"/>
        <xs:attribute name="null" type="xs:string" use="required"/>
        <xs:attribute name="increment" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="relationships">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="relationship"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="relationship">
    <xs:complexType>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="id_parent" type="xs:string" use="required"/>
        <xs:attribute name="id_child" type="xs:string" use="required"/>
<xs:attribute name="legend_parent" type="xs:string" use="required"/>
<xs:attribute name="legend_child" type="xs:string" use="required"/>
        <xs:attribute name="min_parent" type="xs:string" use="required"/>
        <xs:attribute name="max_parent" type="xs:string" use="required"/>
        <xs:attribute name="min_child" type="xs:string" use="required"/>
        <xs:attribute name="max_child" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
</xs:schema>

```

#### 4.4.4. Ejemplo de notación ER y formato de entrada

A manera de ejemplo, y para ayudar a la comprensión de la estructura que debe tener el XML se toma el siguiente diagrama simple:



**FIGURA No. 4.5 Modelo entidad-relación de ejemplo**

El anterior diagrama conceptualiza la relación que existe entre empleados y departamento, es decir, un empleado pertenece a uno y solo un departamento, y que un departamento tiene cero o muchos empleados asignados.

Cada entidad posee solamente dos atributos para simplificar el ejemplo: el identificador y un atributo de nombre. Esta simplificación se debe a que se quiere mostrar la descripción de atributos dentro del XML y con dos atributos es suficiente.

La cardinalidad 1..1 del departamento representa que todo empleado pertenece a un departamento, no hay nulos en la llave foránea de la entidad empleado, es decir, que cuando un empleado ingresa a la empresa tendrá asignado siempre un departamento específico.

Mientras que la cardinalidad 0..\* del empleado representa que un departamento puede no tener empleados asignados, por ejemplo, si los empleados asignados renuncian y no hay reemplazos aún.

La línea de asociación discontinua indica que las entidades tienen identificadores independientes, es decir, cada entidad cuenta con su propia identificación y sus identificadores no dependen mutuamente, por lo que ambas entidades tienen llaves primarias diferentes.

En el XML hay elementos que corresponden a conceptos lógicos del diagrama entidad-relación y también elementos que describen características propias del dibujo como las coordenadas. El archivo de intercambio o la definición en XML del anterior diagrama se muestra a continuación:

```

<?xml version="1.0" ?>
<relation name="xml/emplea.xml" pass="emplea" version="1.0"
relationship_type="Opposite">
  <entities>
    <entity name="empleado" x="586px" y="285px">
      <attributes>
        <attribute name="id_empleado" type="" width="12" scale="0"
          null="false" identity="false" seed="0" increment="0"
          pk="true" />
        <attribute name="nombre_empleado" type="" width="12"
          scale="0" null="false" identity="false" seed="0"
          increment="0" pk="false" />
      </attributes>
    </entity>
    <entity name="departamento" x="238px" y="286px">
      <attributes>
        <attribute name="id_departamento" type="" width="12" scale="0"
          null="false" identity="false" seed="0" increment="0"
          pk="true" />
        <attribute name="nombre_departamento" type="" width="12"
          scale="0" null="false" identity="false" seed="0"
          increment="0" pk="false" />
      </attributes>
    </entity>
  </entities>
  <relationships>
    <relationship parent="departamento" child="empleado"
      card_parent="11" card_child="0N" dependency="true" />
  </relationships>
</relation>

```

## 4.5. Arquitectura del software

### 4.5.1. Representación arquitectónica

La representación arquitectónica es mostrada por medio de vistas:

- Vista de casos de uso: Por medio de un diagrama de casos de uso primarios se representan las funcionalidades generales del sistema.

- Vista lógica: Describe partes importantes del diseño como subsistemas y paquetes, asimismo la descripción de operaciones.
- Vista de proceso: Describe los procesos organizados por grupos para comunicación o interacción.
- Vista de despliegue: Describe la configuración de red física en que el software es desplegado, indicando nodos de ejecución.
- Vista de implementación: Describe la estructura del modelo de implementación, la descomposición del software en capas y componentes.

#### **4.5.2. Objetivos de arquitectura y limitaciones**

Entre los principales objetivos de la arquitectura resaltan seis atributos -discutidos en las decisiones del diseño- que proveen principalmente: la integración entre especificaciones por medio de un archivo de intercambio en formato XML; este formato coadyuva a la portabilidad de los datos, cualquier herramienta externa puede hacer uso de la información conociendo el esquema de los datos; otra parte primordial es la simplicidad de la herramienta, de modo que el usuario final puede hacer uso de la herramienta de una manera fácil y rápida; esta simplicidad no debe afectar la funcionalidad del sistema, se definieron casos de uso que describen los requerimientos funcionales de la herramienta; además, de presentar aspectos simples, también los datos tienen un esquema simple de estructuración para que el rendimiento en cuanto al procesamiento y comunicación sea eficiente; y por último, se decidió utilizar un esquema de “software como servicio” orientado a Web para que la herramienta siempre esté disponible.

La herramienta de software es un prototipo funcional, por tanto, existen limitaciones en cuanto a concurrencia de acceso al almacenamiento y administración de usuarios; estos factores no alteran la funcionalidad de la integración entre paradigmas de especificación de software, por tanto, no son indispensables.

Para estandarizar el desarrollo se utilizó el IDE NetBeans versión 6.8 para desarrollo en PHP que permite establecer código limpio, ordenado y preciso.

### 4.5.3. Vista de casos de uso

La vista de casos de uso representa las operaciones fundamentales de cada tipo de usuario, en este caso, son los casos de uso primarios para el usuario y el usuario con contraseña. Para más detalle se puede ver el diagrama completo de los casos de uso en la parte de requerimientos específicos de este capítulo.

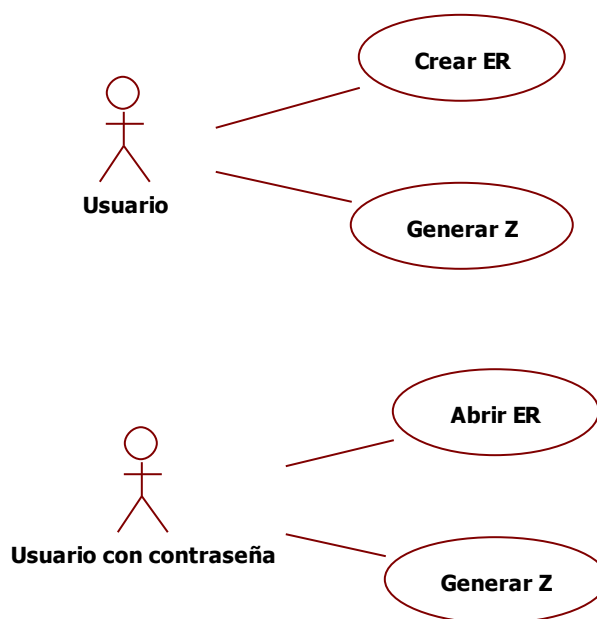
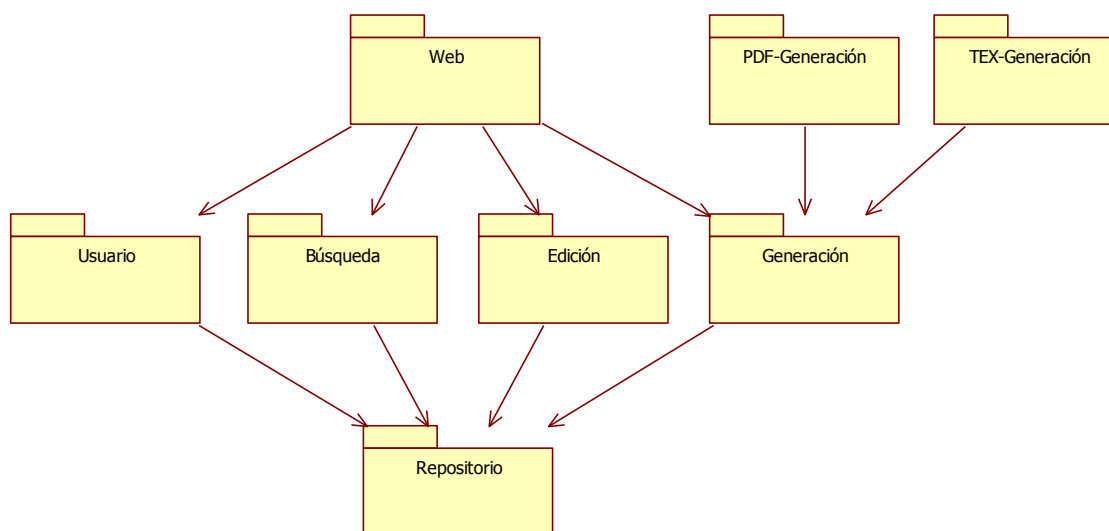


FIGURA No. 4.6 Vista de casos de uso primarios

#### 4.5.4. Vista lógica

Aunque el enfoque de la herramienta de software no es orientado a objetos, un diagrama de paquetes describe muy bien los subsistemas y operaciones que están relacionadas con la herramienta.

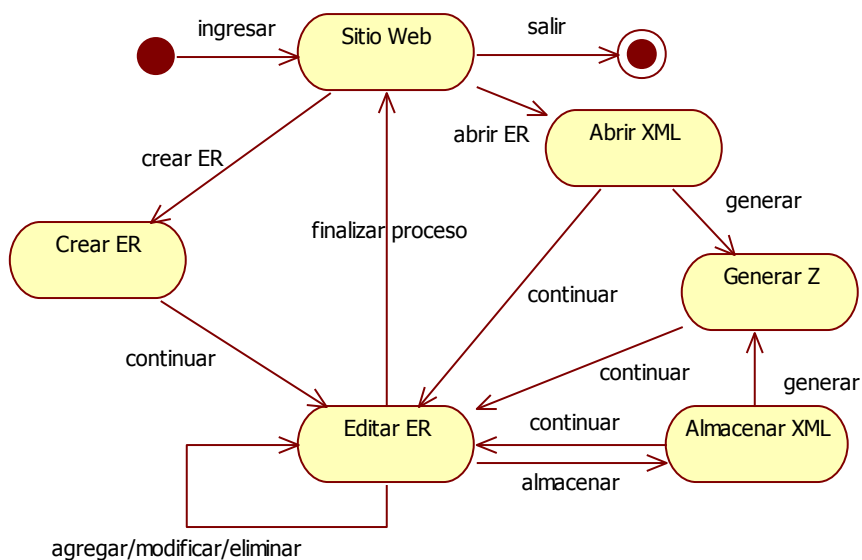


**FIGURA No. 4.7 Diagrama de paquetes del software**

El sitio Web dependiendo de quién sea el usuario puede buscar y editar un ER, y generar una especificación en Z; esta especificación puede generarse en formato PDF o TEX; todos los paquetes hacen uso del repositorio que almacena los archivos XML.

#### 4.5.5. Vista de proceso

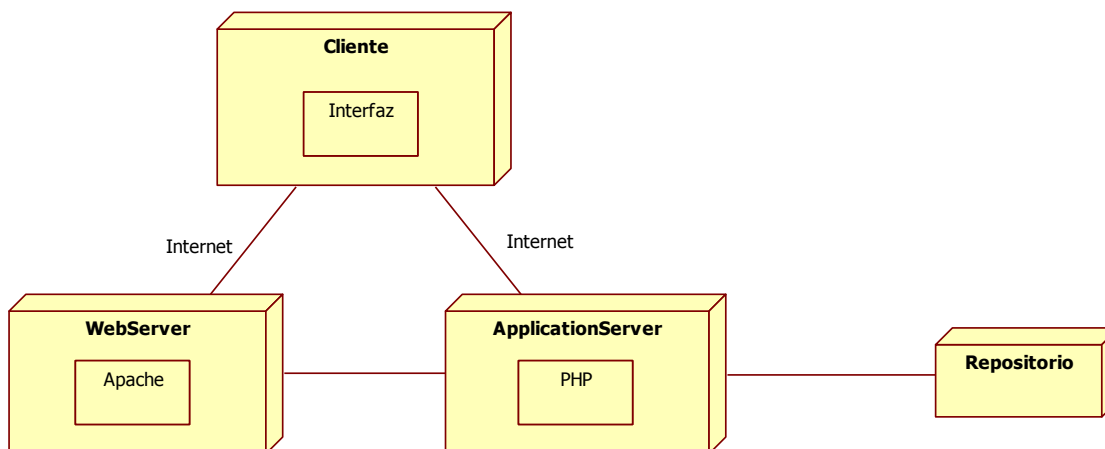
La vista de proceso se puede representar con un diagrama de actividades que muestra los diferentes procesos que contiene el software.



**FIGURA No. 4.8** Diagrama de actividades del software

#### 4.5.6. Vista de despliegue

El despliegue se basa en tres capas que serán explicadas en la vista de implementación; la capa del cliente cuenta con una interfaz o navegador Web, esta se comunica con la capa de aplicación; esta capa cuenta con dos servidores: de internet y de aplicación donde reside PHP y donde la herramienta estará alojada para comunicación con el repositorio.



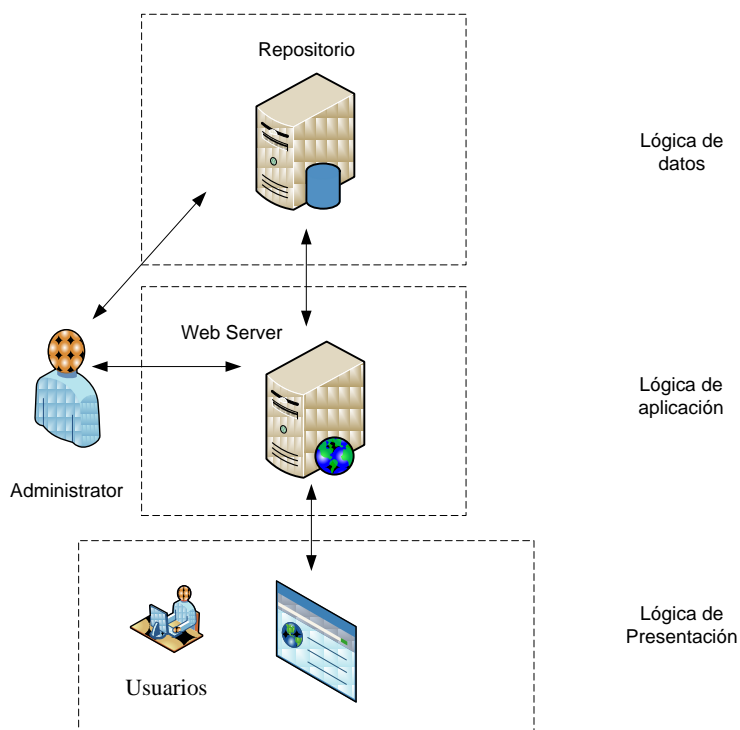
**FIGURA No. 4.9** Diagrama de despliegue del software

#### 4.5.7. Vista de implementación

La vista de implementación está basada en tres capas:

- Capa de presentación
- Capa de aplicación
- Capa de datos

La capa de presentación está dividida en dos partes, una ejecutada en el cliente por medio de JavaScript para el manejo gráfico de la herramienta, por otra parte, se ha utilizado PHP como lenguaje de programación Web y que inherentemente se ejecuta en el servidor. Los resultados son mostrados en el cliente por medio de HTML.



**FIGURA No. 4.10** Arquitectura de tres capas de la herramienta de software



La capa de aplicación basada en PHP es la encargada de hacer diferentes transformaciones de datos entre el modelo entidad-relación y su equivalente representación en Z, además, también se maneja la administración del archivo de intercambio o XML y la presentación del lenguaje Z en estilo PDF y TEX.

La capa de datos conformada por la lógica de acceso y archivos planos de almacenamiento es ejecutada en el servidor, para trabajos futuros se espera tener una base de datos para el manejo de usuarios y archivos para evitar los problemas de concurrencia en la edición de archivos.

## **5. DESARROLLO DE LA HERRAMIENTA DE SOFTWARE**

En este capítulo se mostrará la manera en que se realizó el proceso de implementación de la herramienta de software propuesta en el anterior capítulo. Básicamente, la importancia de este capítulo es mostrar el enfoque de desarrollo de software utilizado, decisiones de implementación, utilización de bibliotecas y la validación del análisis y diseño contra la puesta en marcha de la herramienta.

### **5.1. Enfoque de desarrollo**

El enfoque de desarrollo utilizado fue híbrido, por una parte se utilizó el Proceso Unificado para la definición de artefactos por medio de un análisis y diseño simplificado, y por otra parte se utilizó un método ágil de desarrollo, donde se definieron ciertas iteraciones agrupadas a partir de dos fases principales.

Cada fase consta de iteraciones o “sprint” (como lo denomina Scrum). Cada sprint tiene una duración de 30 días. El desarrollo fue híbrido, tal como se mencionó, ya que se tienen aspectos de diferentes métodos de desarrollo: la duración de cada sprint fue tomada de Scrum; la duración total del desarrollo fue de 6 meses, similar a la duración de los proyectos de eXtreme Programming (XP), aunque solamente el autor realizó la programación y XP sugiere programación en parejas; ciertos requerimientos fueron evolucionando durante el desarrollo y fueron introducidos en la herramienta, una manera de desarrollo utilizada en EVO; y por último toda la documentación del análisis y diseño se realizó con base en ciertos aspectos del Proceso Unificado y UML.

Se definieron dos fases principales, cada una relacionada con un módulo:

- Módulo de diagramación del modelo entidad-relación.
- Módulo de conversión del modelo entidad-relación a lenguaje Z.

### 5.1.1. Diagramador entidad-relación

El módulo que consiste en el diagramador del modelo entidad-relación se realizó con lenguaje JavaScript y posee un mecanismo de drag and drop y menú contextual para administrar las diferentes entidades y relaciones.

Se definieron tres iteraciones que se detallan a continuación:

#### 5.1.1.1. Iteración 1: Diagramado

Esta iteración consiste en la implementación del diagramador, cuyo objetivo es crear objetos, llamados entidades, que pueden moverse a través de la pantalla del navegador por manipulación directa (estilo “drag and drop”). Cada entidad puede tener ciertos atributos y estos atributos poseer ciertas propiedades que lo describen como tipo de dato, si es identificador, etc.

Para conseguir la administración por entidad se introdujo un menú contextual que permite editar y eliminar la entidad, así como sus atributos y relaciones con otras entidades.

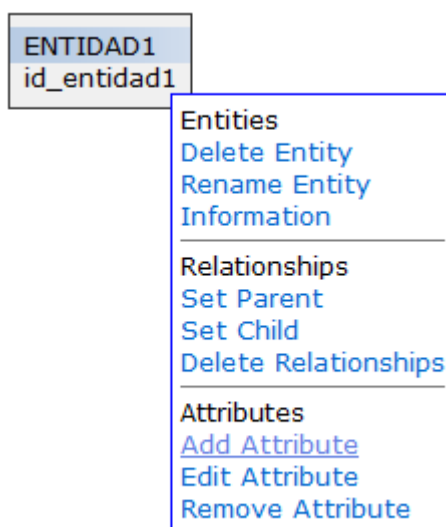


FIGURA No. 5.1 Menú contextual de entidad

En la siguiente figura se muestra la pantalla de ingreso de atributos. Es importante destacar que muchos datos no se utilizan aún en la especificación de lenguaje Z por la limitación de este trabajo, pero para trabajos futuros se dejan previstos.

Add Attribute:

Entity: entidad1

Name: id\_entidad1

Type: Char

Width & Scale: 12 0

Allow Nulls?

Identity?

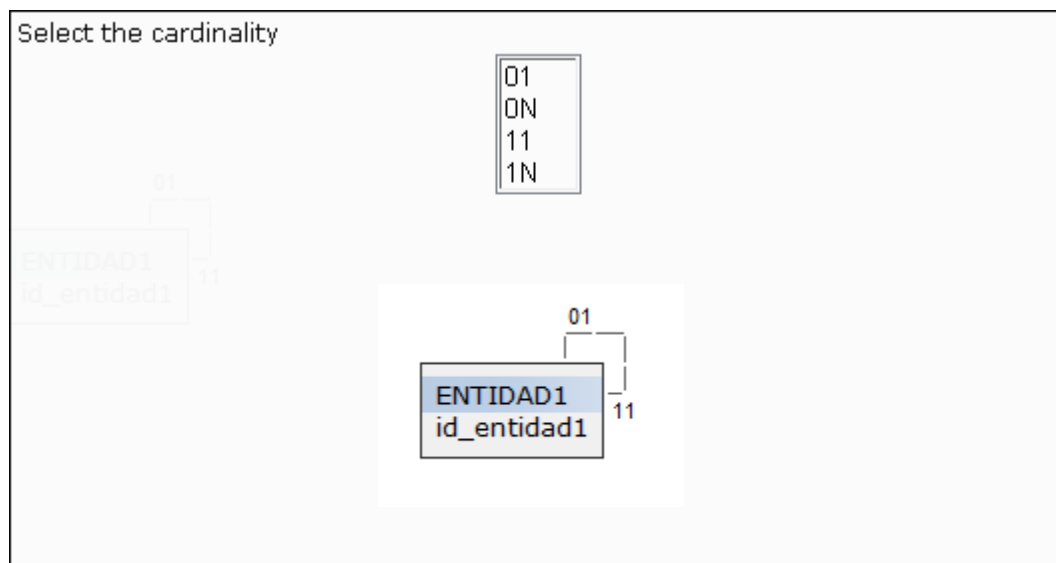
Seed & Increment: 0 0

Primary Key?

Add Close

**FIGURA No. 5.2 Opciones para agregar atributos**

Para crear una asociación es necesario ingresar la cardinalidad mínima y máxima para crear la asociación como se muestra a continuación:



**FIGURA No. 5.3 Opciones para agregar cardinalidad a cierta relación**

### 5.1.1.2. Iteración 2: Almacenamiento

El almacenamiento se manejó mediante un XML, tal como se mencionó en el análisis y diseño, el cual tiene la estructura idónea para almacenar la parte estática del modelo entidad-relación. A continuación se muestra un ejemplo:



FIGURA No. 5.4 Ejemplo para mostrar el almacenamiento

```
<?xml version="1.0" ?>
<relation name="xml/prueba.xml" pass="prueba" version="1.0"
relationship_type="Opposite">
  <entities>
    <entity name="departamento" x="258px" y="257px" />
    <entity name="empleado" x="568px" y="256px" />
  </entities>
  <relationships>
    <relationship parent="departamento" child="empleado"
card_parent="11" card_child="0N" dependency="true" />
  </relationships>
</relation>
```

Para crear el archivo anterior se necesita la información temporal del modelo. Mientras este se va editando se crean tres arreglos en JavaScript: entidades, atributos y relaciones. Luego a partir de una serialización de estos arreglos se envían a PHP y este por medio de instrucciones de DOMDocument almacena la estructura.

### **5.1.1.3. Iteración 3: Recuperación**

Este proceso es el inverso de la iteración 2, por medio de instrucciones DOMDocument de PHP se obtiene la estructura del XML y se van creando los arreglos necesarios en JavaScript, luego se ejecuta la función de mostrar modelo entidad-relación basado en los nuevos arreglos construidos a partir de un XML.

## **5.1.2. Convertidor de entidad-relación a lenguaje Z**

El convertidor es necesario para transformar un modelo entidad-relación al lenguaje Z. Existen dos tipos de transformaciones, una que muestra el lenguaje en formato PDF y otra que los muestra en formato TEX para que pueda ser compilado posteriormente por FUZZ con miras a comprobar sintaxis y tipos. LATEX es procesador de TEX que produce documentos en formato matemático.

### **5.1.2.1. Iteración 1: Marco de referencia**

Inicialmente se debe establecer un marco de referencia. Este fue definido en el capítulo de integración entre el modelo entidad-relación y el lenguaje Z. Para más detalle se puede ver dicho capítulo.

### **5.1.2.2. Iteración 2: Datos de salida en PDF**

Al tener el marco de referencia definido, se establece el formato de salida, que en este caso es PDF, por lo que se utiliza la transformación del modelo entidad-relación a su equivalente en Z para mostrarlo en PDF.

### Schemas for operations

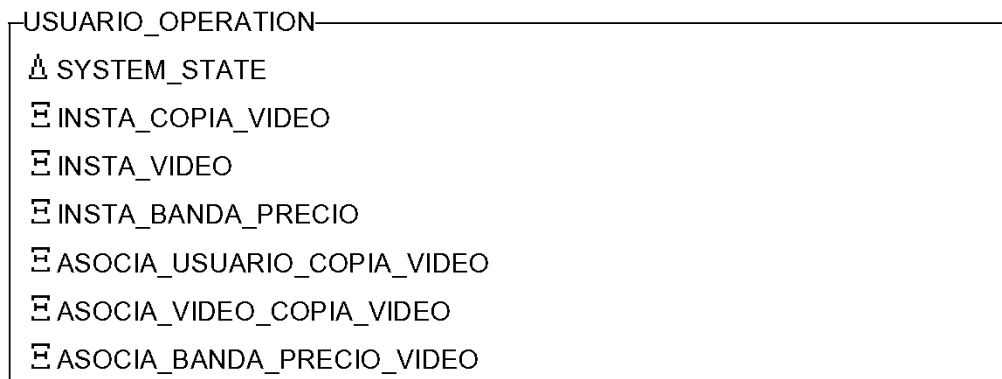


FIGURA No. 5.5 Formato de salida en PDF

#### 5.1.2.3. Iteración 3: Datos de salida en TEX

Como última iteración se utilizó también el formato TEX, este es muy útil debido a que se puede compilar con FUZZ para agregar funcionalidades a la especificación.

```

schemas for operations
\begin{schema}{USUARIO\_OPERATION}
\Delta SYSTEM\_STATE ||
\Xi INSTA\_COPIA\_VIDEO ||
\Xi INSTA\_VIDEO ||
\Xi INSTA\_BANDA\_PRECIO ||
\Xi ASOCIA\_USUARIO\_COPIA\_VIDEO ||
\Xi ASOCIA\_VIDEO\_COPIA\_VIDEO ||
\Xi ASOCIA\_BANDA\_PRECIO\_VIDEO
\end{schema}

```

FIGURA No. 5.6 Formato de salida en TEX

Posteriormente este documento puede ser susceptible a publicación por medio de Latex con los esquemas bien formado y con el adecuado formato matemático.

## **5.2. Decisiones de implementación**

Existen dos decisiones importantes de implementación, el paradigma y el lenguaje de programación.

### **5.2.1. Paradigma de programación**

Durante el inicio de la investigación se analizaron diferentes paradigmas de programación para seleccionar el más idóneo. Uno de los aspectos importantes por tomar en cuenta fue la portabilidad. No se quería enlazar la herramienta con un *framework* o con un sistema operativo en particular. Otro aspecto que se requería era cierto nivel de extensibilidad para que el prototipo desarrollado pueda evolucionar con el tiempo.

El paradigma seleccionado fue el desarrollo Web con orientación al software como servicio de Cloud Computing. El desarrollo Web permite crear aplicaciones portátiles y además permite escalabilidad.

El término software como servicio hace mención a que la herramienta de software desarrollada estará publicada en un sitio Web y podrá ser utilizada por cualquier persona como un servicio. En general, el usuario no debe preocuparse ni de la versión de lenguaje de programación, ni del sistema operativo y tampoco necesita saber si se requiere de bibliotecas adicionales.

### **5.2.2. Lenguaje y plataforma de programación**

Con el desarrollo Web como base para la implementación de la herramienta de software, falta definir el lenguaje de programación por utilizar. Es bien sabido que el desarrollo Web tiene una arquitectura cliente/servidor, debido a esto existen lenguajes de programación que se ejecutan en el cliente y otros que se ejecutan en el servidor.



El lenguaje que se seleccione del lado del cliente debe soportar el llamado *front-end* de la herramienta, es decir, debe tener las características necesarias para cubrir los requerimientos funcionales en lo que respecta al manejo de la interfaz.

Por otro lado, el lenguaje que se ejecuta del lado del servidor servirá como el llamado *back-end* de la herramienta, es decir, sería el lenguaje que soporte los procesos necesarios para la integración y transformación de las especificaciones.

Entre los lenguajes de lado del cliente predominantes se pueden mencionar a los basados en *scripts*, los basados en tecnologías Flash y los basados en tecnologías Java. Tal como se mencionó, se busca un lenguaje que sea independiente, portátil y escalable, los lenguajes relacionados con Flash y Java son dependientes de su tecnología, quedando los lenguajes basados en *scripts*. El lenguaje seleccionado es JavaScript debido a que es un lenguaje soportado por cualquier navegador, además es un lenguaje que tiene mucho auge y se ha vuelto muy robusto.

Entre los lenguajes del lado servidor hay tres opciones importantes: PHP, ASPX y JSP. Estos lenguajes son dependientes del servicio que ofrece el servidor Web, sin embargo, de cierta manera los lenguajes ASPX y JSP están muy apegados a su tecnología, en el caso de ASPX con el framework de Microsoft y en el caso de JSP con su máquina virtual (JVM). Por lo tanto el lenguaje seleccionado es PHP por su simplicidad de ejecución e interoperabilidad que presenta.

Para complementar a los lenguajes JavaScript y PHP se utilizará una técnica de desarrollo Web para hacer interactiva la herramienta de software llamada AJAX, manteniendo una comunicación asíncrona entre el cliente y servidor. Además es necesario destacar la utilización de XML como formato de intercambio entre las comunicaciones de ambas capas de la arquitectura.

### 5.3. Utilización de bibliotecas

Para facilitar el desarrollo de la herramienta se utilizaron principalmente tres aspectos: el menú contextual, para mostrar opciones de las entidades; hacer *drag and drop*, para mover entidades dentro de la página Web; y una biblioteca gráfica para dibujar asociaciones. Los primeros dos aspectos se elaboraron con base en tutoriales y el tercer aspecto con base en una biblioteca.

#### 5.3.1. Tutoriales utilizados

En [34] se encontraron dos tutoriales importantes para aprender el desarrollo de menús contextuales y la operación *drag and drop* o movimiento de etiquetas.

- El primero es el menú contextual. La idea es mostrar al usuario final las opciones que una entidad puede tener y no el menú general de la página Web, básicamente se trata de crear una etiqueta DIV con cierto estilo de lista que tienen enlaces que puede ejecutarse.
- El segundo es la operación *drag and drop* o movimiento de etiquetas, la idea es permitir al usuario final mover las entidades dentro de la página Web como si fuera un objeto.

#### 5.3.2. Biblioteca gráfica 2D para JavaScript

Para complementar la funcionalidad de la herramienta se utilizó la biblioteca jsDraw2D [35], esta es una biblioteca escrita en puro JavaScript para dibujar en sitios Web. La biblioteca es libre y gratis bajo la licencia LGPL

Específicamente esta biblioteca se utilizó solamente para dibujar las líneas de las asociaciones entre las entidades, por otra parte las entidades fueron dibujadas con HTML por medio de la etiqueta DIV que define una división en el documento HTML y se le da formato con estilos tipo CSS.

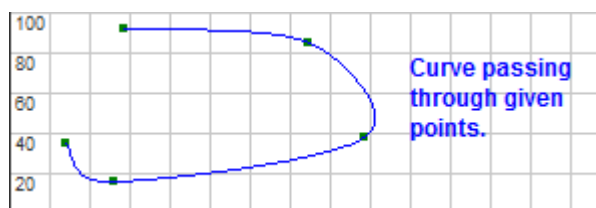


FIGURA No. 5.7 Segmento de diagrama publicado por jsDraw2D [35]

### 5.3.3. Biblioteca FPDF para PHP

FPDF [36] más que una biblioteca es una clase escrita en PHP que permite generar documentos PDF desde PHP; es una clase porque generalmente las bibliotecas deben configurarse con el servidor. Esta clase no utiliza la famosa biblioteca PDFLib. La F de FPDF es por *free* o gratis porque la clase puede ser utilizada para cualquier propósito y puede ser modificada.

La clase puede ser ejecutada ya sea en PHP4 o PHP5; la eficiencia en documentos muy extensos es menor si se le compara con la de la biblioteca PDFLib pero no es observable y para la herramienta es suficiente.



FIGURA No. 5.8 Logo de la biblioteca FPDF [36]

La razón principal de la selección de esta clase es por la independencia que conlleva, esta hace cumplir algunos requerimientos no funcionales respecto de portabilidad, reutilización e interoperabilidad; los archivos PDF generados por la especificación no son muy extensos por lo que no afecta la comparación con PDFLib respecto de la eficiencia.

#### **5.4. Plan y ejecución de pruebas**

En [19] se propone un plan de pruebas destinado a herramientas Web, no obstante, es importante mantener las pruebas como un proceso de ejercitar el software con el objetivo de encontrar errores, que en general se aplica a cualquier software. Específicamente para herramientas Web se toman en cuenta diferentes dimensiones que buscan la calidad del software.

En general, el plan debe contener el conjunto de tareas, los productos de trabajo y la manera en que se ejecute cada tarea.

La estrategia de prueba por utilizar es el de caja negra, debido a que la herramienta se publicó en un sitio Web para obtener realimentación. La diferencia entre la estrategia de caja negra y de caja blanca es que las de caja negra no es necesario tener la documentación del diseño de la herramienta, sino que está orientado a medir las diferentes entradas y salidas de la herramienta de software.

Es importante mencionar que además de las pruebas documentadas a continuación, que son de la herramienta ya desarrollada, se realizaron también pruebas unitarias de cada módulo e iteración a fin de verificar su funcionalidad. Es importante destacar que los casos de prueba son los idóneos para detectar un defecto en la herramienta, además hay varios casos de prueba que no fueron documentados porque el resultado es muy obvio y el resultado será satisfactorio. Y que los casos no satisfactorios encontrados fueron corregidos para garantizar la funcionalidad de la herramienta.

El plan incluye casos de prueba de los siguientes aspectos:

- Prueba de contenido
- Prueba de interfaz
- Prueba de navegación
- Prueba de componentes
- Prueba de configuración
- Prueba de seguridad
- Prueba de desempeño

#### 5.4.1. Prueba de contenido

El objetivo de la prueba de contenido es descubrir errores sintácticos, semánticos y de estructura del contenido presentado al usuario final.

**CUADRO No. 5.1 Resultados de los casos de pruebas de contenido**

NO. CASO	DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO	APROBADO / NO APROBADO
C01	Ingresar una cadena vacía como nombre de un modelo.	La herramienta no debe crear ningún modelo e indicar el error.	La herramienta no creó ningún modelo.	APROBADO
C02	Ingresar espacios en blanco en el nombre de una entidad.	No debe crear la entidad e indicar el error.	Se crea la entidad con espacios en blanco.	NO APROBADO (Defecto 1)
C03	Ingresar una cadena vacía como nombre de una entidad.	No debe crear la entidad e indicar el error.	No crear la entidad.	APROBADO
C04	Crear una entidad con un nombre ya existente.	No debe crear la entidad e indicar el error.	No se crea la entidad.	APROBADO

C04	Crear un atributo como llave primaria que permita nullos.	No debe realizarse la acción e indicar el error.	Se crea el atributo.	NO APROBADO (Defecto 2)
C05	Ingresar una cadena vacía como nombre de un atributo.	No crear el atributo e indicar el error.	Se crea el atributo.	NO APROBADO (Defecto 3)
C06	Editar un atributo específico.	Mostrar la lista de atributos para editar.	Se muestra la lista y se edita bien el atributo.	APROBADO
C07	Crear un atributo incremental no numérico.	No debe crear el atributo.	Se crea el atributo.	NO APROBADO (Defecto 4)
C08	Crear un atributo con longitud 0.	No debe crear el atributo.	Se crea el atributo.	NO APROBADO (Defecto 5)

#### 5.4.2. Prueba de interfaz

El objetivo de la prueba de interfaz es probar los mecanismos de interfaz según las reglas de diseño y los casos de uso, además verificar la semántica y la facilidad de uso.

La herramienta se publicó en un sitio Web y se realizaron pruebas por parte de usuarios externos, pruebas tipo beta, y la realimentación fue la siguiente:

- El marco de trabajo es fijo y no se redimensiona con la ventana del navegador, en modelos grandes puede verse afectado el diagramador.
- Al crear entidades aparecen ubicadas en el mismo lugar y pueden transponerse entre sí, lo que puede provocar problemas al usuario.
- Las relaciones no tienen menú contextual, por lo que se hace difícil la facilidad de uso.

Además es importante verificar que la programación de la interfaz esté de acuerdo a los estándares de HTML y CSS. En [37] y [38] están los verificadores de HTML y de CSS de la W3C, el sitio se sometió a dichos verificadores y los resultados fueron los siguientes:

Con respecto del HTML se encontraron 3 advertencias, relacionadas con el SHORTTAG YES, el problema radica en la doble interpretación que puede tener y que ciertos navegadores pueden tener problema con eso.

- *Line 6, column 65*: NET-enabling start-tag requires SHORTTAG YES  

```
<link rel="stylesheet" type="text/css" href="style.css" !>
```
- *Line 76, column 91*: NET-enabling start-tag requires SHORTTAG YES  

```
<th colspan="2">
```
- *Line 111, column 117*: NET-enabling start-tag requires SHORTTAG YES  

```
...="http://analytics.hosting24.com/count.php" alt="web hosting" !></a></noscript>
```

**FIGURA No. 5.9 Resultados de la verificación de HTML de la W3C**

Con respecto de CSS se encontraron 6 errores, todos están relacionados con la propiedad OPACITY que en la nuevas versiones de CSS está descartada.

Disculpas! Hemos encontrado las siguientes errores (6)		
URI : <a href="http://erz.comli.com/style.css">http://erz.comli.com/style.css</a>		
15	div.dialogBackground	Error de análisis sintáctico opacity=50)
16	div.dialogBackground	La propiedad -moz-opacity no existe : 0.5 0.5
17	div.dialogBackground	La propiedad opacity no existe en CSS versión 2.1 pero existe en : 0.5 0.5
29	div.dialogBox	Error de análisis sintáctico opacity=95)
30	div.dialogBox	La propiedad -moz-opacity no existe : 0.95 0.95
31	div.dialogBox	La propiedad opacity no existe en CSS versión 2.1 pero existe en : 0.95 0.95

**FIGURA No. 5.10 Resultados de la verificación de CSS de la W3C**

### 5.4.3. Prueba de navegación

El objetivo de la prueba de navegación es verificar la sintaxis y semántica de las rutas de navegación de la herramienta.

Las rutas de navegación en la herramienta son escasas y todas pasaron las pruebas:

- Nuevo XML: Crear y cerrar.
- Abrir XML: Abrir y cerrar.
- Subir XML: Subir y cerrar.
- Contacto: Enviar y cerrar
- Manual
- Agregar entidad
- Guardar modelo
- Ver XML
- Almacenar XML localmente
- Mostrar PDF con especificación Z
- Mostrar TEX con especificación Z

### 5.4.4. Prueba de componentes

El objetivo de la prueba de componentes es descubrir errores en las funciones de la herramienta.

**CUADRO No. 5.2 Resultados de los casos de prueba de componentes**

NO. CASO	DESCRIPCIÓN	RESULTADO ESPERADO	RESULTADO OBTENIDO	APROBADO / NO APROBADO
C09	Subir un archivo XML mal formado.	No mostrar modelo e indicar errores.	No muestra modelo.	APROBADO



C10	Subir un archivo de diferente formato.	No debe subir el archivo.	Sube el archivo.	NO APROBADO (Defecto 6)
C11	Generar una especificación (pdf o tex) sin haber almacenado el modelo.	Mostrar un error y no mostrar la especificación.	No muestra la especificación pero no maneja de excepción.	NO APROBADO (Defecto 7)

#### 5.4.5. Prueba de configuración

El objetivo de la prueba de configuración es verificar por un lado los factores de configuración relacionados con el servidor y por otro lado los factores de configuración relacionados con el cliente. En general, los aspectos de configuración se refieren al sistema operativo, sistema de archivos, servidores, etc.

Del lado del cliente se probó con dos navegadores: Internet Explorer y Firefox.

Con ambos navegadores las pruebas fueron satisfactorias, sin embargo, con terceros navegadores como Chrome dio problemas con el IFRAME por compatibilidad. Sin embargo, desde el diseño de la herramienta se limitó a trabajar con los dos navegadores principales, cualquier compatibilidad con otros sería un trabajo futuro de investigación.

También del lado del navegador se probó con los sistemas operativos Windows y Linux, y las pruebas también fueron satisfactorias.

Del lado del servidor se utilizó la siguiente configuración:

- Sistema operativo: Linux 2.6.18-92.1.22 i686
- Software: Apache 2.0
- Lenguaje: PHP/5.2.10

#### 5.4.6. Prueba de seguridad

El objetivo de la prueba de seguridad es probar la vulnerabilidad en el ambiente tanto del lado del cliente como del lado del servidor.

Del lado del cliente se utilizó la herramienta Browser Security Test [39], esta herramienta ejecuta pruebas a navegadores, la prueba se realizó en una computadora con sistema operativo Windows Vista, con navegador MSIE versión 7.0, los resultados se muestran en la siguiente figura.

##### Test results

- ✓ Internet Explorer bait & switch race condition - passed
- ✓ Internet Explorer createTextRange arbitrary code execution - passed
- ✓ Windows MDAC ADODB ActiveX control invalid length - passed
- ✓ Adobe Flash Player video file parsing integer overflow - passed
- ✓ XMLDOM substringData() heap overflow - passed
- ✓ Apple QuickTime MOV file JVTCompEncodeFrame heap overflow - passed
- ✓ Apple QuickTime 'QTPlugin.ocx' ActiveX Control Multiple Buffer Overflows - passed
- ✓ Window location property cross-domain scripting - passed
- ✓ Internet Explorer XML nested SPAN elements memory corruption - passed

**Congratulations! The test has found no vulnerabilities in your browser!**

**FIGURA No. 5.11 Resultados de la prueba automática de seguridad del cliente**

Del lado del servidor se utilizó la herramienta Unmask Parasities[40], esta herramienta busca vulnerabilidades en el sitio buscando contenido ilícito, el resultado de la herramienta se muestra en la siguiente figura.

## Report

### General

**Title:** ERZ: Entity Relationship to Z Notation

**URL:** <http://erz.comli.com>

**Google:** not currently listed as suspicious\* ([details](#))

**Last checked:** 0 minutes ago (results are cached for 1 hour)

**This report:** <http://www.UnmaskParasites.com/security-report/?page=erz.comli.com>

**FIGURA No. 5.12 Resultados de la prueba automática de seguridad del servidor**

### 5.4.7. Prueba de desempeño

El objetivo de la prueba de desempeño es medir el tiempo de respuesta con pruebas de carga y tensión. Teniendo los datos de la medición, la finalidad es minimizarlos.

Para realizar este tipo de prueba se utilizó una herramienta de análisis [41], el reporte fue el siguiente:

#### Web Page Speed Report

URL: erz.comli.com

Title: ERZ: Entity Relationship to Z Notation

Date: Report run on Thu Aug 26 14:55:21EDT2010

#### Diagnosis

Global Statistics

Total HTTP Requests: 4

Total Size: 11663 bytes

## Object Size Totals

CUADRO No. 5.3 Tamaños totales de objetos de prueba

Object type	Size (bytes)	Download @ 56K (seconds)	Download @ T1 (seconds)
HTML:	6070	1.41	0.23
HTML Images:	4380	1.07	0.22
CSS Images:	0	0	0
Total Images:	4380	1.07	0.22
Javascript:	0	0	0
CSS:	1213	0.44	0.21
Multimedia:	0	0	0
Other:	0	0	0

## Download Times

CUADRO No. 5.4 Tiempo de descarga

Connection Rate	Download Time
14.4K	9.84 seconds
28.8K	5.32 seconds
33.6K	4.67 seconds
56K	3.12 seconds
ISDN 128K	1.51 seconds
T1 1.44Mbps	0.86 seconds

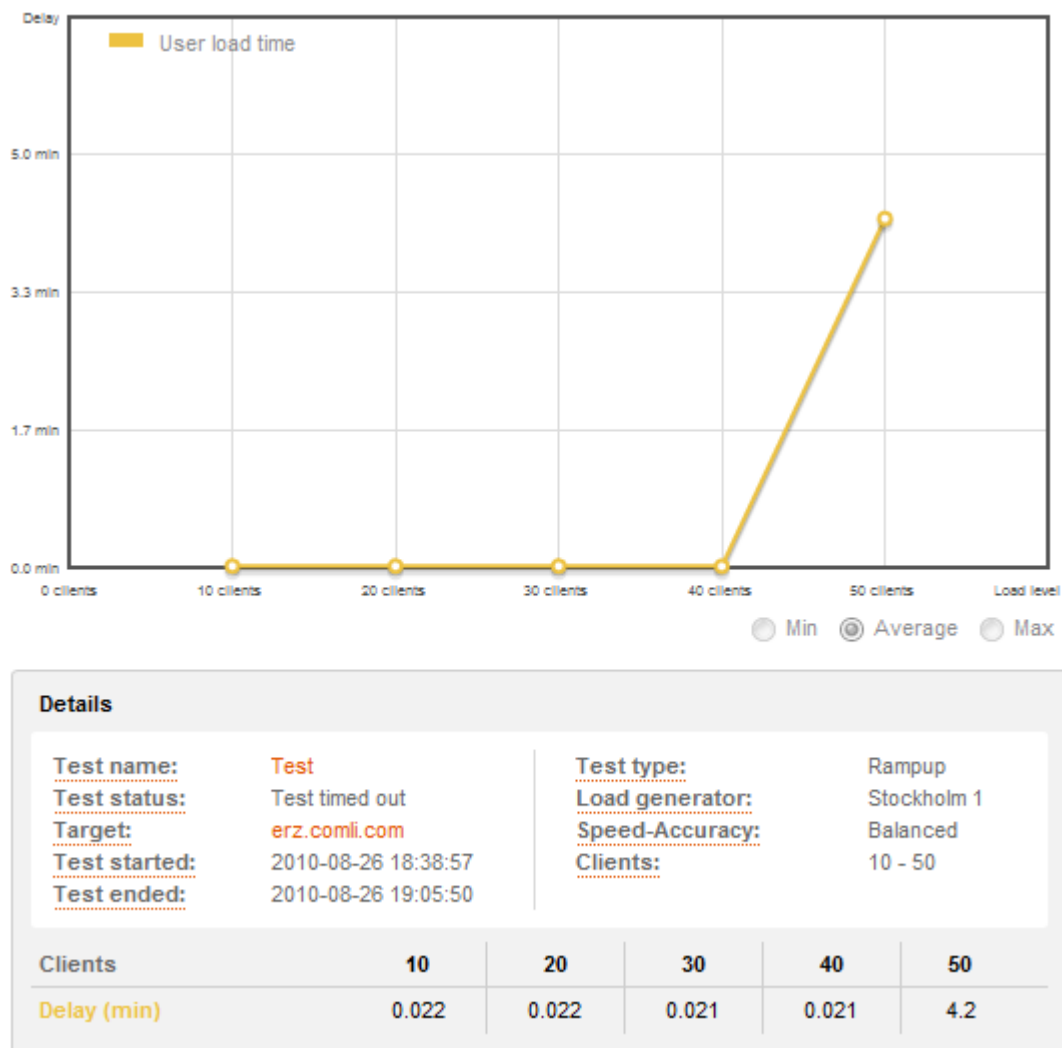
## Análisis y recomendaciones:

- TOTAL\_HTML (Aprobado): El número total de archivos HTML en la página incluyendo el index es 1 cuando la mayoría de navegadores utilizan hilos múltiples. Minimizando las peticiones HTTP es la clave de la optimización del sitio.

- TOTAL\_OBJECTS (Aprobado): El total de objetos en la página incluyendo el HTML es 4 que en la mayoría de navegadores que utilizan hilos múltiples en un tiempo razonable, la optimización se consigue minimizando la sobrecarga de objetos.
- TOTAL\_IMAGES (Aprobado): El número total de imágenes es 1, muchos navegadores pueden hacer peticiones múltiples para mostrar múltiples imágenes.
- TOTAL\_CSS (Aprobado): El número total de archivos CSS es 1, los archivos CSS se incrustan en el HEAD del documento, eso conlleva a mucha carga inicial, se aconseja cargar los archivos de JavaScript al final del documento para mostrar de manera progresiva el documento.
- TOTAL\_SIZE (Aprobado): El tamaño total de la página inicial es de 11,663 bytes, la página tuvo un tiempo de lectura de 3.12 segundos a una velocidad de 56kbps, a esa velocidad el tiempo ideal debe ser menor de 20 segundos con realimentación progresiva, idealmente una página debe leerse de 3 a 4 segundos en una conexión broadband.
- TOTAL\_SCRIPT (Aprobado): El número total de script externos es de 1, los script externos son agregados a memoria cache y se sugiere que los script sean recuperados al final del documento HTML.
- HTML\_SIZE (Aprobado): El tamaño total del HTML inicial es de 6,070 bytes, lo que es menor de 50K, esta medida debería ser el tamaño máximo de un archivo HTML.
- IMAGES\_SIZE (Aprobado): El tamaño total de las imágenes fue de 4380 bytes, similar al tamaño del HTML no debe sobrepasar los 50K para leerse en menos de 20 segundos en un ancho de banda de 56kbps, aunque lo ideal es que una imagen debería ser menor de 1,160 bytes para ajustarlo en un paquete TCP-IP.
- CSS\_SIZE (Aprobado): El tamaño total del CSS es 1,213 bytes, lo que es menor de 8K que es el límite máximo.
- MULTIM\_SIZE (Aprobado): El tamaño total de todos los archivos multimedia es de 0 bytes (no hay), lo que es menor del límite máximo de 10K.

Para la prueba de carga y estrés se utilizó la herramienta Load Impact [42], los resultados fueron los siguientes:

**CUADRO No. 5.5 Tiempo de carga de usuario**



En el cuadro anterior se muestra el tiempo de descarga de la página principal, sometiendo el sitio a descargas concurrentes de 10 hasta 50 clientes; el promedio de descarga desde 10 clientes hasta 40 fue de 1.32 segundos, siendo menor de 2 segundos que se considera normal; sin embargo, en cuanto a eficiencia el sitio no soporta más de 50 clientes concurrentes porque el tiempo de descarga se vuelve exponencial, aunque en [42] dice que es normal.

Aunque está medida no es fallida ya que se supone que aunque 50 clientes estén utilizando el sitio estos no van a hacer uso concurrentemente del sitio, para los fines de la herramienta se considera exitosa la prueba.

**CUADRO No. 5.6 Tiempo de carga acumulada**

Clients	10	20	30	40	50
Delay (s)	1.57	1.61	1.58	1.57	41.5

El cuadro anterior indica el tiempo de carga acumulada incluyendo otros objetos de descarga, el tiempo relativo a 1.32 segundos no cambió mucho de 10 a 40 clientes, sigue siendo estable alrededor de 1.57 segundos.

**CUADRO No. 5.7 Utilización de ancho de banda**

Clients	10	20	30	40	50
Bandwidth (kbit/s)	41	85.192	129.783	52.076	13.6

El cuadro anterior indica la utilización de ancho de banda utilizado con diferentes escalas de clientes, aunque los resultados se esperarían que fueran similares cada escala mostró diferente ancho de banda, este efecto es producido por los controles que ejerce el *hosting* cuando los clientes concurrentes son mayores de 40, es decir, aplica un límite de ancho de banda y esta es reducida.

**CUADRO No. 5.8 Comparación de tiempo de carga**

Clients	10	20	30	40	50
Delay (ms)	393	400	394	392	418
Delay (ms)	588	600	592	587	623

Comparación de tiempo de carga entre un archivo de texto y una imagen; las medidas del archivo de texto se indican en la primer línea y las de la imagen en la segunda. El promedio de descarga para un archivo de texto es de 0.4 segundos y para una imagen es de 0.6 segundos.

En general, el sitio<sup>3</sup> donde se encuentra alojada la herramienta pasa bien las pruebas de carga y estrés realizadas con la herramienta Load Impact.

---

<sup>3</sup> <http://erz.comli.com>



## 6. CASO DE ESTUDIO

Para validar la integración de los paradigmas de especificación de software, específicamente entre el modelo entidad-relación y el lenguaje Z, se elaboró un caso de estudio basado en el capítulo 11 de [43], relacionado con la especificación de una tienda de video.

El objetivo de este capítulo es, por una lado, mostrar al usuario como analizar un problema, como construir un modelo entidad-relación en la herramienta ERZ y como generar el equivalente en lenguaje Z; por otro lado, mostrar como extender la especificación generada para incluir invariantes que son imposibles colocar en el modelo entidad-relación, de modo que se aprovechen las ventajas de ambos paradigmas de especificación para describir de manera precisa el comportamiento de un sistema para garantizar su calidad.

### 6.1. Requerimiento

Se requiere especificar el comportamiento de una tienda de videos, en ella se tienen usuarios que pueden alquilar videos según la edad, por tanto, se necesita la fecha de nacimiento. Los videos son clasificados de acuerdo a categorías, por ejemplo, comedia o drama. Cada video tiene asociado una banda de precio dependiendo de la popularidad y que tan reciente se adquirió.

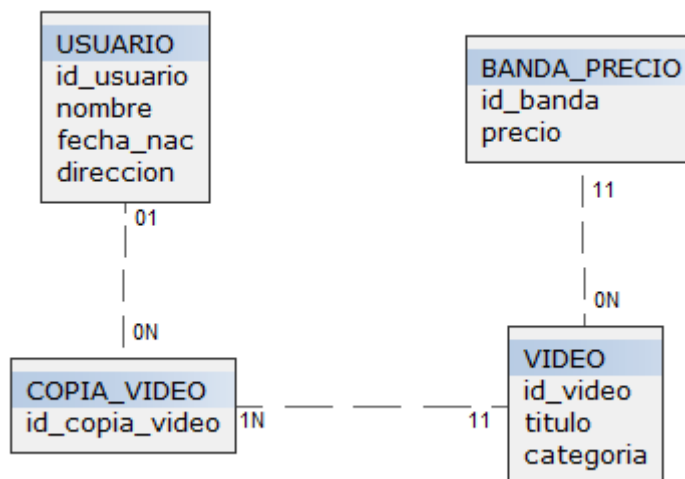
La especificación se debe realizar, por un lado, con modelado entidad-relación para describir la estructura de datos a utilizar, por otro lado, con en lenguaje Z para describir las operaciones; sin embargo, el comportamiento que no se puede especificar con un modelo entidad-relación se debe especificar de manera manual extendiendo la especificación generada automáticamente.

## 6.2. Modelo entidad-relación

Para elaborar el modelo entidad-relación se requiere la comprensión entre las entidades identificadas y la relación entre ellas, a continuación se muestran las diferentes asociaciones encontradas según el requerimiento:

- Cada usuario PUEDE alquilar UNA O MUCHAS copias de video
- Cada copia de video PUEDE ser alquilada por UN Y SOLO UN usuario.
- Un video DEBE tener asociado UNA O MUCHAS copias de video.
- Una copia de video DEBE corresponder a UN Y SOLO video.
- Una banda de precio PUEDE ser utilizada por UN O MUCHOS videos.
- Un video DEBE tener asociada UNA Y SOLO UNA banda de precio.

Con base en las asociaciones presentadas anteriormente se construyó el modelo entidad-relación tal como se muestra en la siguiente figura, posteriormente, se procederá a explicar su construcción en la herramienta ERZ.



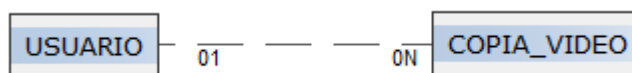
**FIGURA No. 6.1** Modelo entidad-relación de la tienda de video

La cardinalidad explícita de cada asociación es identificada conforme a la lectura del comportamiento de las asociaciones como se muestra a continuación.

Las cardinalidades de la asociación usuario-copia\_video se construye según las dos siguientes frases:

- Cada usuario PUEDE (0) alquilar UNA O MUCHAS (N) copias de video
- Cada copia de video PUEDE (0) ser alquilada por UN Y SOLO UN (1) usuario.

La cardinalidad explícita para el extremo de copia\_video es (0N) y la cardinalidad explícita para el extremo de usuario es (01); por tanto, la asociación usuario-copia\_video tiene la siguiente cardinalidad: 01..0N.

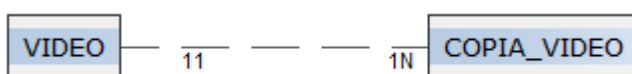


**FIGURA No. 6.2 Segmento usuario-copia\_video del ER**

Las cardinalidades de la asociación video-copia\_video se construye según las dos siguientes frases:

- Un video DEBE (1) tener asociado UNA O MUCHAS (N) copias de video.
- Una copia de video DEBE (1) corresponder a UN Y SOLO (1) video.

La cardinalidad explícita para el extremo de copia\_video es (1N) y la cardinalidad explícita para el extremo de video es (11); por tanto, la asociación video-copia\_video tiene la siguiente cardinalidad: 11..1N.



**FIGURA No. 6.3 Segmento video-copia\_video del ER**

Las cardinalidades de la asociación banda\_precio-video se construye según las dos siguientes frases:

- Una banda de precio PUEDE (0) ser utilizada por UN O MUCHOS (N) videos.
- Un video DEBE (1) tener asociada UNA Y SOLO UNA (1) banda de precio.

La cardinalidad explícita para el extremo de video es (0N) y la cardinalidad explícita para el extremo de banda\_precio es (11); por tanto, la asociación banda\_precio-video tiene la siguiente cardinalidad: 11..0N.

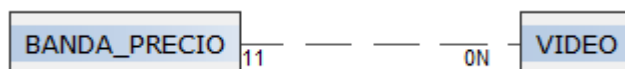


FIGURA No. 6.4 Segmento banda\_precio-video del ER

### 6.3. Elaboración del diagrama entidad-relación en la herramienta ERZ

Inicialmente se agregan las entidades por medio de la opción de “ADD ENTITY” de la siguiente figura:

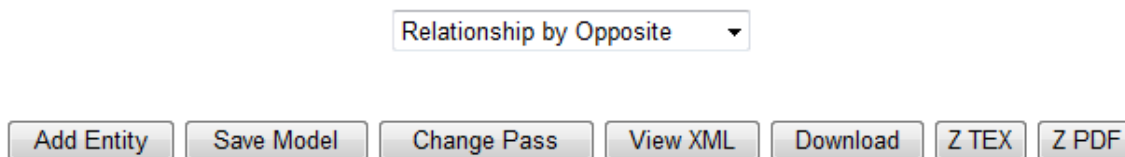
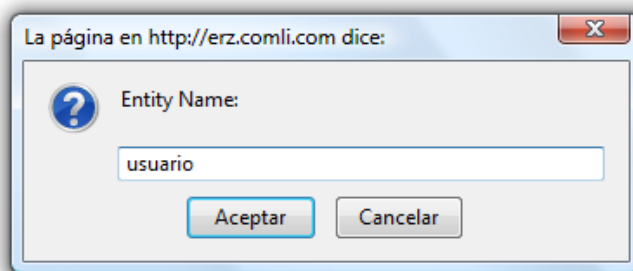


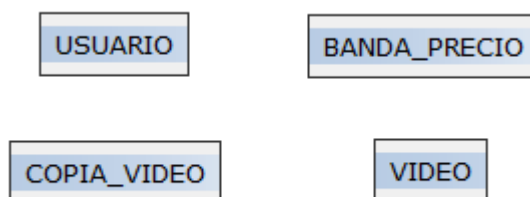
FIGURA No. 6.5 Opciones principales de la herramienta ERZ

Luego se escribe el nombre de la entidad como se muestra a continuación:



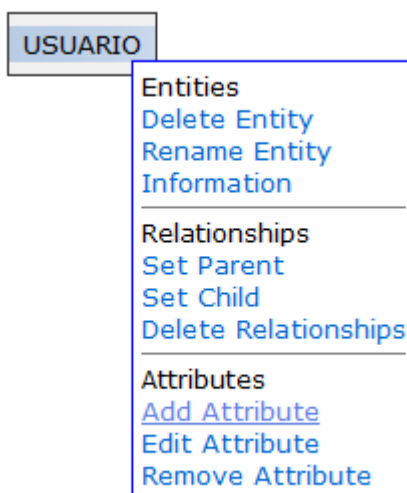
**FIGURA No. 6.6 Cuadro de diálogo para introducir el nombre de una entidad**

Para este caso de estudio solamente se identifican cuatro entidades:



**FIGURA No. 6.7 Vista temporal de las entidades agregadas**

Luego se agregan los atributos necesarios haciendo clic derecho encima de cada entidad y seleccionado la opción "ADD ATTRIBUTE":



**FIGURA No. 6.8 Menú contextual de una entidad**

Para cada atributo es necesario ingresar los siguientes datos:

Add Attribute:

Entity: usuario

Name:

Type: Char

Width & Scale: 12 0

Allow Nulls?

Identity?

Seed & Increment: 0 0

Primary Key?

Add Close

**FIGURA No. 6.9** Pantalla de ingreso de datos para una atributo

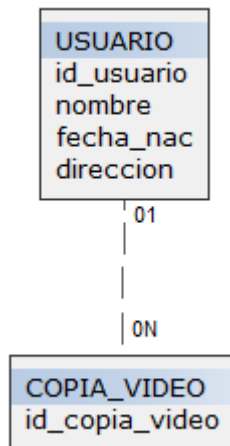
Luego de agregar todos los atributos y principalmente los identificadores se deben definir las asociaciones, debido a la dependencia de existencia y para facilitar la creación de las asociaciones se puede definir un rol a cada entidad, tal como padre e hijo, con las opciones del menú contextual “SET PARENT” y luego “SET CHILD”, seleccionando las cardinalidades adecuadas:

Select the cardinality

01  
0N  
11  
1N

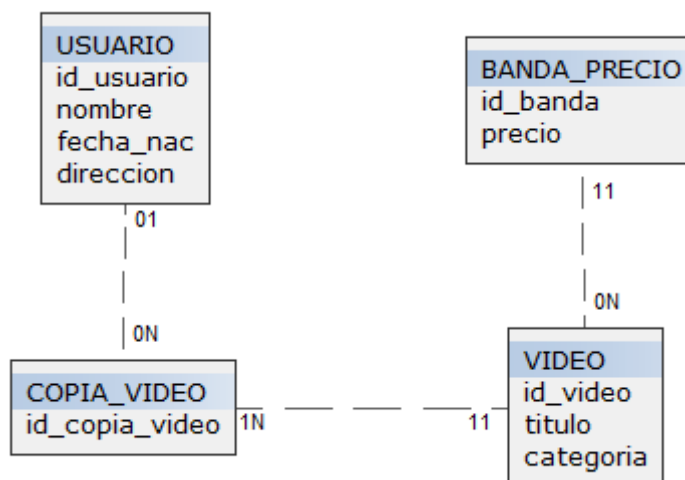
**FIGURA No. 6.10** Ingreso de la cardinalidad de la entidad padre

Al ingresar la cardinalidades de la entidad padre y de la entidad hija la asociación aparecerá uniéndose ambas entidades:



**FIGURA No. 6.11 Asociación ente usuario y copia de video**

Al completar todas las asociaciones se obtendrá el modelo antes descrito, en el cual, se describen las cuatro entidades y tres asociaciones entre ellas:



**FIGURA No. 6.12 Modelo entidad-relación de la tienda de video**

Para finalizar se debe almacenar el modelo, esto se hace con la opción “SAVE MODEL” del menú principal y mostrará un mensaje cuando el proceso haya terminado.

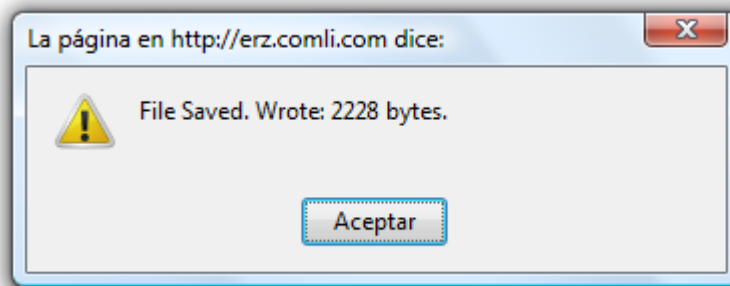


FIGURA No. 6.13 Cuadro de diálogo al almacenar un modelo en XML

## 6.4. Transformación e integración del modelo entidad-relación a lenguaje Z

En el menú principal de la herramienta hay dos botones que transforman el modelo entidad-relación a su equivalente especificación en Z, en dos formatos, uno es PDF y otro TEX. Como bien es sabido el PDF solo es para que el usuario pueda visualizar cómo quedaría la especificación formal y el TEX servirá para aumentar la especificación y que esta pueda ser verificada con el chequeador de tipos FUZZ.



FIGURA No. 6.14 Opciones de transformación de la herramienta ERZ

### 6.4.1. Generación de lenguaje Z en formato PDF

A continuación se muestran ciertos partes fundamentales de la especificación en formato PDF, la especificación completa se encuentra en el apéndice.



Tal como se mencionó en el capítulo de integración, una entidad es representada por el diccionario de datos, un esquema para la información de la entidad y otro esquema para describir la instancia que asocia un identificador con la información por medio de un par ordenado:

Entity: USUARIO

[ID\_USUARIO, NOMBRE, FECHA\_NAC, DIRECCION]

INFO\_USUARIO

nombre: NOMBRE

fecha\_nac: FECHA\_NAC

direccion: DIRECCION

INSTA\_USUARIO

id\_usuario:  $\mathbb{P}$  ID\_USUARIO

info\_usuario: ID\_USUARIO  $\leftrightarrow$  INFO\_USUARIO

id\_usuario = dom info\_usuario

En la descripción anterior se muestra únicamente la especificación de la entidad “usuario”. De la misma manera se procede con las demás entidades: video, copia\_video y banda\_precio.

Con respecto de las asociaciones, se especifica con un esquema donde utiliza el conjunto de identificadores de las entidades participantes, una función que puede ser parcial indicando una relación de uno a muchos y que asocia ambos identificadores indicando la cardinalidad de la asociación, y un invariante que define cual será el dominio y el rango de la función indicando la opcionalidad de la asociación.

Relationship: USUARIO-COPIA\_VIDEO

ASOCIA_USUARIO_COPIA_VIDEO
id_usuario: $\mathbb{P}$ ID_USUARIO
id_copia_video: $\mathbb{P}$ ID_COPIA_VIDEO
asocia_usuario_copia_video: ID_USUARIO $\leftrightarrow$ ID_COPIA_VIDEO
dom <i>asocia_usuario_copia_video</i> $\subseteq$ ID_USUARIO
ran <i>asocia_usuario_copia_video</i> $\subseteq$ ID_COPIA_VIDEO
<i>asocia_usuario_copia_video</i> $\sim \in$ ID_COPIA_VIDEO $\leftrightarrow$ ID_USUARIO

En la descripción anterior se muestra la relación que hay entre un usuario y una copia de video, se tienen los conjuntos de identificadores y una función parcial que representa la asociación de uno a muchos, y el invariante que restringe al dominio y el rango con el símbolo de subconjunto equivalente en ambos casos porque la opcionalidad se dio para ambas entidades.

En este caso de estudio existe una asociación que es opcional para ambas entidades, es decir, desde un punto de vista lógico la entidad que tiene la llave foránea del padre ocasionalmente puede tener valores nulos –cabe destacar que esto no aplica conceptualmente hablando–, la única manera de manejar esto en lenguaje Z [23] es por medio de una definición de tipo libre en Z, la cual se muestra continuación:

FreeType Definitions:

$$\text{NULL\_ID\_USUARIO} ::= \text{null\_id\_usuario} \mid \text{id\_usuario} \{ \{ \text{ID\_USUARIO} \} \}$$

El estado del sistema es representado por un esquema que contiene las instancias de las entidades así como las asociaciones existentes, luego hay que inicializar dicho estado:

**System State**

SYSTEM_STATE
INSTA_USUARIO
INSTA_COPIA_VIDEO
INSTA_VIDEO
INSTA_BANDA_PRECIO
ASOCIA_USUARIO_COPIA_VIDEO
ASOCIA_VIDEO_COPIA_VIDEO
ASOCIA_BANDA_PRECIO_VIDEO

INITIAL_SYSTEM_STATE
SYSTEM_STATE'
id_usuario' = ∅
id_copia_video' = ∅
id_video' = ∅
id_banda' = ∅
asocia_usuario_copia_video' = ∅
asocia_video_copia_video' = ∅
asocia_banda_precio_video' = ∅

Es importante mencionar que para un sistema debe existir un usuario administrador; debido a esto, habría que extender la especificación modificando el predicado de la siguiente manera:

$$id\_usuario' = \emptyset \quad \text{por} \quad id\_usuario' \neq \emptyset$$

Ya sea esta modificación u otra más precisa debe hacerse de manera manual, ya que el modelo entidad-relación no es capaz de describir este comportamiento y este es el punto fundamental de la integración de los paradigmas de especificación de software que se propone.

Para continuar especificando las operaciones es necesario describir un esquema común de operación, en el cual, se indique que esquemas del estado del sistema son afectadas por dicha operación; este esquema de operación común se debe describir para cada entidad, a continuación se muestra únicamente el esquema de operación de la entidad usuario:

USUARIO_OPERATION $\Delta$ SYSTEM_STATE $\exists$ INSTA_COPIA_VIDEO $\exists$ INSTA_VIDEO $\exists$ INSTA_BANDA_PRECIO $\exists$ ASOCIA_USUARIO_COPIA_VIDEO $\exists$ ASOCIA_VIDEO_COPIA_VIDEO $\exists$ ASOCIA_BANDA_PRECIO_VIDEO
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

En lo que respecta a las operaciones, estas son las que hacen al lenguaje Z una especificación más completa en comparación con el modelo entidad-relación. Se mostrará un esquema que define la operación agregar para la entidad usuario:

USUARIO_ADD_OK USUARIO_OPERATION id_usuario_in? : ID_USUARIO info_usuario_in? : INFO_USUARIO
$id\_usuario\_in? \notin id\_usuario$ $id\_usuario' = id\_usuario \sqcup \{id\_usuario\_in?\}$ $info\_usuario' = info\_usuario \sqcup \{id\_usuario\_in? \mapsto info\_usuario\_in?\}$

De manera similar existe un esquema para cada una de las operaciones básicas: eliminar, modificar y consultar.

USUARIO_DELETE_OK USUARIO_OPERATION id_usuario_in? : ID_USUARIO
id_usuario_in? $\in$ id_usuario id_usuario_in? $\notin$ dom asocia_usuario_copia_video info_usuario' = {id_usuario_in?} $\Leftarrow$ info_usuario id_usuario' = id_usuario \ {id_usuario_in?}

USUARIO_UPDATE_OK USUARIO_OPERATION id_usuario_in? : ID_USUARIO info_usuario_in? : INFO_USUARIO
id_usuario_in? $\in$ id_usuario info_usuario' = info_usuario $\oplus$ {id_usuario_in? $\mapsto$ info_usuario_in?}

USUARIO_QUERY_OK $\exists$ SYSTEM_STATE id_usuario_in? : ID_USUARIO info_usuario_out! : INFO_USUARIO
id_usuario_in? $\in$ id_usuario info_usuario_out! = info_usuario(id_usuario_in?)

Para complementar las operaciones se necesitan los esquemas que muestran los procesos exitosos o fallidos para luego definir las operaciones completas; primero se define un tipo libre de respuesta y el esquema de respuesta exitosa:

Response Messages:

RESPONSE ::= OK | EXISTS | NOT\_FOUND | HAS\_RELATIONSHIPS

SUCCESS\_OP

response! : RESPONSE

response! = OK

Luego se define un esquema común para que lo utilicen los esquemas de error que contiene el estado del sistema sin cambios, el identificador de entrada y la respuesta.

USUARIO\_ERROR

SYSTEM\_STATE

id\_usuario\_in? : ID\_USUARIO

response! : RESPONSE

Al definir este esquema se describen los de error que en este caso se definieron tres: cuando el usuario existe, cuando el usuario no existe y cuando el usuario participa en alguna asociación:

USUARIO\_EXISTS

USUARIO\_ERROR

id\_usuario\_in? ∈ id\_usuario

response! = EXISTS

USUARIO\_NOT\_FOUND

USUARIO\_ERROR

id\_usuario\_in? ∉ id\_usuario

response! = NOT\_FOUND

USUARIO\_HAS\_RELATIONSHIPS

USUARIO\_ERROR

(id\_usuario\_in? ∈ dom asocia\_usuario\_copia\_video ∧

response! = HAS\_RELATIONSHIPS)

Al crear los esquemas anteriores se pueden especificar operaciones completas que incorporen los errores que el sistema soporte; en el caso de agregar usuario puede ocurrir por un lado que se agrega correctamente o por otro lado el usuario ya existe y se debe mostrar un error; de la misma manera se definen las demás operaciones completas de eliminar, actualizar y consultar:

#### Total Operations

$$\text{USUARIO\_ADD} \triangleq (\text{USUARIO\_ADD\_OK} \wedge \text{SUCCESS\_OP})$$

$$\vee \text{USUARIO\_EXISTS}$$

$$\text{USUARIO\_DELETE} \triangleq (\text{USUARIO\_DELETE\_OK} \wedge \text{SUCCESS\_OP})$$

$$\vee \text{USUARIO\_NOT\_FOUND}$$

$$\vee \text{USUARIO\_HAS\_RELATIONSHIPS}$$

$$\text{USUARIO\_UPDATE} \triangleq (\text{USUARIO\_UPDATE\_OK} \wedge \text{SUCCESS\_OP})$$

$$\vee \text{USUARIO\_NOT\_FOUND}$$

$$\text{USUARIO\_QUERY} \triangleq (\text{USUARIO\_QUERY\_OK} \wedge \text{SUCCESS\_OP})$$

$$\vee \text{USUARIO\_NOT\_FOUND}$$

### 6.4.2. Generación de lenguaje Z en formato TEX

En la herramienta toda especificación también puede generarse en formato TEX; ya se ha mencionado que la idea de soportar este formato es ampliar la especificación, de tal manera, que puedan agregarse invariantes que el modelo entidad-relación no puede representar; por ejemplo, en la tienda de video existen ciertas películas que no pueden alquilarse a cualquier usuario si no cumple con requisitos de edad, o que a una persona solo se le puede alquilar un máximo de 3 copias, etc. Todos estos invariantes se deben agregar de manera manual en el formato TEX.

Un fragmento del formato TEX se muestra continuación:

```

\documentstyle[fuzz]{article}
\begin{document}
Z Notation
Generated by ERZ
Entities and attributes
Entity: USUARIO
\begin{zed}
[ID\_USUARIO, NOMBRE, FECHA\_NAC, DIRECCION]
\end{zed}
\begin{schema}{INFO\_USUARIO}
nombre: NOMBRE\\
fecha\_nac: FECHA\_NAC\\
direccion: DIRECCION
\end{schema}
\begin{schema}{INSTA\_USUARIO}
id\_usuario: \power ID\_USUARIO \\
info\_usuario: ID\_USUARIO \pfun INFO\_USUARIO
\where
id\_usuario = \dom info\_usuario
\end{schema}
Entity: COPIA\_VIDEO
\begin{zed}
[ID\_COPIA\_VIDEO] \also
NULL\_COPIA\_VIDEO ::= no\_information
\end{zed} \begin{schema}{INFO\_COPIA\_VIDEO}
att\_copia\_video: NULL\_COPIA\_VIDEO
\end{schema}
\begin{schema}{INSTA\_COPIA\_VIDEO}
id\_copia\_video: \power ID\_COPIA\_VIDEO \\
info\_copia\_video: ID\_COPIA\_VIDEO \pfun INFO\_COPIA\_VIDEO
\where
id\_copia\_video = \dom info\_copia\_video
\end{schema}

```

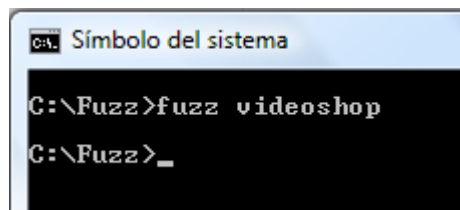
### 6.4.3. Verificación de tipos y extensión de la especificación

Teniendo la especificación en formato Latex hay dos cosas importantes de mencionar: por un lado, esta especificación puede ser verificada en FUZZ para comprobar que está formada correctamente; y por otro lado, esta especificación puede ser extendida para ampliar la especificación en relación con invariantes que no pueden definirse en un modelo entidad-relación. Y esta es la importancia de la integración de paradigmas de especificación de software.



### 6.4.3.1. Verificación de tipos con FUZZ

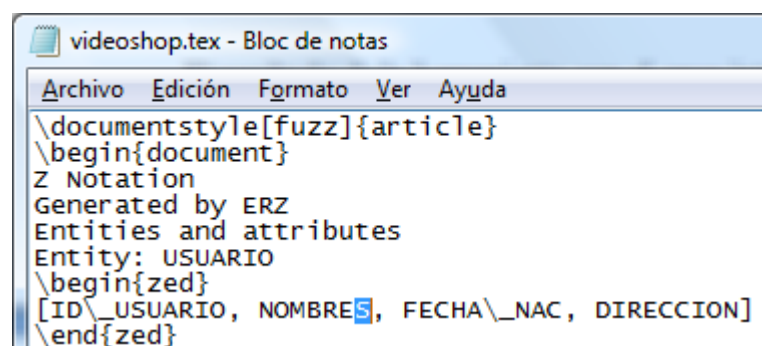
La finalidad de la generación del lenguaje Z en formato TEX es verificar los tipos de la especificación por medio de la herramienta de software FUZZ [29], se procedió a instalar la herramienta y desde la línea de comandos se ejecutó -utilizando el archivo TEX generado- como se ilustra a continuación:



```
C:\Fuzz>fuzz videoshop
C:\Fuzz>
```

FIGURA No. 6.15 Verificación del caso de estudio con FUZZ

FUZZ se basa en mostrar los errores; si el archivo no tiene ningún error el resultado es que no muestra ningún mensaje. Se editó el archivo para introducir un error para mostrar al lector el funcionamiento de la herramienta; el error consistió en agregar una letra “S” al conjunto NOMBRE como se muestra a continuación:



```
videoshop.tex - Bloc de notas
Archivo Edición Formato Ver Ayuda
\documentstyle[fuzz]{article}
\begin{document}
Z Notation
Generated by ERZ
Entities and attributes
Entity: USUARIO
\begin{zed}
[ID\_USUARIO, NOMBRE S, FECHA\_NAC, DIRECCION]
\end{zed}
```

FIGURA No. 6.16 Inserción de un error en la especificación

El resultado de la herramienta con el error introducido se muestra a continuación:

```

C:\Fuzz>fuzz videoshop
"videoshop.tex", line 12: Identifier NOMBRE is not declared

C:\Fuzz>_

```

FIGURA No. 6.1 Resultado de la verificación con archivo inválido

#### 6.4.3.2. Extensión de la descripción en formato TEX

La especificación generada automáticamente se basa exclusivamente en el modelo entidad-relación. Cualquier extensión que se deba realizar se debe realizar manualmente sobre el archivo TEX.

Para explicar el proceso se describirá la restricción llamada “PUEDE\_ALQUILAR” que aumentará el esquema del estado del sistema. Esta restricción es la encargada de verificar y censurar la edad del usuario; se procede a calcular la edad del usuario y verificar la censura para determinar si se puede o no alquilar la copia. El alcance de esta demostración es la operación para alquilar. Para lograr esto se deben agregar y modificar varios esquemas de la especificación generada automáticamente.

Primero, hay que agregar todos los tipos necesarios para que la descripción sea correcta. Se necesita de las restricciones, la edad y la edad calculada. Las declaraciones se muestran a continuación:

$$\begin{aligned}
 \text{RESTRICCION} ::= & \text{libre} \mid \text{infantil} \mid \text{mayores\_12\_anos} \mid \text{acompanado\_padres} \\
 & \mid \text{mayores\_15\_anos} \mid \text{mayores\_18\_anos}
 \end{aligned}$$

$$\text{EDAD} == \mathbb{N}$$

$$\mid \text{edad\_hoy} : \text{FECHA} \times \text{FECHA} \rightarrow \text{EDAD}$$

Segundo, hay que modificar por un lado el tipo de “fecha\_nac” del esquema “INFO\_USUARIO” para que quede del mismo tipo FECHA y por otro lado el esquema “INFO\_VIDEO” para que contenga la restricción de las edades:

```
INFO_VIDEO
-----
titulo: TITULO
categoria: CATEGORIA
restriccion: RESTRICCION
```

Tercero, hay que crear el esquema “PUEDE\_ALQUILAR” que maneja las diferentes restricciones en su invariante; lo que se pretende es verificar la edad y el certificado del video:

```
PUEDE_ALQUILAR
-----
INSTA_USUARIO
INSTA_COPIA_VIDEO
INSTA_VIDEO
INSTA_BANDA_PRECIO
ASOCIA_USUARIO_COPIA_VIDEO
ASOCIA_VIDEO_COPIA_VIDEO
ASOCIA_BANDA_PRECIO_VIDEO
hoy :FECHA
puede_alquilar : ID_USUARIO ↔ ID_VIDEO
-----
(puede_alquilar) =
{ u: id_usuario; v: ran asocia_video_copia_video
/ let v_restriccion == (info_video(v)).restriccion ;
  e == edad_hoy (hoy,(info_usuario(u)).fecha_nac)
  •
  (e ≥ 18
  ∨ v_restriccion ∈ {libre, infantil, acompañado_padres }
  ∨ (e ≥ 15 ∧ v_restriccion = mayores_15_anos)
  ∨ (e ≥ 12 ∧ v_restriccion = mayores_12_anos))
}
```

Cuarto, hay que modificar el invariante del estado del sistema para incluir el esquema de “PUEDE\_ALQUILAR”; además expresa la restricción de que todas las copias en préstamo han sido prestadas a usuarios con edad suficiente:

Para esto se modifica el esquema inicial generado del estado del sistema y se utiliza el esquema “puede\_alquilar”, este esquema ya tiene los esquemas que inicialmente tenía el estado del sistema, por tanto, no es necesario escribir duplicada esa información y la declaración del esquema solo quedaría el esquema “puede\_alquilar”.

SYSTEM\_STATE

PUEDE\_ALQUILAR

$\forall c: \text{dom } \text{asocia\_usuario\_copia\_video} \bullet (\text{asocia\_usuario\_copia\_video } c )$   
 $\text{puede\_alquilar } (\text{asocia\_video\_copia\_video } c )$

Quinto, hay que agregar al estado inicial del sistema la declaración de la fecha e inicializar cada conjunto con el símbolo “ $\emptyset$ ” de conjunto vacío:

INITIAL\_SYSTEM\_STATE

SYSTEM\_STATE' ; hoy? : FECHA

$\text{id\_usuario}' = \emptyset$

$\text{id\_copia\_video}' = \emptyset$

$\text{id\_video}' = \emptyset$

$\text{id\_banda}' = \emptyset$

$\text{asocia\_usuario\_copia\_video}' = \emptyset$

$\text{asocia\_video\_copia\_video}' = \emptyset$

$\text{asocia\_banda\_precio\_video}' = \emptyset$

$\text{hoy}' = \text{hoy?}$

$\text{puede\_alquilar}' = \emptyset$

Y sexto, por último hay que crear la operación alquilar copia. En esta operación se asocia una copia de video con un usuario; cabe mencionar que no se tomó en cuenta ni el precio ni el mensaje de respuesta ya que únicamente se quiere demostrar la funcionalidad de la extensión de la especificación.

*ALQUILAR\_COPIA*

*COPIA\_VIDEO\_OPERATION*

$\exists$  *INSTA\_COPIA\_VIDEO*

$\exists$  *ASOCIA\_VIDEO\_COPIA\_VIDEO*

*id\_usuario\_in?* : *ID\_USUARIO*

*id\_copia\_video\_in?* : *ID\_COPIA\_VIDEO*

$id\_usuario\_in? \in id\_usuario$

$id\_copia\_video\_in? \in id\_copia\_video$

$id\_copia\_video\_in? \in \text{ran } asocia\_video\_copia\_video$

$id\_copia\_video\_in? \notin \text{ran } asocia\_usuario\_copia\_video$

$id\_usuario\_in? puede\_alquilar (asocia\_video\_copia\_video \ id\_copia\_video\_in?)$

$asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video \cup$

$\{ id\_usuario\_in? \mapsto id\_copia\_video\_in? \}$

Toda la explicación anterior de la extensión que se realizó se reescribirá en formato TEX; este nuevo segmento se debe agregar a la especificación generada y luego verificarla de nuevo con FUZZ.

A continuación se muestran las definiciones y esquemas básicos:

```
\begin{zed}
RESTRICCION ::= libre | infantil | mayores\_12\_anos \\
| acompañado\_padres | mayores\_15\_anos | mayores\_18\_anos
\also EDAD == \nat
\end{zed}

\begin{axdef}
edad_hoy : FECHA \cross FECHA \fun EDAD
\end{axdef}
```

```

\begin{schema}{INFO\_VIDEO}
titulo: TITULO \\\
categoria: CATEGORIA \\\
restriccion: RESTRICCION
\end{schema}

```

Luego se agrega el esquema “PUEDE\_ALQUILAR”; siendo este el esquema a agregar en la especificación:

```

\begin{schema}{PUEDE\_ALQUILAR}
INSTA\_USUARIO \\\
INSTA\_COPIA\_VIDEO \\\
INSTA\_VIDEO \\\
INSTA\_BANDA\_PRECIO \\\
ASOCIA\_USUARIO\_COPIA\_VIDEO \\\
ASOCIA\_VIDEO\_COPIA\_VIDEO \\\
ASOCIA\_BANDA\_PRECIO\_VIDEO \\\
hoy: FECHA \\\
puede\_alquilar: ID\_USUARIO \rel ID\_VIDEO
\where
(puede\_alquilar) = \{ u: id\_usuario; \\\
v: \ran asocia\_video\_copia\_video \\\
| \LET v\_restriccion == (info\_video(v)).restriccion; \\\
e == edad\_hoy (hoy, (info\_usuario(u)).fecha\_nac) @ \\\
(e \geq 18 \lor v\_restriccion \in \\\
\{libre, infantil, acompanado\_padres\}) \\\
\lor (e \geq 15 \land v\_restriccion = mayores\_15\_anos) \\\
\lor (e \geq 12 \land v\_restriccion = mayores\_12\_anos)) \} \\\
\end{schema}

```

Por último se agrega el estado del sistema y la operación alquilar para probar la funcionalidad de la extensión a la especificación:

```

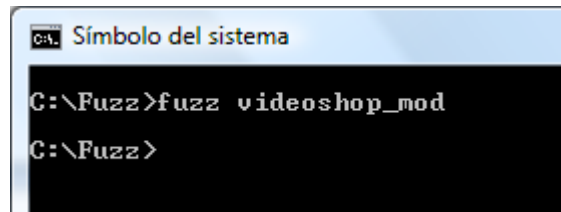
\begin{schema}{SYSTEM\_STATE}
PUEDE\_ALQUILAR
\where
\forall c: \dom asocia\_usuario\_copia\_video @ \
(asocia\_usuario\_copia\_video(c)) \inrel{puede\_alquilar} \
(asocia\_video\_copia\_video(c))
\end{schema}

\begin{schema}{INITIAL\_SYSTEM\_STATE}
SYSTEM\_STATE' \ hoy? : FECHA
\where
id\_usuario' = \emptyset \ id\_copia\_video' = \emptyset \
id\_video' = \emptyset \ id\_banda' = \emptyset \
asocia\_usuario\_copia\_video' = \emptyset \
asocia\_video\_copia\_video' = \emptyset \
asocia\_banda\_precio\_video' = \emptyset \ hoy' = hoy? \
puede\_alquilar' = \emptyset
\end{schema}

\begin{schema}{ALQUILAR\_COPIA}
COPIA\_VIDEO\_OPERATION \
\Xi INSTA\_COPIA\_VIDEO \ \Xi ASOCIA\_VIDEO\_COPIA\_VIDEO \
id\_usuario\_in?: ID\_USUARIO \
id\_copia\_video\_in?: ID\_COPIA\_VIDEO
\where
id\_usuario\_in? \in id\_usuario \
id\_copia\_video\_in? \in id\_copia\_video \
id\_copia\_video\_in? \in \dom asocia\_video\_copia\_video \
id\_copia\_video\_in? \notin \dom asocia\_usuario\_copia\_video \
(id\_usuario\_in?) \inrel{puede\_alquilar} \
(asocia\_video\_copia\_video(id\_copia\_video\_in?)) \
asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video \
\cup \{id\_copia\_video\_in? \mapsto id\_usuario\_in? \}
\end{schema}

```

El nuevo archivo “videoshop\_mod.tex” con los cambios agregados se verificó con FUZZ. La terminología sin mensajes de error corresponde a finalización exitosa y se muestra a continuación el resultado:



```
ca. Símbolo del sistema
C:\Fuzz>fuzz videoshop_mod
C:\Fuzz>
```

**FIGURA No. 6.17** Resultado de la ejecución de FUZZ con videoshop\_mod

En resumen, se demostró cómo agregar invariantes a la especificación que en el modelo entidad-relación no pueden diagramarse ni generarse automáticamente. Esto muestra el nivel de integración al que puede llegarse con paradigmas de especificación de software tan diferentes como los diagramáticos y los formales, y cómo esta integración coadyuva a describir de mejor manera el comportamiento del software para que en un futuro sea construido sin ambigüedad y de forma precisa para garantizar la calidad del software.



## 7. ANÁLISIS DE RESULTADOS

En esta sección se discutirán ciertos aspectos relacionados con el cumplimiento de los objetivos, lo relacionado con el desarrollo de la herramienta de software y, en general, con los resultados de la investigación.

### 7.1. Discusión

Es importante destacar la integración de paradigmas de especificación como una manera de unir dos especificaciones para aumentar las ventajas y aminorar las desventajas de tal forma que se produzca un software de alta calidad.

Tal como se estableció en los alcances y limitaciones de la investigación la herramienta de software está a un nivel de prototipo funcional, cumpliendo con el objetivo de producir como salida un archivo en formato PDF y/o TEX con el lenguaje Z equivalente a la especificación basada en un modelo entidad-relación.

Es importante destacar que la herramienta como está alojada en un sitio Web se puso a pruebas por parte de usuarios externos, mucha información se realimentó de estas pruebas para mejorar el producto. Se agradece a todas las personas que probaron la herramienta.

### 7.2. Dificultades encontradas

JavaScript es un lenguaje de programación que de facto se ha vuelto el principal lenguaje entre navegadores por su simplicidad. Sin embargo, esta simplicidad no significa que el lenguaje no sea robusto; con JavaScript se pueden hacer muchas cosas que se hacen con cualquier lenguaje de programación.

Pero no todo es tan sencillo, hay que notar que graficar dentro de un navegador hace muy complicada su programación porque los navegadores cumplen con ciertos estándares y, como es bien sabido, hay ciertos aspectos que para Internet Explorer son diferentes que para Firefox.

La parte de dibujar las entidades se solucionó con el tutorial mencionado anteriormente. El mayor problema era dibujar las asociaciones entre esas entidades. Se probaron varias bibliotecas pero algunas tenían problemas de rendimiento a la hora de redibujar las conexiones. Al final se solucionó por medio de la biblioteca jsDraw2D, esta biblioteca es muy rápida y satisface las necesidades requeridas para dibujar.

Otro problema encontrado era cómo hacer la conexión entre el lenguaje JavaScript y PHP, uno de lado del cliente y otro del lado del servidor, esta conexión conlleva dos dificultades: la conexión en sí y la transferencia de los datos; la más común es enviar los datos por medio de un formulario a PHP, pero la herramienta trabaja en una sola página para evitar las recargas innecesarias; la conexión se realizó por medio de AJAX para mandar a ejecutar un PHP que pueda realizar las operaciones necesarias como almacenamiento y transformación, y los datos como estaban en estructuras de datos tipo arreglo fueron enviados a partir de una serialización del arreglo y en PHP solamente se hace la conversión para su almacenamiento y para su transformación se utiliza el archivo XML directamente por medio del DOMDocument de PHP.

### **7.3. Nivel de satisfacción de objetivos**

En general la herramienta realiza el trabajo requerido. La idea fundamental es crear un modelo entidad-relación, almacenarlo y luego transformarlo a su equivalente en lenguaje Z. Para mayor comodidad del usuario se utilizaron los dos formatos mencionados de PDF y TEX.

Como limitaciones de la herramienta aún se encuentra la posibilidad de extender la transformación para soportar un conjunto de identificadores, así como ciertos aspectos de usabilidad asociados a la herramienta. En general, la herramienta tiene un aspecto simplista ya que el objetivo era producir un archivo de especificación que pueda ser extendido posteriormente.

Otra limitación de la herramienta es que para cada par de entidades solamente se puede crear una asociación en una sola vía; el nombre de la asociación se hace automáticamente y estandarizada de la siguiente manera: `asocia_entidad1_entidad2`, por tanto no puede haber otra asociación con el mismo nombre.

#### **7.4. Hallazgos encontrados**

Se realizó un estudio de lenguajes y plataformas de programación, las más idóneas para este tipo de herramienta Web fueron JavaScript del lado del cliente y PHP del lado del servidor. Cabe destacar que JavaScript tiene un enorme potencial en cuanto a programación Web y se considera que muchos proyectos puedan ser realizados en este lenguaje, instando a los lectores a que conozcan mejor el lenguaje y las características que provee.

En cuanto al marco de referencia se realizaron varias modificaciones que fueron elaboradas durante el desarrollo, tal como el caso de los nulos en llaves foráneas, además se agregó la funcionalidad de modelar asociaciones de muchos a muchos para no limitar la herramienta a usuarios que conceptualmente manejan este tipo de asociaciones.

También se encontró que es útil manejar los dos tipos de notación existentes: la asociación por participación y la asociación con referencia al opuesto; la primera utilizada en la notación de Peter Chen y la segunda utilizada por varias notaciones donde destaca la notación de Barker. Debido a lo anterior, la herramienta soporta ambos tipos de notaciones en lo que respecta a las cardinalidades de las asociaciones y no tanto al dibujo de la asociación.

## **8. CONCLUSIONES Y RECOMENDACIONES**

### **8.1. Conclusiones**

Se realizó una investigación documental de los paradigmas de especificación de software, asimismo esto sirvió para examinar las características de otras herramientas relacionadas a la integración de paradigmas que se muestran en el capítulo 2.

Con base en la teoría investigada y la experiencia adquirida por el autor se desarrolló un marco de referencia entre el modelo entidad-relación y el lenguaje Z para integrar de manera efectiva ambos paradigmas de especificación, lo que constituye el valor científico agregado a la investigación.

Para comprobar dicho marco de referencia se desarrolló una herramienta Web que diagrama modelos entidad-relación y es capaz de transformar dicho modelo a su equivalente en lenguaje Z, integrando de esta manera ambos paradigmas y posibilitando aumentar la calidad de la solución.

Como ejemplo se elaboró un caso de estudio de una tienda de video para mostrar las cualidades del marco de referencia por medio de la herramienta desarrollada que representa su dominio de aplicación, y con base en todo lo anterior llegar a cumplir los objetivos planteados.

### **8.2. Recomendaciones**

A lo largo del trabajo se fueron encontrando ciertas características que pueden ser desarrolladas como trabajo futuro para continuar esta línea de investigación.

Entre algunas mejoras están: la seguridad en cuanto a los usuarios que acceden a la herramienta; soportar otros tipos de notaciones gráficas; soportar un conjunto de identificadores; permitir la creación de asociaciones n-arias no solamente binarias; manejar los nombres de asociaciones de manera personalizada para no restringir a una sola asociación por cada par de entidades; establecer invariantes y proponer ciertas transacciones que pueden ser especificadas de manera automática y considerar operaciones complejas con al menos dos entidades y una o más asociaciones.

Se recomienda a la escuela de postgrado del TEC incentivar a sus estudiantes a utilizar dicha herramienta y de esta manera encontrar más temas de investigación relacionados con la extensión de este trabajo. Se considera que la integración entre paradigmas de especificación de software es importante para destacar las ventajas que cada uno posee y no solo guiarse por lo simple o complejo que es cada paradigma de especificación de software por separado.

## 9. REFERENCIAS BIBLIOGRÁFICAS

- [1] J. M. Spivey, *The Z Notation: A Reference Manual*, 2nd ed. England: Oriel College, Oxford., 1998.
- [2] I. Vessey, "Cognitive fit: A theory-based analysis of graphs vs. tables literature," in *Decision Sci.*, vol. 22, 1991, p. 219–240.
- [3] I. Vessey and D. Galletta, "Cognitive Fit: An Empirical Study of Information Acquisition," in *Information Systems Research*, vol. 2, 1991, pp. 63-84.
- [4] L. Espino, "Notación ESB: Modelado Semántico de Datos basado en Diagramas Entidad-Relación Normalizados," Instituto Tecnológico de Costa Rica, 2009.
- [5] C. J. Date, *Introducción a los Sistemas de Bases de Datos*, 5th ed. México: Addison Wesley Iberoamericana, 1992.
- [6] C. Bachman, "Summary of current work ANSI/X3/SPARC/study group: database systems," in *ACM SIGMOD Record*, vol. 6, United States of America, 1974, pp. 16-39.
- [7] T. Halpin and T. Morgan, *Information Modeling and Relational Databases*, 2nd ed. United States of America: Elsevier Inc., 2008.
- [8] P. Chen, "The Entity-Relationship Model - Toward a Unified View of Data," in *ACM Transactions on Database Systems*, vol. 1. No.1, 1976, pp. 9-36.
- [9] R. Barker, *CASE\*METHOD Entity Relationship Modelling*. UK: Addison-Wesley, 1990.
- [10] E. Codd, *The Relational Model for Database Management: version 2*. USA: Addison-Wesley Publishing Company, Inc., 1990.
- [11] E. Codd, "Extending the Database Relational model to Capture More Meaning," in *ACM Transaction on Database Systems*, vol. 4, No. 4, California, USA, 1979, pp. 397-434.
- [12] T. Halpin and T. Morgan, *Information Modeling and Relational Databases*, 2nd ed. United States of America: Elsevier Inc., 2008.

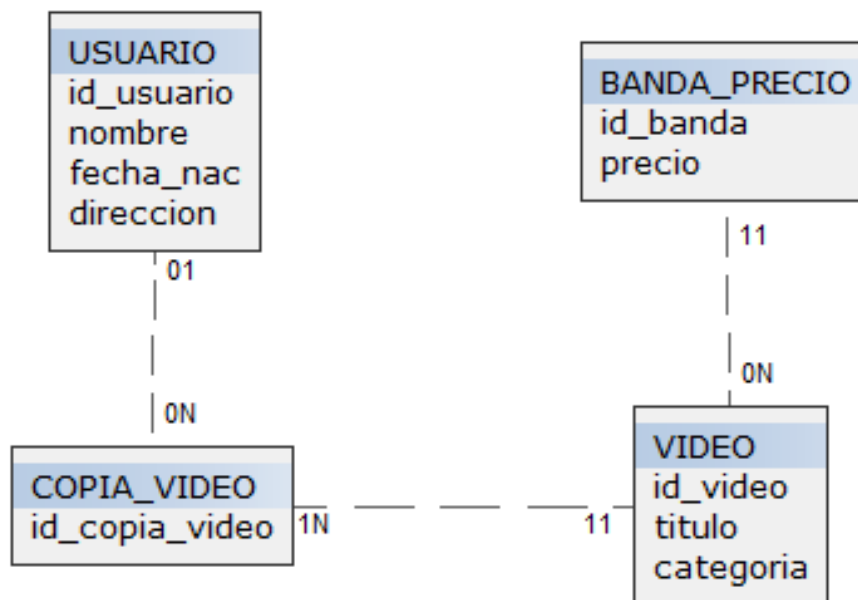
- [13] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Fundamentos de Bases de Datos*, 4th ed. España: McGraw-Hill Inc., 2002.
- [14] R. Elmasri and S. Navathe, *Sistemas de Bases de datos. Conceptos Fundamentales*, 2nd ed. México: Addison-Wesley Iberoamericana, 1994.
- [15] E. Harold, *XML 1.1 Bible*, 3rd ed. United States of America: Wiley Publishing, Inc., 2004.
- [16] A. Silberschatz, H. Korth, and S. Sudarshan, *Fundamentos de Bases de Datos*, 5th ed. España: McGraw-Hill, 2006.
- [17] L. Espino, "Definición de Esquemas Relacionales en XML," Instituto Tecnológico de Costa Rica, 2009.
- [18] I. Sommerville, *Ingeniería del software*. Madrid, España: Pearson Educación, S.A., 2005.
- [19] R. S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*. España: McGraw-Hill, 2005.
- [20] H. Gabbar, *Modern Formal Methods and Applications*. Netherlands: Springer, 2006.
- [21] H. Habrias and M. Frappier, *Software Specification Methods*. Great Britain: ISTE, Ltd., 2006.
- [22] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. United States of America: MIT Press, 2006.
- [23] J. Woodcock and J. Davies, *Using Z: Specification, Refinement, and Proof*. Englad: University of Oxford.
- [24] L. E. Galvao Martins, *An Empirical Study Using Z and UML for the Requirements Specification of an Information System*. Brasil: UNIMEP, 2002.
- [25] J. Cordero Rios, "Bases para la integración de modelos estructurados y formales en la especificación de sistemas de información," Instituto Tecnológico de Costa Rica Tesis (Magister Scientiae en Computación), 1996.
- [26] I. Van Coppenhagen, *From Specifications in Z and UML through Refinement to Implementation in Java: A Comparative Study on the use of Relationships*. South Africa: UNISA, 2004.

- [27] S. Dupuy, *RoZ version 0.3 an Environment for the Integration of UML and Z*. France: IMAG, 2007.
- [28] IEEE Computer Society, "IEEE Recommended Practice for Software Requirements Specifications," IEEE IEEE-SA Standard Board 830-1998, 1998.
- [29] J. M. Spivey. (2010, Aug.) The fuzz type-checker for Z. [Online]. <http://spivey.oriel.ox.ac.uk/mike/fuzz/>
- [30] J. M. Spivey, *The fuzz Manual*, 2nd ed. Oxford, England, 2000.
- [31] ISO/IEC, "Software engineering - Product quality - Part 1: Quality model," ISO/IEC ISO/IEC 9126-1:2001(E), 2001.
- [32] J. McCall, P. Richards, and G. Walters, "Factors in Software Quality," General Electric Command & Information Systems, California, USA., Tech. Report 77CIS02, 1977.
- [33] C. Campbell, A. Eisenberg, and J. Melton, "XML Schema," in *SIGMOD Record*, Vol. 32, No. 2, 2003, pp. 86-101.
- [34] L. Breuer. (2006) Interactive Tutorials. [Online]. <http://luke.breuer.com/tutorial/>
- [35] jsFiction. (2009) jsDraw2D. [Online]. <http://jsdraw2d.jsfiction.com/>
- [36] O. Plathey. (August, ) FPDF Library - PDF generator. [Online]. <http://www.fpdf.org/>
- [37] W3C. (2010, Aug.) Markup Validation Service. [Online]. <http://validator.w3.org/>
- [38] W3C. (2010, Aug.) CSS Validation Service. [Online]. <http://jigsaw.w3.org/css-validator/>
- [39] Scanit. (2010, Aug.) Scanit. [Online]. <http://www.scanit.be/>
- [40] U. Parasites. (2010, Aug.) Unmask Parasites. [Online]. <http://www.unmaskparasites.com/>
- [41] WebSiteOptimization. (2010, Aug.) WebSite Optimization. [Online]. <http://www.websiteoptimization.com/>
- [42] LoadImpact. (2010, Aug.) Load Impact. [Online]. <http://loadimpact.com>
- [43] R. Barden, S. Stepney, and D. Cooper, *Z in Practice*. USA: Prentice-Hall, Inc., 1994.
- [44] C. Schenk. (2010, Aug.) MiKTeX. [Online]. <http://miktex.org/>



## 10. APÉNDICE: TIENDA DE VIDEO

### 10.1. Modelo entidad-relación de la tienda de video



### 10.2. Lenguaje Z en PDF generado a partir del modelo entidad-relación

A continuación se adjunta el PDF generado por la herramienta de software desarrollada, basada en la integración efectiva de paradigmas de especificación de software, específicamente entre el modelo entidad-relación y el lenguaje Z.

## Z Notation

Generated by ERZ

### Entities and attributes

Entity: USUARIO

[ID\_USUARIO, NOMBRE, FECHA\_NAC, DIRECCION]

INFO\_USUARIO

nombre: NOMBRE

fecha\_nac: FECHA\_NAC

direccion: DIRECCION

INSTA\_USUARIO

id\_usuario:  $\mathbb{P}$  ID\_USUARIO

info\_usuario: ID\_USUARIO  $\leftrightarrow$  INFO\_USUARIO

id\_usuario = dom info\_usuario

Entity: COPIA\_VIDEO

[ID\_COPIA\_VIDEO]

NULL\_COPIA\_VIDEO ::= no\_information\_copia\_video

INFO\_COPIA\_VIDEO

att\_copia\_video: NULL\_COPIA\_VIDEO

INSTA\_COPIA\_VIDEO

id\_copia\_video:  $\mathbb{P}$  ID\_COPIA\_VIDEO

info\_copia\_video: ID\_COPIA\_VIDEO  $\leftrightarrow$  INFO\_COPIA\_VIDEO

id\_copia\_video = dom info\_copia\_video

Entity: VIDEO

[ID\_VIDEO, TITULO, CATEGORIA]

INFO_VIDEO
titulo: TITULO
categoria: CATEGORIA

INSTA_VIDEO
id_video: $\mathbb{P}$ ID_VIDEO
info_video: ID_VIDEO $\leftrightarrow$ INFO_VIDEO
id_video = dom info_video

Entity: BANDA\_PRECIO

[ID\_BANDA, PRECIO]

INFO_BANDA_PRECIO
precio: PRECIO

INSTA_BANDA_PRECIO
id_banda: $\mathbb{P}$ ID_BANDA
info_banda_precio: ID_BANDA $\leftrightarrow$ INFO_BANDA_PRECIO
id_banda = dom info_banda_precio

## Relationships

Relationship: VIDEO-COPIA\_VIDEO

ASOCIA_VIDEO_COPIA_VIDEO
id_video: $\mathbb{P}$ ID_VIDEO
id_copia_video: $\mathbb{P}$ ID_COPIA_VIDEO
asocia_video_copia_video: ID_VIDEO $\leftrightarrow$ ID_COPIA_VIDEO
dom asocia_video_copia_video = ID_VIDEO
ran asocia_video_copia_video = ID_COPIA_VIDEO
asocia_video_copia_video $\sim \in$ ID_COPIA_VIDEO $\leftrightarrow$ ID_VIDEO

Relationship: BANDA\_PRECIO-VIDEO

ASOCIA_BANDA_PRECIO_VIDEO id_banda: $\mathbb{P}$ ID_BANDA id_video: $\mathbb{P}$ ID_VIDEO asocia_banda_precio_video: ID_BANDA $\leftrightarrow$ ID_VIDEO
dom asocia_banda_precio_video $\subseteq$ ID_BANDA ran asocia_banda_precio_video = ID_VIDEO asocia_banda_precio_video $\sim \in$ ID_VIDEO $\leftrightarrow$ ID_BANDA

Relationship: USUARIO-COPIA\_VIDEO

ASOCIA_USUARIO_COPIA_VIDEO id_usuario: $\mathbb{P}$ ID_USUARIO id_copia_video: $\mathbb{P}$ ID_COPIA_VIDEO asocia_usuario_copia_video: ID_USUARIO $\leftrightarrow$ ID_COPIA_VIDEO
dom asocia_usuario_copia_video $\subseteq$ ID_USUARIO ran asocia_usuario_copia_video $\subseteq$ ID_COPIA_VIDEO asocia_usuario_copia_video $\sim \in$ ID_COPIA_VIDEO $\leftrightarrow$ ID_USUARIO

FreeType Definitions:

NULL\_ID\_USUARIO ::= null\_id\_usuario | nid\_usuario  $\{\{$  ID\_USUARIO  $\}\}$

### System State

SYSTEM_STATE INSTA_USUARIO INSTA_COPIA_VIDEO INSTA_VIDEO INSTA_BANDA_PRECIO ASOCIA_VIDEO_COPIA_VIDEO ASOCIA_BANDA_PRECIO_VIDEO ASOCIA_USUARIO_COPIA_VIDEO
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

INITIAL\_SYSTEM\_STATE

SYSTEM\_STATE'

id\_usuario'= ∅

id\_copia\_video'= ∅

id\_video'= ∅

id\_banda'= ∅

asocia\_video\_copia\_video'= ∅

asocia\_banda\_precio\_video'= ∅

asocia\_usuario\_copia\_video'= ∅

### Schemas for operations

USUARIO\_OPERATION

△ SYSTEM\_STATE

▢ INSTA\_COPIA\_VIDEO

▢ INSTA\_VIDEO

▢ INSTA\_BANDA\_PRECIO

▢ ASOCIA\_VIDEO\_COPIA\_VIDEO

▢ ASOCIA\_BANDA\_PRECIO\_VIDEO

▢ ASOCIA\_USUARIO\_COPIA\_VIDEO

COPIA\_VIDEO\_OPERATION

△ SYSTEM\_STATE

▢ INSTA\_USUARIO

▢ INSTA\_VIDEO

▢ INSTA\_BANDA\_PRECIO

▢ ASOCIA\_BANDA\_PRECIO\_VIDEO

VIDEO\_OPERATION

---

- △ SYSTEM\_STATE
- ▢ INSTA\_USUARIO
- ▢ INSTA\_COPIA\_VIDEO
- ▢ INSTA\_BANDA\_PRECIO
- ▢ ASOCIA\_VIDEO\_COPIA\_VIDEO
- ▢ ASOCIA\_USUARIO\_COPIA\_VIDEO

---

BANDA\_PRECIO\_OPERATION

---

- △ SYSTEM\_STATE
- ▢ INSTA\_USUARIO
- ▢ INSTA\_COPIA\_VIDEO
- ▢ INSTA\_VIDEO
- ▢ ASOCIA\_VIDEO\_COPIA\_VIDEO
- ▢ ASOCIA\_BANDA\_PRECIO\_VIDEO
- ▢ ASOCIA\_USUARIO\_COPIA\_VIDEO

---

## Basic Operations

### Basic Operations for USUARIO

USUARIO\_ADD\_OK

---

USUARIO\_OPERATION

id\_usuario\_in? : ID\_USUARIO

info\_usuario\_in? : INFO\_USUARIO

---

id\_usuario\_in?  $\notin$  id\_usuario

id\_usuario' = id\_usuario  $\sqcup$  {id\_usuario\_in?}

info\_usuario' = info\_usuario  $\sqcup$  {id\_usuario\_in?  $\mapsto$  info\_usuario\_in?}

---

USUARIO_DELETE_OK USUARIO_OPERATION id_usuario_in? : ID_USUARIO
id_usuario_in? ∈ id_usuario id_usuario_in? ∉ dom asocia_usuario_copia_video info_usuario' = {id_usuario_in?} ↪ info_usuario id_usuario' = id_usuario \ {id_usuario_in?}

USUARIO_UPDATE_OK USUARIO_OPERATION id_usuario_in? : ID_USUARIO info_usuario_in? : INFO_USUARIO
id_usuario_in? ∈ id_usuario info_usuario' = info_usuario ⊕ {id_usuario_in? ↪ info_usuario_in?}

USUARIO_QUERY_OK ∃SYSTEM_STATE id_usuario_in? : ID_USUARIO info_usuario_out! : INFO_USUARIO
id_usuario_in? ∈ id_usuario info_usuario_out! = info_usuario(id_usuario_in?)

#### Response Messages:

RESPONSE ::= OK | EXISTS | NOT\_FOUND | HAS\_RELATIONSHIPS

SUCCESS_OP response! : RESPONSE
response! = OK

USUARIO_ERROR ∃SYSTEM_STATE id_usuario_in? : ID_USUARIO response! : RESPONSE
---------------------------------------------------------------------------------------

USUARIO_EXISTS USUARIO_ERROR
$id\_usuario\_in? \in id\_usuario$ $response! = EXISTS$
USUARIO_NOT_FOUND USUARIO_ERROR
$id\_usuario\_in? \notin id\_usuario$ $response! = NOT\_FOUND$
USUARIO_HAS_RELATIONSHIPS USUARIO_ERROR
$(id\_usuario\_in? \in dom\ asocia\_usuario\_copia\_video \wedge$ $response! = HAS\_RELATIONSHIPS)$

#### Basic Operations for COPIA\_VIDEO

COPIA_VIDEO_ADD_OK COPIA_VIDEO_OPERATION $id\_copia\_video\_in? : ID\_COPIA\_VIDEO$ $info\_copia\_video\_in? : INFO\_COPIA\_VIDEO$ $id\_video\_in? : ID\_VIDEO$ $null\_id\_usuario\_in? : NULL\_ID\_USUARIO$
$id\_copia\_video\_in? \notin id\_copia\_video$ $id\_copia\_video' = id\_copia\_video \sqcup \{id\_copia\_video\_in?\}$ $info\_copia\_video' = info\_copia\_video \sqcup \{id\_copia\_video\_in? \mapsto info\_copia\_video\_in?\}$ $asocia\_video\_copia\_video' = asocia\_video\_copia\_video \sqcup$ $\{id\_video\_in? \mapsto id\_copia\_video\_in?\}$ $(( null\_id\_usuario\_in? \neq null\_id\_usuario \wedge$ $nid\_usuario \sim null\_id\_usuario\_in? \in ID\_USUARIO \wedge$ $asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video \sqcup$ $\{nid\_usuario \sim null\_id\_usuario\_in? \mapsto id\_copia\_video\_in?\} )$ $\forall asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video )$



---

COPIA\_VIDEO\_DELETE\_OK

COPIA\_VIDEO\_OPERATION

id\_copia\_video\_in? : ID\_COPIA\_VIDEO

---

id\_copia\_video\_in? ∈ id\_copia\_video

asocia\_video\_copia\_video' = asocia\_video\_copia\_video ≡ {id\_copia\_video\_in?}

asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video ≡ {id\_copia\_video\_in?}

info\_copia\_video' = {id\_copia\_video\_in?} ↦ info\_copia\_video

id\_copia\_video' = id\_copia\_video \ {id\_copia\_video\_in?}

---



---

COPIA\_VIDEO\_UPDATE\_OK

COPIA\_VIDEO\_OPERATION

id\_copia\_video\_in? : ID\_COPIA\_VIDEO

info\_copia\_video\_in? : INFO\_COPIA\_VIDEO

---

id\_copia\_video\_in? ∈ id\_copia\_video

info\_copia\_video' = info\_copia\_video ⊕ {id\_copia\_video\_in? ↦ info\_copia\_video\_in?}

asocia\_video\_copia\_video' = asocia\_video\_copia\_video

asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video

---



---

COPIA\_VIDEO\_QUERY\_OK

SYSTEM\_STATE

id\_copia\_video\_in? : ID\_COPIA\_VIDEO

info\_copia\_video\_out! : INFO\_COPIA\_VIDEO

---

id\_copia\_video\_in? ∈ id\_copia\_video

info\_copia\_video\_out! = info\_copia\_video(id\_copia\_video\_in?)

---

#### Response Messages:

---

COPIA\_VIDEO\_ERROR

SYSTEM\_STATE

id\_copia\_video\_in? : ID\_COPIA\_VIDEO

response! : RESPONSE

---

COPIA_VIDEO_EXISTS
COPIA_VIDEO_ERROR
$id\_copia\_video\_in? \in id\_copia\_video$
response! = EXISTS

COPIA_VIDEO_NOT_FOUND
COPIA_VIDEO_ERROR
$id\_copia\_video\_in? \notin id\_copia\_video$
response! = NOT_FOUND

### Basic Operations for VIDEO

VIDEO_ADD_OK
VIDEO_OPERATION
$id\_video\_in? : ID\_VIDEO$
$info\_video\_in? : INFO\_VIDEO$
$id\_banda\_in? : ID\_BANDA$
$id\_video\_in? \notin id\_video$
$id\_banda\_in? \in id\_banda$
$id\_video' = id\_video \cup \{id\_video\_in?\}$
$info\_video' = info\_video \cup \{id\_video\_in? \mapsto info\_video\_in?\}$
$asocia\_banda\_precio\_video' = asocia\_banda\_precio\_video \cup \{id\_banda\_in? \mapsto id\_video\_in?\}$

VIDEO_DELETE_OK
VIDEO_OPERATION
$id\_video\_in? : ID\_VIDEO$
$id\_video\_in? \in id\_video$
$id\_video\_in? \notin \text{dom } asocia\_video\_copia\_video$
$asocia\_banda\_precio\_video' = asocia\_banda\_precio\_video \setminus \{id\_video\_in?\}$
$info\_video' = \{id\_video\_in?\} \Leftarrow info\_video$
$id\_video' = id\_video \setminus \{id\_video\_in?\}$

<p>VIDEO_UPDATE_OK</p> <p>VIDEO_OPERATION</p> <p>id_video_in? : ID_VIDEO</p> <p>info_video_in? : INFO_VIDEO</p> <hr/> <p>id_video_in? ∈ id_video</p> <p>info_video' = info_video ⊗ {id_video_in? ↦ info_video_in?}</p> <p>asocia_banda_precio_video' = asocia_banda_precio_video</p>
<p>VIDEO_QUERY_OK</p> <p>SYSTEM_STATE</p> <p>id_video_in? : ID_VIDEO</p> <p>info_video_out! : INFO_VIDEO</p> <hr/> <p>id_video_in? ∈ id_video</p> <p>info_video_out! = info_video(id_video_in?)</p>

## Response Messages:

<p>VIDEO_ERROR</p> <p>SYSTEM_STATE</p> <p>id_video_in? : ID_VIDEO</p> <p>response! : RESPONSE</p>
<p>VIDEO_EXISTS</p> <p>VIDEO_ERROR</p> <hr/> <p>id_video_in? ∈ id_video</p> <p>response! = EXISTS</p>
<p>VIDEO_NOT_FOUND</p> <p>VIDEO_ERROR</p> <hr/> <p>id_video_in? ∉ id_video</p> <p>response! = NOT_FOUND</p>

VIDEO\_HAS\_RELATIONSHIPS

VIDEO\_ERROR

$(id\_video\_in? \in \text{dom } \text{asocia\_video\_copia\_video} \wedge$   
 $\text{response!} = \text{HAS\_RELATIONSHIPS})$

#### Basic Operations for BANDA\_PRECIO

BANDA\_PRECIO\_ADD\_OK

BANDA\_PRECIO\_OPERATION

$id\_banda\_in? : ID\_BANDA$

$info\_banda\_precio\_in? : INFO\_BANDA\_PRECIO$

$id\_banda\_in? \notin id\_banda$

$id\_banda' = id\_banda \cup \{id\_banda\_in?\}$

$info\_banda\_precio' = info\_banda\_precio \cup \{id\_banda\_in? \mapsto info\_banda\_precio\_in?\}$

BANDA\_PRECIO\_DELETE\_OK

BANDA\_PRECIO\_OPERATION

$id\_banda\_in? : ID\_BANDA$

$id\_banda\_in? \in id\_banda$

$id\_banda\_in? \notin \text{dom } \text{asocia\_banda\_precio\_video}$

$info\_banda\_precio' = \{id\_banda\_in?\} \Leftarrow info\_banda\_precio$

$id\_banda' = id\_banda \setminus \{id\_banda\_in?\}$

BANDA\_PRECIO\_UPDATE\_OK

BANDA\_PRECIO\_OPERATION

$id\_banda\_in? : ID\_BANDA$

$info\_banda\_precio\_in? : INFO\_BANDA\_PRECIO$

$id\_banda\_in? \in id\_banda$

$info\_banda\_precio' = info\_banda\_precio \oplus \{id\_banda\_in? \mapsto info\_banda\_precio\_in?\}$

BANDA_PRECIO_QUERY_OK $\exists$ SYSTEM_STATE id_banda_in? : ID_BANDA info_banda_precio_out! : INFO_BANDA_PRECIO
id_banda_in? $\in$ id_banda info_banda_precio_out! = info_banda_precio(id_banda_in?)

#### Response Messages:

BANDA_PRECIO_ERROR $\exists$ SYSTEM_STATE id_banda_in? : ID_BANDA response! : RESPONSE
-------------------------------------------------------------------------------------------------

BANDA_PRECIO_EXISTS BANDA_PRECIO_ERROR id_banda_in? $\in$ id_banda response! = EXISTS
------------------------------------------------------------------------------------------------

BANDA_PRECIO_NOT_FOUND BANDA_PRECIO_ERROR id_banda_in? $\notin$ id_banda response! = NOT_FOUND
---------------------------------------------------------------------------------------------------------

BANDA_PRECIO_HAS_RELATIONSHIPS BANDA_PRECIO_ERROR (id_banda_in? $\in$ dom asocia_banda_precio_video $\wedge$ response! = HAS_RELATIONSHIPS)
------------------------------------------------------------------------------------------------------------------------------------------------------

#### Complete Operations

Complete operations for USUARIO

$$\text{USUARIO\_ADD} \triangleq (\text{USUARIO\_ADD\_OK} \wedge \text{SUCCESS\_OP})$$

$$\vee \text{USUARIO\_EXISTS}$$

USUARIO\_DELETE  $\triangleq$  (USUARIO\_DELETE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  USUARIO\_NOT\_FOUND

$\vee$  USUARIO\_HAS\_RELATIONSHIPS

USUARIO\_UPDATE  $\triangleq$  (USUARIO\_UPDATE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  USUARIO\_NOT\_FOUND

USUARIO\_QUERY  $\triangleq$  (USUARIO\_QUERY\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  USUARIO\_NOT\_FOUND

Complete operations for COPIA\_VIDEO

COPIA\_VIDEO\_ADD  $\triangleq$  (COPIA\_VIDEO\_ADD\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  COPIA\_VIDEO\_EXISTS

$\vee$  VIDEO\_NOT\_FOUND

$\vee$  USUARIO\_NOT\_FOUND

COPIA\_VIDEO\_DELETE  $\triangleq$  (COPIA\_VIDEO\_DELETE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  COPIA\_VIDEO\_NOT\_FOUND

COPIA\_VIDEO\_UPDATE  $\triangleq$  (COPIA\_VIDEO\_UPDATE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  COPIA\_VIDEO\_NOT\_FOUND

COPIA\_VIDEO\_QUERY  $\triangleq$  (COPIA\_VIDEO\_QUERY\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  COPIA\_VIDEO\_NOT\_FOUND

Complete operations for VIDEO

VIDEO\_ADD  $\triangleq$  (VIDEO\_ADD\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  VIDEO\_EXISTS

$\vee$  BANDA\_PRECIO\_NOT\_FOUND

VIDEO\_DELETE  $\triangleq$  (VIDEO\_DELETE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  VIDEO\_NOT\_FOUND

$\vee$  VIDEO\_HAS\_RELATIONSHIPS

VIDEO\_UPDATE  $\triangleq$  (VIDEO\_UPDATE\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  VIDEO\_NOT\_FOUND

VIDEO\_QUERY  $\triangleq$  (VIDEO\_QUERY\_OK  $\wedge$  SUCCESS\_OP)

$\vee$  VIDEO\_NOT\_FOUND

Complete operations for BANDA\_PRECIO

```

BANDA_PRECIO_ADD  $\triangleq$  (BANDA_PRECIO_ADD_OK  $\wedge$  SUCCESS_OP)
     $\forall$  BANDA_PRECIO_EXISTS

BANDA_PRECIO_DELETE  $\triangleq$  (BANDA_PRECIO_DELETE_OK  $\wedge$  SUCCESS_OP)
     $\forall$  BANDA_PRECIO_NOT_FOUND
     $\forall$  BANDA_PRECIO_HAS_RELATIONSHIPS

BANDA_PRECIO_UPDATE  $\triangleq$  (BANDA_PRECIO_UPDATE_OK  $\wedge$  SUCCESS_OP)
     $\forall$  BANDA_PRECIO_NOT_FOUND

BANDA_PRECIO_QUERY  $\triangleq$  (BANDA_PRECIO_QUERY_OK  $\wedge$  SUCCESS_OP)
     $\forall$  BANDA_PRECIO_NOT_FOUND

```

### 10.3. Lenguaje Z en TEX generado a partir del modelo entidad-relación

A continuación se adjunta el TEX generado por la herramienta de software desarrollada, basada en la integración efectiva de paradigmas de especificación de software, específicamente entre el modelo entidad-relación y el lenguaje Z.

```

\documentstyle[fuzz]{article}
\begin{document}
Z Notation
Generated by ERZ
Entities and attributes
Entity: USUARIO
\begin{zed}
[ID\_USUARIO, NOMBRE, FECHA\_NAC, DIRECCION]
\end{zed}
\begin{schema}{INFO\_USUARIO}
nombre: NOMBRE\\
fecha\_nac: FECHA\_NAC\\
direccion: DIRECCION
\end{schema}
\begin{schema}{INSTA\_USUARIO}
id\_usuario: \power ID\_USUARIO \\
info\_usuario: ID\_USUARIO \pfun INFO\_USUARIO
\where
id\_usuario = \dom info\_usuario

```

```

\end{schema}
Entity: COPIA\_VIDEO
\begin{zed}
[ID\_COPIA\_VIDEO]
\also
NULL\_COPIA\_VIDEO ::= no\_information\_copia\_video
\end{zed}
\begin{schema}{INFO\_COPIA\_VIDEO}
att\_copia\_video: NULL\_COPIA\_VIDEO
\end{schema}
\begin{schema}{INSTA\_COPIA\_VIDEO}
id\_copia\_video: \power ID\_COPIA\_VIDEO \
info\_copia\_video: ID\_COPIA\_VIDEO \pfun INFO\_COPIA\_VIDEO
\where
id\_copia\_video = \dom info\_copia\_video
\end{schema}
Entity: VIDEO
\begin{zed}
[ID\_VIDEO, TITULO, CATEGORIA]
\end{zed}
\begin{schema}{INFO\_VIDEO}
titulo: TITULO\
categoria: CATEGORIA
\end{schema}
\begin{schema}{INSTA\_VIDEO}
id\_video: \power ID\_VIDEO \
info\_video: ID\_VIDEO \pfun INFO\_VIDEO
\where
id\_video = \dom info\_video
\end{schema}
Entity: BANDA\_PRECIO
\begin{zed}
[ID\_BANDA, PRECIO]
\end{zed}
\begin{schema}{INFO\_BANDA\_PRECIO}
precio: PRECIO
\end{schema}
\begin{schema}{INSTA\_BANDA\_PRECIO}
id\_banda: \power ID\_BANDA \
info\_banda\_precio: ID\_BANDA \pfun INFO\_BANDA\_PRECIO
\where
id\_banda = \dom info\_banda\_precio
\end{schema}
Relationships
Relationship: VIDEO-COPIA\_VIDEO
\begin{schema}{ASOCIA\_VIDEO\_COPIA\_VIDEO}
id\_video: \power ID\_VIDEO \
id\_copia\_video: \power ID\_COPIA\_VIDEO \
asocia\_video\_copia\_video: ID\_VIDEO \rel ID\_COPIA\_VIDEO
\where

```



```

\dom asocia\_video\_copia\_video = ID\_VIDEO
\also
\ran asocia\_video\_copia\_video = ID\_COPIA\_VIDEO
\also
asocia\_video\_copia\_video \inv \in ID\_COPIA\_VIDEO \pfun ID\_VIDEO
\end{schema}
Relationship: BANDA\_PRECIO-VIDEO
\begin{schema}{ASOCIA\_BANDA\_PRECIO\_VIDEO}
id\_banda: \power ID\_BANDA \
id\_video: \power ID\_VIDEO \
asocia\_banda\_precio\_video: ID\_BANDA \rel ID\_VIDEO
\where
\dom asocia\_banda\_precio\_video \subseteq ID\_BANDA
\also
\ran asocia\_banda\_precio\_video = ID\_VIDEO
\also
asocia\_banda\_precio\_video \inv \in ID\_VIDEO \pfun ID\_BANDA
\end{schema}
Relationship: USUARIO-COPIA\_VIDEO
\begin{schema}{ASOCIA\_USUARIO\_COPIA\_VIDEO}
id\_usuario: \power ID\_USUARIO \
id\_copia\_video: \power ID\_COPIA\_VIDEO \
asocia\_usuario\_copia\_video: ID\_USUARIO \rel ID\_COPIA\_VIDEO
\where
\dom asocia\_usuario\_copia\_video \subseteq ID\_USUARIO
\also
\ran asocia\_usuario\_copia\_video \subseteq ID\_COPIA\_VIDEO
\also
asocia\_usuario\_copia\_video \inv \in ID\_COPIA\_VIDEO \pfun
ID\_USUARIO
\end{schema}
FreeType Definitions:
\begin{zed}
NULL\_ID\_USUARIO ::= null\_id\_usuario | nid\_usuario \ldata ID\_USUARIO
\rdata
\end{zed}
System State
\begin{schema}{SYSTEM\_STATE}
INSTA\_USUARIO \
INSTA\_COPIA\_VIDEO \
INSTA\_VIDEO \
INSTA\_BANDA\_PRECIO \
ASOCIA\_VIDEO\_COPIA\_VIDEO \
ASOCIA\_BANDA\_PRECIO\_VIDEO \
ASOCIA\_USUARIO\_COPIA\_VIDEO
\end{schema}
\begin{schema}{INITIAL\_SYSTEM\_STATE}
SYSTEM\_STATE'
\where
id\_usuario' = \emptyset

```

```

\also
id\_copia\_video'= \emptyset
\also
id\_video'= \emptyset
\also
id\_banda'= \emptyset
\also
asocia\_video\_copia\_video'= \emptyset
\also
asocia\_banda\_precio\_video'= \emptyset
\also
asocia\_usuario\_copia\_video'= \emptyset
\end{schema}
Schemes for operations
\begin{schema}{USUARIO\_OPERATION}
\Delta SYSTEM\_STATE \\\
\Xi INSTA\_COPIA\_VIDEO \\\
\Xi INSTA\_VIDEO \\\ \Xi INSTA\_BANDA\_PRECIO \\\
\Xi ASOCIA\_VIDEO\_COPIA\_VIDEO \\\
\Xi ASOCIA\_BANDA\_PRECIO\_VIDEO \\\
\Xi ASOCIA\_USUARIO\_COPIA\_VIDEO
\end{schema}
\begin{schema}{COPIA\_VIDEO\_OPERATION}
\Delta SYSTEM\_STATE \\\
\Xi INSTA\_USUARIO \\\ \Xi INSTA\_VIDEO \\\
\Xi INSTA\_BANDA\_PRECIO \\\ \Xi ASOCIA\_BANDA\_PRECIO\_VIDEO
\end{schema}
\begin{schema}{VIDEO\_OPERATION}
\Delta SYSTEM\_STATE \\\
\Xi INSTA\_USUARIO \\\ \Xi INSTA\_COPIA\_VIDEO \\\
\Xi INSTA\_BANDA\_PRECIO \\\ \Xi ASOCIA\_VIDEO\_COPIA\_VIDEO \\\
\Xi ASOCIA\_USUARIO\_COPIA\_VIDEO
\end{schema}
\begin{schema}{BANDA\_PRECIO\_OPERATION}
\Delta SYSTEM\_STATE \\\
\Xi INSTA\_USUARIO \\\ \Xi INSTA\_COPIA\_VIDEO \\\
\Xi INSTA\_VIDEO \\\ \Xi ASOCIA\_VIDEO\_COPIA\_VIDEO \\\
\Xi ASOCIA\_BANDA\_PRECIO\_VIDEO \\\
\Xi ASOCIA\_USUARIO\_COPIA\_VIDEO
\end{schema}
Basic Operations
Basic Operations for USUARIO
\begin{schema}{USUARIO\_ADD\_OK}
USUARIO\_OPERATION \\\
id\_usuario\_in? : ID\_USUARIO \\\
info\_usuario\_in? : INFO\_USUARIO \\\
\where
id\_usuario\_in? \notin id\_usuario
\also
id\_usuario' = id\_usuario \cup \{id\_usuario\_in?\}

```

```

\also
info\_usuario' = info\_usuario \cup \{id\_usuario\_in? \mapsto
info\_usuario\_in?\}
\end{schema}
\begin{schema}{USUARIO\_DELETE\_OK}
USUARIO\_OPERATION \\\
id\_usuario\_in? : ID\_USUARIO \\\
\where
id\_usuario\_in? \in id\_usuario
\also
id\_usuario\_in? \notin \dom asocia\_usuario\_copia\_video\also
info\_usuario' = \{id\_usuario\_in?\} \ndres info\_usuario\also
id\_usuario' = id\_usuario \setminus \{id\_usuario\_in?\}
\end{schema}
\begin{schema}{USUARIO\_UPDATE\_OK}
USUARIO\_OPERATION \\\
id\_usuario\_in? : ID\_USUARIO \\\
info\_usuario\_in? : INFO\_USUARIO \\\
\where
id\_usuario\_in? \in id\_usuario
\also
info\_usuario' = info\_usuario \oplus \{id\_usuario\_in? \mapsto
info\_usuario\_in?\}
\end{schema}
\begin{schema}{USUARIO\_QUERY\_OK}
\Xi SYSTEM\_STATE \\\
id\_usuario\_in? : ID\_USUARIO \\\
info\_usuario\_out! : INFO\_USUARIO
\where
id\_usuario\_in? \in id\_usuario
\also
info\_usuario\_out! = info\_usuario(id\_usuario\_in?)
\end{schema}
Response Messages:
\begin{zed}
RESPONSE ::= OK | EXISTS | NOT\_FOUND | HAS\_RELATIONSHIPS
\end{zed}
\begin{schema}{SUCCESS\_OP}
response! : RESPONSE
\where
response! = OK
\end{schema}
\begin{schema}{USUARIO\_ERROR}
\Xi SYSTEM\_STATE
\also
id\_usuario\_in? : ID\_USUARIO
\also
response! : RESPONSE
\end{schema}
\begin{schema}{USUARIO\_EXISTS}

```

```

USUARIO\_ERROR
\where
id\_usuario\_in? \in id\_usuario
\also
response! = EXISTS
\end{schema}
\begin{schema}{USUARIO\_NOT\_FOUND}
USUARIO\_ERROR
\where
id\_usuario\_in? \notin id\_usuario
\also
response! = NOT\_FOUND
\end{schema}
\begin{schema}{USUARIO\_HAS\_RELATIONSHIPS}
USUARIO\_ERROR
\where
(id\_usuario\_in? \in \dom asocia\_usuario\_copia\_video \land
response! = HAS\_RELATIONSHIPS)
\end{schema}
Basic Operations for COPIA\_VIDEO
\begin{schema}{COPIA\_VIDEO\_ADD\_OK}
COPIA\_VIDEO\_OPERATION \\\
id\_copia\_video\_in? : ID\_COPIA\_VIDEO \\\
info\_copia\_video\_in? : INFO\_COPIA\_VIDEO \\\
id\_video\_in? : ID\_VIDEO \\\
null\_id\_usuario\_in? : NULL\_ID\_USUARIO
\where
id\_copia\_video\_in? \notin id\_copia\_video
\also
id\_copia\_video' = id\_copia\_video \cup \{id\_copia\_video\_in?\}
\also
info\_copia\_video' = info\_copia\_video \cup \{id\_copia\_video\_in?\}
\mapsto info\_copia\_video\_in?\}
\also
asocia\_video\_copia\_video' = asocia\_video\_copia\_video \cup
\{id\_video\_in? \mapsto id\_copia\_video\_in?\}
\also
(( null\_id\_usuario\_in? \neq null\_id\_usuario \land \\\
nid\_usuario \inv null\_id\_usuario\_in? \in ID\_USUARIO \land \\\
asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video \cup \\\
\{nid\_usuario \inv null\_id\_usuario\_in? \mapsto id\_copia\_video\_in?
\} ) \lor asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video
)\end{schema}
\begin{schema}{COPIA\_VIDEO\_DELETE\_OK}
COPIA\_VIDEO\_OPERATION \\\
id\_copia\_video\_in? : ID\_COPIA\_VIDEO \\\
\where
id\_copia\_video\_in? \in id\_copia\_video\also
\also

```

```

asocia\_video\_copia\_video' = asocia\_video\_copia\_video \nrres
\{id\_copia\_video\_in?\} \also
\also
asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video \nrres
\{id\_copia\_video\_in?\} \also
info\_copia\_video' = \{id\_copia\_video\_in?\} \ndres
info\_copia\_video\also
id\_copia\_video' = id\_copia\_video \setminus \{id\_copia\_video\_in?\}
\end{schema}
\begin{schema}{COPIA\_VIDEO\_UPDATE\_OK}
COPIA\_VIDEO\_OPERATION \
id\_copia\_video\_in? : ID\_COPIA\_VIDEO \
info\_copia\_video\_in? : INFO\_COPIA\_VIDEO \
\where
id\_copia\_video\_in? \in id\_copia\_video
\also
info\_copia\_video' = info\_copia\_video \oplus \{id\_copia\_video\_in?
\mapsto info\_copia\_video\_in?\}\also
asocia\_video\_copia\_video' = asocia\_video\_copia\_video
\also
asocia\_usuario\_copia\_video' = asocia\_usuario\_copia\_video
\end{schema}
\begin{schema}{COPIA\_VIDEO\_QUERY\_OK}
\Xi SYSTEM\_STATE \
id\_copia\_video\_in? : ID\_COPIA\_VIDEO \
info\_copia\_video\_out! : INFO\_COPIA\_VIDEO
\where
id\_copia\_video\_in? \in id\_copia\_video
\also
info\_copia\_video\_out! = info\_copia\_video(id\_copia\_video\_in?)
\end{schema}
Response Messages:
\begin{schema}{COPIA\_VIDEO\_ERROR}
\Xi SYSTEM\_STATE
\also
id\_copia\_video\_in? : ID\_COPIA\_VIDEO
\also
response! : RESPONSE
\end{schema}
\begin{schema}{COPIA\_VIDEO\_EXISTS}
COPIA\_VIDEO\_ERROR
\where
id\_copia\_video\_in? \in id\_copia\_video
\also
response! = EXISTS
\end{schema}
\begin{schema}{COPIA\_VIDEO\_NOT\_FOUND}
COPIA\_VIDEO\_ERROR
\where
id\_copia\_video\_in? \notin id\_copia\_video

```

```

\also
response! = NOT\_FOUND
\end{schema}
Basic Operations for VIDEO
\begin{schema}{VIDEO\_ADD\_OK}
VIDEO\_OPERATION \\\
id\_video\_in? : ID\_VIDEO \\\
info\_video\_in? : INFO\_VIDEO \\\
id\_banda\_in? : ID\_BANDA
\where
id\_video\_in? \notin id\_video
\also
id\_banda\_in? \in id\_banda
\also
id\_video' = id\_video \cup \{id\_video\_in?\}
\also
info\_video' = info\_video \cup \{id\_video\_in? \mapsto
info\_video\_in?\}
\also
asocia\_banda\_precio\_video' = asocia\_banda\_precio\_video \cup
\{id\_banda\_in? \mapsto id\_video\_in?\}
\end{schema}
\begin{schema}{VIDEO\_DELETE\_OK}
VIDEO\_OPERATION \\\
id\_video\_in? : ID\_VIDEO \\\
\where
id\_video\_in? \in id\_video
\also
id\_video\_in? \notin \text{dom } asocia\_video\_copia\_video\also
\also
asocia\_banda\_precio\_video' = asocia\_banda\_precio\_video \nrres
\{id\_video\_in?\} \also
info\_video' = \{id\_video\_in?\} \ndres info\_video\also
id\_video' = id\_video \setminus \{id\_video\_in?\}
\end{schema}
\begin{schema}{VIDEO\_UPDATE\_OK}
VIDEO\_OPERATION \\\
id\_video\_in? : ID\_VIDEO \\\
info\_video\_in? : INFO\_VIDEO \\\
\where
id\_video\_in? \in id\_video
\also
info\_video' = info\_video \oplus \{id\_video\_in? \mapsto
info\_video\_in?\}\also
asocia\_banda\_precio\_video' = asocia\_banda\_precio\_video
\end{schema}
\begin{schema}{VIDEO\_QUERY\_OK}
\Xi SYSTEM\_STATE \\\
id\_video\_in? : ID\_VIDEO \\\
info\_video\_out! : INFO\_VIDEO

```

```

\where
id\_video\_in? \in id\_video
\also
info\_video\_out! = info\_video(id\_video\_in?)
\end{schema}
Response Messages:
\begin{schema}{VIDEO\_ERROR}
\Xi SYSTEM\_STATE
\also
id\_video\_in? : ID\_VIDEO
\also
response! : RESPONSE
\end{schema}
\begin{schema}{VIDEO\_EXISTS}
VIDEO\_ERROR
\where
id\_video\_in? \in id\_video
\also
response! = EXISTS
\end{schema}
\begin{schema}{VIDEO\_NOT\_FOUND}
VIDEO\_ERROR
\where
id\_video\_in? \notin id\_video
\also
response! = NOT\_FOUND
\end{schema}
\begin{schema}{VIDEO\_HAS\_RELATIONSHIPS}
VIDEO\_ERROR
\where
(id\_video\_in? \in \dom asocia\_video\_copia\_video \land
response! = HAS\_RELATIONSHIPS)
\end{schema}
Basic Operations for BANDA\_PRECIO
\begin{schema}{BANDA\_PRECIO\_ADD\_OK}
BANDA\_PRECIO\_OPERATION \\\
id\_banda\_in? : ID\_BANDA \\\
info\_banda\_precio\_in? : INFO\_BANDA\_PRECIO \\\
\where
id\_banda\_in? \notin id\_banda
\also
id\_banda' = id\_banda \cup \{id\_banda\_in?\}
\also
info\_banda\_precio' = info\_banda\_precio \cup \{id\_banda\_in? \mapsto
info\_banda\_precio\_in?\}
\end{schema}
\begin{schema}{BANDA\_PRECIO\_DELETE\_OK}
BANDA\_PRECIO\_OPERATION \\\
id\_banda\_in? : ID\_BANDA \\\
\where

```

```

id\_banda\_in? \in id\_banda
\also
id\_banda\_in? \notin \dom asocia\_banda\_precio\_video\also
info\_banda\_precio' = \{id\_banda\_in?\} \ndres info\_banda\_precio\also
id\_banda' = id\_banda \setminus \{id\_banda\_in?\}
\end{schema}
\begin{schema}{BANDA\_PRECIO\_UPDATE\_OK}
BANDA\_PRECIO\_OPERATION \\\
id\_banda\_in? : ID\_BANDA \\\
info\_banda\_precio\_in? : INFO\_BANDA\_PRECIO \\\
\where
id\_banda\_in? \in id\_banda
\also
info\_banda\_precio' = info\_banda\_precio \oplus \{id\_banda\_in?
\mapsto info\_banda\_precio\_in?\}
\end{schema}
\begin{schema}{BANDA\_PRECIO\_QUERY\_OK}
\Xi SYSTEM\_STATE \\\
id\_banda\_in? : ID\_BANDA \\\
info\_banda\_precio\_out! : INFO\_BANDA\_PRECIO
\where
id\_banda\_in? \in id\_banda
\also
info\_banda\_precio\_out! = info\_banda\_precio(id\_banda\_in?)
\end{schema}
Response Messages:
\begin{schema}{BANDA\_PRECIO\_ERROR}
\Xi SYSTEM\_STATE
\also
id\_banda\_in? : ID\_BANDA
\also
response! : RESPONSE
\end{schema}
\begin{schema}{BANDA\_PRECIO\_EXISTS}
BANDA\_PRECIO\_ERROR
\where
id\_banda\_in? \in id\_banda
\also
response! = EXISTS
\end{schema}
\begin{schema}{BANDA\_PRECIO\_NOT\_FOUND}
BANDA\_PRECIO\_ERROR
\where
id\_banda\_in? \notin id\_banda
\also
response! = NOT\_FOUND
\end{schema}
\begin{schema}{BANDA\_PRECIO\_HAS\_RELATIONSHIPS}
BANDA\_PRECIO\_ERROR
\where

```



```

(id\_banda\_in? \in \dom asocia\_banda\_precio\_video \land
response! = HAS\_RELATIONSHIPS)
\end{schema}
Complete Operations
Complete operations for USUARIO
\begin{zed}
USUARIO\_ADD \defs (USUARIO\_ADD\_OK \land SUCCESS\_OP)
\lor USUARIO\_EXISTS
\also
USUARIO\_DELETE \defs (USUARIO\_DELETE\_OK \land SUCCESS\_OP)
\lor USUARIO\_NOT\_FOUND
\lor USUARIO\_HAS\_RELATIONSHIPS
\also
USUARIO\_UPDATE \defs (USUARIO\_UPDATE\_OK \land SUCCESS\_OP)
\lor USUARIO\_NOT\_FOUND
\also
USUARIO\_QUERY \defs (USUARIO\_QUERY\_OK \land SUCCESS\_OP)
\lor USUARIO\_NOT\_FOUND
\end{zed}
Complete operations for COPIA\_VIDEO
\begin{zed}
COPIA\_VIDEO\_ADD \defs (COPIA\_VIDEO\_ADD\_OK \land SUCCESS\_OP)
\lor COPIA\_VIDEO\_EXISTS
\lor VIDEO\_NOT\_FOUND
\lor USUARIO\_NOT\_FOUND
\also
COPIA\_VIDEO\_DELETE \defs (COPIA\_VIDEO\_DELETE\_OK \land SUCCESS\_OP)
\lor COPIA\_VIDEO\_NOT\_FOUND
\also
COPIA\_VIDEO\_UPDATE \defs (COPIA\_VIDEO\_UPDATE\_OK \land SUCCESS\_OP)
\lor COPIA\_VIDEO\_NOT\_FOUND
\also
COPIA\_VIDEO\_QUERY \defs (COPIA\_VIDEO\_QUERY\_OK \land SUCCESS\_OP)
\lor COPIA\_VIDEO\_NOT\_FOUND
\end{zed}
Complete operations for VIDEO
\begin{zed}
VIDEO\_ADD \defs (VIDEO\_ADD\_OK \land SUCCESS\_OP)
\lor VIDEO\_EXISTS
\lor BANDA\_PRECIO\_NOT\_FOUND
\also
VIDEO\_DELETE \defs (VIDEO\_DELETE\_OK \land SUCCESS\_OP)
\lor VIDEO\_NOT\_FOUND
\lor USUARIO\_HAS\_RELATIONSHIPS
\also
VIDEO\_UPDATE \defs (VIDEO\_UPDATE\_OK \land SUCCESS\_OP)
\lor VIDEO\_NOT\_FOUND
\also
VIDEO\_QUERY \defs (VIDEO\_QUERY\_OK \land SUCCESS\_OP)
\lor VIDEO\_NOT\_FOUND

```

```

\end{zed}
Complete operations for BANDA\_PRECIO
\begin{zed}
BANDA\_PRECIO\_ADD \defs (BANDA\_PRECIO\_ADD\_OK \land SUCCESS\_OP)
\lor BANDA\_PRECIO\_EXISTS
\also
BANDA\_PRECIO\_DELETE \defs (BANDA\_PRECIO\_DELETE\_OK \land SUCCESS\_OP)
\lor BANDA\_PRECIO\_NOT\_FOUND
\lor USUARIO\_HAS\_RELATIONSHIPS
\also
BANDA\_PRECIO\_UPDATE \defs (BANDA\_PRECIO\_UPDATE\_OK \land SUCCESS\_OP)
\lor BANDA\_PRECIO\_NOT\_FOUND
\also
BANDA\_PRECIO\_QUERY \defs (BANDA\_PRECIO\_QUERY\_OK \land SUCCESS\_OP)
\lor BANDA\_PRECIO\_NOT\_FOUND
\end{zed}
\end{document}

```

#### 10.4. Documento generado en Latex

Con el formato TEX generado se puede procesar el archivo en cualquier editor Latex, por ejemplo, utilizando el editor MiKTeX [44] se puede generar un documento con formato matemático susceptible de publicación. A continuación se muestran fragmentos de la especificación en formato Latex:

Entity: BANDA\\_PRECIO

$[ID\_BANDA, PRECIO]$

$INFO\_BANDA\_PRECIO$   
 $precio : PRECIO$

$INSTA\_BANDA\_PRECIO$   
 $id\_banda : \mathbb{P} ID\_BANDA$   
 $info\_banda\_precio : ID\_BANDA \rightarrow INFO\_BANDA\_PRECIO$   
 $id\_banda = \text{dom } info\_banda\_precio$

Relationship: BANDA\_PRECIO-VIDEO

*ASOCIA\_BANDA\_PRECIO\_VIDEO*

$id\_banda : \mathbb{P} ID\_BANDA$

$id\_video : \mathbb{P} ID\_VIDEO$

$asocia\_banda\_precio\_video : ID\_BANDA \leftrightarrow ID\_VIDEO$

$\text{dom } asocia\_banda\_precio\_video \subseteq ID\_BANDA$

$\text{ran } asocia\_banda\_precio\_video = ID\_VIDEO$

$asocia\_banda\_precio\_video \sim \in ID\_VIDEO \leftrightarrow ID\_BANDA$

*BANDA\_PRECIO\_DELETE\_OK*

*BANDA\_PRECIO\_OPERATION*

$id\_banda\_in? : ID\_BANDA$

$id\_banda\_in? \in id\_banda$

$id\_banda\_in? \notin \text{dom } asocia\_banda\_precio\_video$

$info\_banda\_precio' = \{id\_banda\_in?\} \triangleleft info\_banda\_precio$

$id\_banda' = id\_banda \setminus \{id\_banda\_in?\}$

*SUCCESS\_OP*

$response! : RESPONSE$

$response! = OK$

*USUARIO\_ERROR*

$\exists SYSTEM\_STATE$

$id\_usuario\_in? : ID\_USUARIO$

$response! : RESPONSE$