GENERACIÓN DE LA INTERFAZ DE USUARIO DE NEGOCIO A PARTIR DE PATRONES DE NEGOCIOS BASADA EN LOS FUNDAMENTOS METODOLÓGICOS DE TD-MBUID

Por

Jorge Iván Triviño Arbelaez

Memoria

Presentada de conformidad con los requisitos para obtener el grado de Magister en Ingeniería con Énfasis en Informática

Asesor

Dr. William J. Giraldo

CoAsesor

Dr. Helmuth Trefftz

Universidad EAFIT

Diciembre 2015

DEDICATORIA

A mis padres, Héctor y Carmelita, quienes gracias a su crianza, exigencia e infinito apoyo han permitido que sea la persona que soy, por enseñarme el valor del esfuerzo, de la perseverancia, de soñar y ayudarme a lograr todos mis sueños, a ellos a quienes les debo mi vida, mis experiencias y todo lo que soy mi infinito agradecimiento y amor.

A mi amada esposa Paula Andrea, quien es el regalo más maravilloso que me ha dado Dios, quien es el motor que me da la energía y alegría para afrontar cada uno de mis días. Por brindarme su amor, apoyo, comprensión y motivación incondicional que me impulsan a lograr cada una de mis metas y a superar cualquier dificultad, Por ser mi amor, mi mejor amiga y porque eres mi vida.

A mis hermano, Fabián y Hugo, por las alegrías y experiencias de mi niñez y adolescencia, por brindarme su amistad y amor.

A mis Suegros, Silvio y Melva, por confiarme la vida de su hija, la joya más preciosa que les pudo dar nuestro Señor, y por aceptarme y recibirme como un hijo.

A Tato (q.e.p.d), quién me regaló desde el primer día que lo recibí y hasta el momento en que nos despedimos su amor y lealtad incondicional, por los momentos y recuerdos muy felices e inolvidables, siempre estarás en mi corazón, gracias por ser la mejor compañía para mis padres, contigo aprendí realmente el significado del dicho “El perro es el amigo más fiel y leal del hombre”.

AGRADECIMIENTOS

En primer lugar, quiero dar gracias a Dios, porque me ha regalado la bendición de mi vida, por colmarme de bendiciones y permitirme ser la persona que hoy soy.

A mi asesor William Giraldo, por brindar su amistad, conocimiento y experiencia en la planeación y dirección del proyecto, por su constante paciencia y motivación, sin las cuales no hubiese sido imposible la realización de este trabajo.

A Helmuth Trefftz, por compartir sus conocimientos los cuales fueron vitales en la culminación de este objetivo en mi vida.

A la universidad del Quindío, a la Universidad EAFIT y a todos aquellos que asumieron el reto de lograr el convenio entre las universidades que hizo posible la realización de la maestría en Ingeniería

A mis queridos amigos y compañeros del Grupo Sinfoci, William, María Lili, Alexandra, Jaime, Faber y Hamilton primero que todo gracias por su amistad, por permitirme ser parte de sus vidas; gracias por compartirme sus conocimientos los cuales fueron fundamentales en la realización de este proyecto, gracias por ofrecerme siempre la motivación, el empujón y el pocillo de café que día a día necesitaba.

Gracias a Robinson Arias, por poner su gran conocimiento y experiencia en función a ayudarme a lograr este gran objetivo, su aporte y ayuda fue fundamental en la culminación de este trabajo.

Gracias a todos aquellos que, aunque no mencioné, de una u otra manera participaron y me apoyaron en la finalización de este gran logro de mi vida, aunque demorado, muy demorado… por fin lo finalice.

RESUMEN

Este trabajo propone un proceso para el desarrollo de Interfaces de Usuario de negocio basada en Modelos. La propuesta está fundamentada en la aproximación metodológica *Task & Data – Model Based User Interface Development* (TD-MBUID) y en la aplicación de una triada formada por asociación de patrones de datos, plantillas de presentación y modelos de interacción. El proceso propuesto busca facilitar al desarrollador de la interfaz la interpretación de los modelos mentales que el usuario tiene de los datos; con el fin de mejorar el tiempo y calidad del diseño de los prototipos.

El proceso de desarrollo de desarrollo y la generación de la interfaz de usuario de negocio está soportado mediante el lenguaje DataForm y herramientas de Eclipse.

ÍNDICE DE GENERAL

DEDICATORIA 3

AGRADECIMIENTOS 5

RESUMEN i

ÍNDICE DE GENERAL iii

ÍNDICE DE FIGURAS x

ÍNDICE DE FIGURAS xiv

1. INTRODUCCIÓN 1

1.1 PLANTEAMIENTO DEL PROBLEMA 3

1.2 HIPÓTESIS DE LA INVESTIGACIÓN 5

1.3 OBJETIVOS 5

1.3.1 Objetivo General 5

1.3.2 Objetivos Específicos 5

1.4 METODOLOGÍA 6

2. MARCO DE REFERENCIA 9

2.1 FUNDAMENTOS TEÓRICOS 9

2.1.1 Desarrollo de Software 10

2.1.2 MDE 11

2.1.3 Desarrollo de Interfaces de Usuario Basado en Modelos (MBUID) 12

2.1.4 TD-MBUID (Task & Data – Model Based User Interface Development). 13

2.1.5 Patrones 15

2.2 ESTADO DEL ARTE 17

2.2.1 UI generation from task, domain and user models: the DB-USE approach" (Tran, 2010) . 17

2.2.2 Generative Pattern-Based Design of User Interfaces (Vanderdonckt & Simarro, 2010) 19

2.2.3 MultiPath (Limbourg, 2004) 20

2.2.4 Just-UI (Molina, Belenguer, & Pastor, 2003) 20
2.2.5 JANUS (Balzert, 1996) 21
2.2.6 Enhancing user interface design patterns with design rationale structures. (Janeiro, Barbosa, Springer, & Schill, 2009) 22
3. FUNDAMENTOS DE TD-MBUID 23
3.1 APROXIMACIÓN METODOLÓGICA TD-MBUID 23
3.1.1 Método de la ventana virtual 24
3.2 DESARROLLO DE LA INTERFAZ DE USUARIO EN TD-MBUID 26
3.3 LA INTERFAZ DE USUARIO DE NEGOCIO 30
3.4 DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO 31
3.4.1 componentes de la interfaz de usuario de negocio 32
3.5 DATAFORM 33
3.5.1 Patrón model – view - view Model 36
4. FUNDAMENTOS DE PATRONES 39
4.1 DEFINICIÓN 39
4.2 CUALIDADES DE LOS PATRONES 41
Encapsulación y abstracción 41
Apertura y Variabilidad 41
Generabilidad y Composicionabilidad 41
Equilibrio 41
4.3 CLASIFICACIÓN DE PATRONES. 41
4.3.1 Patrones arquitectónicos. 42
4.3.2 Patrones de diseño. 42
4.3.3 Patrones de programación, Idioms. 42
4.3.4 Patrones conceptuales. 42
4.3.5 Patrones de diseño. 42

4.3.6 Patrones de programación 43
4.3.7 Patrones de negocio. 43
4.3.8 Patrones de modelado de datos. 43
4.4 ADOPCIÓN DE LOS PATRONES A APLICAR EN ESTA TESIS 43
4.5 CATÁLOGO DE PATRONES 44
5. PROCESO METODOLÓGICO PARA LA DEFINICIÓN DEL MÉTODO PROPUESTO 47
5.1 DEFINICIÓN DE PATRONES 47
5.1.1 Identificación de patrones 48
5.1.2 Identificación de variabilidades 49
5.1.3 Definición de presentaciones para los patrones de negocios 50
5.1.4 Definición de la combinación entre patrones 55
5.1.5 Instancia de patrones. 56
5.2 DEFINICIÓN DE DATAFORM PARA CADA PATRÓN 57
5.2.1 Construir del Domain de cada patrón 61
5.2.2 Construcción de los modelos DataForm basada en los Patrones 61
5.2.3 Implementación de la Interacción 63
5.3 conformación de La triada que conforma LA VARIABILIDAD de un PATRÓN 64
5.4 CONSTRUCCIÓN DE ÁRBOL DE NAVEGACIÓN 67
5.5 PROTOTIPO DE INTERFAZ A PARTIR DE LAS VARIABILIDADES DE LOS PATRONES 68
5.5.1 Definición del contexto 68
5.5.2 Selección del patrón y variabilidad 69
6. PATRONES EN ESTA TESIS 71
6.1 PROCESO DE IDENTIFICACIÓN DE PATRONES 71
6.1.1 Identificación y asociación de patrones de negocios y patrones de presentación 71
6.1.2 Identificación de patrón de negocio 72
6.1.3 Plantilla de Presentación 72

6.1.4 Patrón de interacción 73
6.2 CATALOGO DE PATRONES DE NEGOCIO PROPUESTO 73
6.3 La Composición de los patrones propuestos: 75
6.4 FORMATO DE ESPECIFICACIÓN 76
6.5 PATRONES DE NEGOCIO 77
6.5.1 Patrón Ubicación Geográfica 77
6.5.2 Patrón Dirección 81
6.5.3 Patrón Unidad Organizacional 85
6.5.4 Patrón Actor 87
6.5.5 Patrón Empleo 91
6.5.6 Patrón Producto 94
6.5.7 Patrón Pago 97
6.5.8 Patrón Maestro Detalle 99
7. MÉTODO PROPUESTO PARA EL DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO 103
7.1 MÉTODO DE DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO 105
7.1.1 Análisis del contexto (Context Analysis) 106
7.1.2 Selección del patrón de negocio (Business Pattern Selection) 107
7.1.3 Adaptación del patrón de negocio (Business Pattern Adaption) 107
7.1.4 Preparación de la interfaz de usuario de negocio (BUI preparation) 107
7.1.5 Validación de la Interfaz de usuario de negocio (BUI validation) 108
7.2 Artefactos del Sistema 109
7.3 Lenguaje Data Form (DFL) 111
7.4 herramienta de soporte 113
7.4.1 Desarrollo de la Herramienta 114
7.4.2 Presentación de la Herramienta 119
7.4.3 Catálogo de Patrones 122

7.5 Resumen del método propuesto 125
8. VALIDACIÓN DE LA METODOLOGÍA PROPUESTA 126
8.1 CASO DE VALIDACIÓN 126
8.1.1 ANÁLISIS DEL CONTEXTO 126
8.1.2 SELECCIÓN DEL PATRÓN DE NEGOCIO 127
8.1.3 ADAPTACIÓN DEL PATRÓN DE NEGOCIO 134
8.1.4 PREPARACIÓN DE LA BUI 134
8.1.5 VALIDACIÓN DE LA BUI 137
8.2 Validación conceptual del enfoque con patrones 138
8.2.1 Objetivo de la Prueba 138
8.2.2 Criterios de Evaluación 138
8.2.3 Aplicación de la Prueba 139
8.2.4 Análisis de Resultados 139
8.2.5 Conclusiones de los resultados 142
9. CONCLUSIONES Y TRABAJOS FUTUROS 144
10. REFERENCIAS 146
ANEXO I ARTEFACTOS GENERADOS EN LA PRUEBA 150
A.1.1 artefactos del Grupo 1 150
VIEW 151
DOMAIN MODEL 151
VIEW MODEL 152
VIEW 153
DOMAIN MODEL 154
VIEW MODEL 154
A.1.2 artefactos del Grupo 2 155
VIEW 155

DOMAIN MODEL 156
VIEW MODEL 156
VIEW 157
DOMAIN MODEL 158
VIEW MODEL 158

ÍNDICE DE FIGURAS

Figura 1 Resumen de la metodología usada. 6
Figura 2 Flujo de desarrollo de la interfaz de usuario en TD-MBUID. (Orozco, 2010). 15
Figura 3 Mapeo modelo de tarea 18
Figura 4 Proceso propuesto para DB-USE 19
Figura 5 Organización de los artefactos del método de la ventana virtual dentro de la propuesta de desarrollo de la interfaz de usuario (Lauesen, 2005) 25
Figura 6 Flujo de desarrollo de la interfaz de usuario en TD-MBUID(Orozco, 2010). 27
Figura 7 Flujo de desarrollo de la interfaz de usuario de negocio(Orozco, 2010). 30
Figura 8 Desarrollo de la Interfaz de Usuario de Negocio(Orozco, 2010) 32
Figura 9 Esquema de Comunicación del Data Form. 34
Figura 10 Componentes intervienen en la creación del Modelo Data Form(Muñoz & Orozco, 2014) 34
Figura 11 Diagrama de un Modelo Data Form 35
Figura 12 Estructura del Patrón MVVM(Microsoft, 2012) 36
Figura 13 Adopcion del patron(Muñoz & Orozco, 2014) 37
Figura 14 Patrón Ubicación Geográfica 49
Figura 15 Ejemplo Variabilidades del patrón Ubicación Geográfica 50
Figura 16 Proceso Identificacion de bisquejo de presentación de los patrones 51
Figura 17 Patrón Dirección Variación Dirección Postal 51
Figura 18 Mockups del formulario para una direccion postal definido por un usuario 1. 53
Figura 19 Mockups del formulario para una dirección postal definido por un usuario 3 53
Figura 20 Descripcion interaccion deseada por los usuarios. 54
Figura 21 Variabilidad y presentaciones identificadas del patrón Dirección 55
Figura 22 Instancia del Patrón 57
Figura 23 Soporte DataForm a múltiples Presentaciones de un domino 58
Figura 24 View generada a partir del DataForm Model 59
Figura 25 Proceso de contrucción del DataForm asociadoa un patron. 60
Figura 26 Domain creado para la variabilidad Factura con emisor y receptor. 61

Figura 27 Ejemplo Modelo DataForm para el patrón Orden-Detalle variabilidad Factura desarrollado en toolDataForm 62
Figura 28 Patron Direccion relacionado con dos presentaciones diferentes del patron Ubicacion 63
Figura 29 Definición de la interacción de los elementos del DataForm 64
Figura 30 Proceso automatización Prototipos BUI 65
Figura 31 Composición de la triada 66
Figura 32 Proceso de Generación de la BUI a partir de la Triada 66
Figura 33 Catalogo de variabilidades implementado 67
Figura 34 Construcción del prototipo de interfaz a partir de los patrones de negocio 70
Figura 35 Proceso Identificacion y asociacion de patrones 71
Figura 36 Composicion de cada patrones de negocio del catalogo propuesto 75
Figura 37 Patrón Ubicación Geográfica 78
Figura 38 Estructura Variabilidad Ubicación Colombiana 79
Figura 39 Presentaciones de la variabilidad Ubicación Colombia 80
Figura 40 Estructura Variabilidad Ubicación Española 80
Figura 41 Presentaciones Variabilidad Ubicación Española 81
Figura 42 Patron Dirección 82
Figura 43 Estructura variabilidad Dirección Postal Colombiana 83
Figura 44 Presentaciones Variabilidad Dirección Postal Colombiana 83
Figura 45 Estructura variabilidad Dirección Postal Europea 84
Figura 46 Presentaciones variabilidad Dirección Postal Europea 84
Figura 47 Estructura Variabilidad Dirección Telefónica 84
Figura 48 Presentaciones Variabilidad Dirección Telefónica 85
Figura 49 Patrón Unidad Organizacional 85
Figura 50 Organigrama de una empresa genérica 86
Figura 51 Presentaciones Patrón Unidad Organizacional 87
Figura 52 Patrón Actor 88
Figura 53 Estructura Variabilidad persona Natural 89
Figura 54 Presentaciones Variabilidad Persona Natural 90
Figura 55 Estructura Variabilidad Persona Jurídica 90
Figura 56 Estructura Variabilidad Persona Jurídica 90
Figura 57 Patrón Empleado 91

Figura 58 Estructura Variabilidad Empleo Simple 93
Figura 59 Presentación Variabilidad Empleo Simple 93
Figura 60 Estructura Variabilidad Empleo Supervisado 94
Figura 61 Presentaciones Variabilidad Empleo Supervisado 94
Figura 62 Patrón Producto 95
Figura 63 Estructura Variabilidad Producto Simple 96
Figura 64 Presentaciones Variabilidad Producto Simple 96
Figura 65 Estructura Variabilidad Producto Compuesto 97
Figura 66 Presentaciones Variabilidad Producto Compuesto 97
Figura 67 Patrón Pago 98
Figura 68 Patrón Pago 99
Figura 69 Estructura Variabilidad Factura 100
Figura 70 Presentaciones Variabilidad Orden Detalle 101
Figura 71 Ejemplo reutilización de la ventana de contexto 104
Figura 72 Elementos que Componen la propuesta 105
Figura 73 flujo de desarrollo del método propuesto para la generación de la Interfaz de Usuario de Negocios a partir de patrones de negocios 106
Figura 74 Ejemplo aplicación de Patrones de Negocios 108
Figura 75 Artefactos para el desarrollo de la interfaz de usuario de negocio 109
Figura 76 Metamodelo de la Herramienta 115
Figura 77 Representación visual del elemento Ventana 116
Figura 78 Definición del mapping para la herramienta 117
Figura 79 ejemplo aplicación de Transformaciones 118
Figura 80 Vista General de la herramienta 119
Figura 81 Vista de selección de patrones 120
Figura 82 Vista de Generación de la Interfaz 121
Figura 83 Aplicación con el prototipo de interfaz generada. 122
Figura 84 Metamodelo del Catálogo de Patrones 123
Figura 85 Catalogo de variabilidades implementado 124
Figura 86 Visualización del catálogo de patrones en la herramienta 128
Figura 87 Árbol de variabilidad del Patrón Actor 129
Figura 88 Árbol variabilidad desplegado del patrón Actor y Paper Form Asociado 130

Figura 89 Business domain data de la variabilidad Persona Natural con Dirección postal 131
Figura 90 DataForm de la variabilidad Persona natural con dirección postal del patrón Actor..... 132
Figura 91 Artefactos Generados para el concepto Producto 133
Figura 92 Artefactos Generados para el concepto Factura de Venta..... 133
Figura 93 Domain model de la aplicación..... 134
Figura 94 Producciones del Domain 135
Figura 95 Implementación de la Interacción 136
Figura 96 Interfaz de Usuario Generada..... 137
Figura 97 PaperForm en representación de modelo mental del usuario..... 138
Figura 98 Proceso de la prueba aplicada 139
Figura 99 View realizada por el participante 1 del Grupo 1 151
Figura 100 Domain model Realizado por el participante 1 del grupo 1 151
Figura 101 View realizada por el participante 1 del grupo 1 152
Figura 102 View realizada por el participante 2 del grupo 1 153
Figura 103 Domain model Realizado por el participante 2 del grupo 1 154
Figura 104 View model realizada por el participante 2 del grupo 1..... 154
Figura 105 View realizada por el participante 1 del grupo 2..... 155
Figura 106 Domain model Realizado por el participante 1 del grupo 2 156
Figura 107 View model realizada por el participante 1 del grupo 2..... 156
Figura 108 View realizada por el participante 2 del grupo 2 157
Figura 109 Domain model Realizado por el participante 2 del grupo 2 158
Figura 110 View model realizada por el participante 2 del grupo 2..... 158

ÍNDICE DE TABLAS

Tabla 1 Ejemplo de combinaciones soportadas por el patron Dirección 56
Tabla 2 Patrones propuesto por clasificación 74
Tabla 3 : Notación DataForm (Muñoz & Orozco, 2014) 112
Tabla 4 Resumen Método Propuesto 125
Tabla 5 Resumen del analisis de los artefactos generados por cada grupo de prueba 140

CAPÍTULO PRIMERO

1. INTRODUCCIÓN

El desarrollo de la interfaz de usuario (IU) es una tarea de gran impacto dentro del proceso de construcción de software, ya que ella es la que ofrece la interacción entre la persona y computador, y por lo tanto constituye la cara visible de las aplicaciones y son responsables de la percepción que tenga el cliente del software (Constantine & Lockwood, 1999). No obstante el diseño de la IU suele ser basado en la experiencia del desarrollador, y por lo tanto de ella depende la calidad del prototipo.

Uno de los principales métodos usados en el ámbito de la ingeniería para plasmar el conocimiento adquirido por la experiencia es el uso de patrones, el cual es una forma de plasmar conocimiento reutilizable y donde la utilización del mismo de forma reiterada es garantía de éxito.

Para apoyar el desarrollo de IU, en los últimos años se ha realizado dentro del ámbito del MDE un gran esfuerzo en la investigación de métodos que permitan la inclusión del diseño de la interfaz de usuario dentro de un proceso de desarrollo basado en modelos, intentando obtener beneficios tales como la automatización de la generación de la interfaz de usuario (Moreno, 2002), y de esta manera disminuir la brecha entre el modelo mental del usuario y el modelo de la solución de la interfaz final del usuario, aumentando el grado de satisfacción del usuario con respecto al producto final.

Dentro de este enfoque se encuentra la aproximación metodológica TD-MBUID (*Task & Data – Model Based User Interface Development*), propuesta desarrollada por el Doctor William Joseph Giraldo, la cual se centra en combinar el diseño de las interfaces basadas en modelos de datos y de tareas. A diferencia de otros métodos MBUID, la interfaz de usuario se divide en dos, la interfaz de usuario de negocio y la interfaz abstracta, concreta y final.

Este trabajo de grado se centra en el nivel de Modelado de negocio de TD-MBUID, en el cual se analizan las tareas del negocio y los datos que las soportan para entender los modelos mentales de los usuarios y a partir de estos identificar la interfaz de usuario de negocio, generalmente formularios en papel, que soportan la entrada de información a los procesos de manera que sean realistas e independientes de la tecnología, así como de la navegación (Orozco, 2010).

Con el desarrollo de esta propuesta se propone un método que permita en el modelado de negocio definir y aplicar patrones de datos dentro de los modelos conceptuales (de dominio) vinculados a contexto de negocio y asociar a ellos patrones de presentación, de tal manera que se facilite al desarrollador la interpretación de los modelos mentales que el usuario tiene de los datos y lograr a partir de esa relación una generación automática de la interfaz de usuario de negocio – BUI (formularios en papel) que soportan la entrada de información a los procesos de negocio de una manera realista e independiente de la tecnología; con el fin de mejorar el tiempo y calidad del diseño de los prototipos, permitiendo en cuestión de minutos generar propuestas de BUI que permitan una validación temprana con los clientes.

Otro aspecto que motiva la realización de este trabajo, es que el Desarrollo de Interfaces de Usuario Basado en Modelos es una de las áreas de mayor interés en el grupo de investigación SINFOCI al que pertenezco, y además TD-MBUID fue concebido y desarrollado por nuestro director, y actualmente es el marco metodológico usado en los proyectos que se realizan a través del grupo de investigación.

Este documento se encuentra estructurado de la siguiente manera.

Capítulo 1 – Introducción: En este capítulo se presenta el contexto en el que surge el proyecto de investigación, se aborda la problemática que le dio origen; así mismo se detallan los objetivos y la metodológica empleada

Capítulo 2 – Marco de Referencia: Este capítulo muestran las bases teóricas y conceptuales que soportan el desarrollo de la tesis, asimismo se presentan trabajos relacionados a la temática de interés

Capítulo 3 – Fundamentos de TD-MBUID: En este capítulo se describe la aproximación metodológica TD-MBUID, se profundiza las disciplinas y procesos involucrados en esta metodología para definición la interfaz de usuario de negocio y se expone el lenguaje DataForm adoptado en la tesis.

Capítulo 4 – Fundamentos de Patrones: En este capítulo se presenta teoría general relacionada con patrones, las cualidades y clasificaciones.

Capítulo 5 – Proceso Metodológico para la Definición del Método Propuesto: En este capítulo se aplican los fundamentos descritos en los CAPÍTULOS 3, 4 Y 5 para construir el método de desarrollo de la interfaz que se trata en esta tesis, en forma de caso de estudio. Se presenta paso a paso las tareas realizadas tomando como base el proceso TD-MBUID y el lenguaje DataForm, a partir de este se va construyendo el método propuesto y se definen los artefactos requeridos para aplicar dicho método.

Capítulo 6 – Patrones en este Trabajo: Se presenta el catálogo de patrones propuestos, su documentación y conformación.

Capítulo 7 – Método para el desarrollo de la Interfaz de Usuario de Negocio: En este capítulo se explica el método propuesto para la generación automática de la interfaz de usuario de negocios a partir de la colección de patrones.

Capítulo 8 – Validación de la Propuesta: En este capítulo se muestra la factibilidad de la propuesta a través de su aplicación en un caso de laboratorio y se expone los resultados de una prueba realizada con estudiantes para analizar la utilización del método y patrones propuesto.

Capítulo 9 – Conclusiones y Trabajos Futuros: En este capítulo se expresa las conclusiones derivadas de la elaboración del trabajo y se hacen presupuestos para nuevos trabajos de investigación.

Anexo I. Artefactos Generador en la Prueba: En este anexo se muestran los modelos realizados por los participantes en la prueba.

1.1 PLANTEAMIENTO DEL PROBLEMA

En el ámbito del software, la interfaz de usuario es el componente que permite la comunicación en doble sentido entre el cliente y las aplicaciones, por lo tanto, garantizan la correcta apreciación de los clientes con respecto al programa de cómputo. Normalmente a un usuario no le interesa la funcionalidad, arquitectura o aspectos técnicos del Software sino la facilidad de uso y los resultados que obtenga a través de él(Granollers, 2004).

En la construcción de la interfaz de usuario, los desarrolladores se enfrentan a la dificultad de tener que interpretar y entender los modelos mentales que el usuario tiene de los datos, de las tareas, de los procesos para en consecuencia generar un prototipo que se aproxime a la necesidad real del cliente, asimismo, los usuarios normalmente no tienen la facilidad de obtener y expresar su propio modelo mental, lo que conlleva a que los diseños elaborados no cumplen con las expectativa de los usuario ni facilitan el normal desempeño de los procesos de los negocios.

Con respecto a esta problemática, **Donald Arthur "Don" Norman,** experto en diseño centrado en el usuario, director del the Design Lab, UC San Diego[1], quien fuese Arquitecto de Experiencia de Usuario en Apple afirmo: “Mi experiencia es que las especificaciones iniciales son normalmente erróneas, ambiguas o incompletas. En parte, porque son desarrolladas por gente que no entiende los problemas que afrontan los usuarios… Peor, los usuarios pueden no saber lo que quieren, así que tenerlos dentro del equipo de desarrollo no es la solución, porque un verdadero entendimiento de una herramienta sólo se consigue con el uso, en parte porque una nueva herramienta cambia el sistema, y de esta manera cambia tanto las necesidades como los requisitos”.

La brecha existente entre el modelo mental del usuario y el modelo de la solución puede generar un diseño inadecuado de la interfaz, ya sea en software para interactuar con una aplicación o en papel para interactuar directamente con el negocio, ocasiona problemas como (Vanderdonckt, 1997): confusión, no sabe qué hacer; frustración, no puede hacerlo que desea; pánico, miedo a cometer un error; estrés, errores repetitivos que hacen que se irrite.

La interfaz de usuario de negocio (interfaz de papel) no es ajena a esta problemática, un ejemplo de esto, es un formato físico para una transacción bancaria, si el usuario no comprende adecuadamente el formulario, la sola idea de equivocarse, le puede generar un conjunto de sentimientos negativos y por lo tanto una mala percepción del negocio.

Adicionalmente, el desarrollo de la interfaz de usuario, se ha convertido en un proceso artesanal en el cual cada diseñador realiza los prototipos de acuerdo a su propia experiencia, desaprovechando el conocimiento que él u otros profesionales hayan adquirido en procesos anteriores, lo que conlleva a pensar cómo solucionar problemas que ya tienen una solución probada.

Por ejemplo, Ciertos esquemas de información pueden ser comunes para múltiples negocios (ventas, compras, cliente), ya sea en el mismo o en distintos dominios, no obstante un desarrollador que no tenga suficiente experiencia tendría que generar desde cero el modelado de los datos y los prototipos de interfaz que los soporten , a menos que, exista la posibilidad de usar un patrones de datos y patrones de interfaz y así ofrecer sin tanto esfuerzo distintas propuestas de la interfaz de usuario ya sea de negocio o final.

En vista de lo expuesto, y como motivación de esta investigación, se busca desarrollar un método que facilite al desarrollador generar propuestas de prototipos de interfaz de usuario de negocio que se ajusten

1 Laboratorio de Diseño de la universidad de California http://designlab.ucsd.edu/

al modelo mental que el usuario tiene de sus propias tareas, haciendo uso de patrones de datos y de interfaz permitiendo una validación temprana de la solución planteada, que sirva de base para en una etapa posterior diseñar y construir con mayor calidad la interfaz de usuario final. En consecuencia, adaptar o extender a TD-MBUID, la cual es la metodología usada en el grupo de investigación, los componentes metodológicos y tecnológicos necesarios para la inclusión de esta facilidad.

1.2 HIPÓTESIS DE LA INVESTIGACIÓN

Esta investigación se sustenta en las siguientes hipótesis, las cuales surgen de acuerdo a lo expuesto en el planteamiento del problema:

Hipótesis 1: Dentro de un domino de negocio, se pueden definir patrones de datos que se asocien a plantillas de diseño de interfaz, basados en los modelos mentales de usuarios que faciliten el desarrollo de la interfaz de usuario de negocio.

Hipótesis 2: A partir de la asociación de patrones de datos y plantillas de interfaz de usuario y aplicando los fundamentos metodológicos de TD-MBUID, se puede lograr la generación automática de la interfaz de usuario de negocio.

Hipótesis 3: A partir de la asociación de patrones de negocio y plantillas de presentación y de los fundamentos metodológicos de TD-MBUID, no se puede lograr la generación automática de la interfaz de usuario de negocio.

1.3 OBJETIVOS

De acuerdo a las hipótesis antes enunciadas y para la realización de este trabajo se definen una serie de objetivos a cumplir.

1.3.1 Objetivo General

Definir los componentes metodológicos necesarios para la generación automática de la interfaz de usuario de negocio a partir de la asociación de patrones de datos y plantillas de diseño de interfaz de usuario.

1.3.2 Objetivos Específicos

- Analizar la estructura y contenido del marco metodológico TD-MBUID, en lo relacionado al modelado del dominio de negocio y al modelado de interfaz de usuario de negocio para conocer las restricciones y modelos que se deben utilizar para adaptarlos en la metodología.

- Definir un catálogo de patrones donde se especifique la asociación de patrones – variabilidad – presentación que permitan la construcción de prototipos de interfaz de usuario de negocios.
- Definir las transformaciones necesarias para la generación automática de la interfaz de usuario negocio y el respectivo prototipo de interfaz de usuario final.
- Validar la factibilidad de la propuesta resultante mediante la aplicación de un caso de laboratorio.
- Adaptar y definir los elementos tecnológicos que aplicados sobre el marco metodológico TD-MBIUD permitan utilizar esta propuesta.

1.4 METODOLOGÍA

Para validar la Hipótesis y alcanzar los objetivos propuestos que se plantea se propone la siguiente metodología.

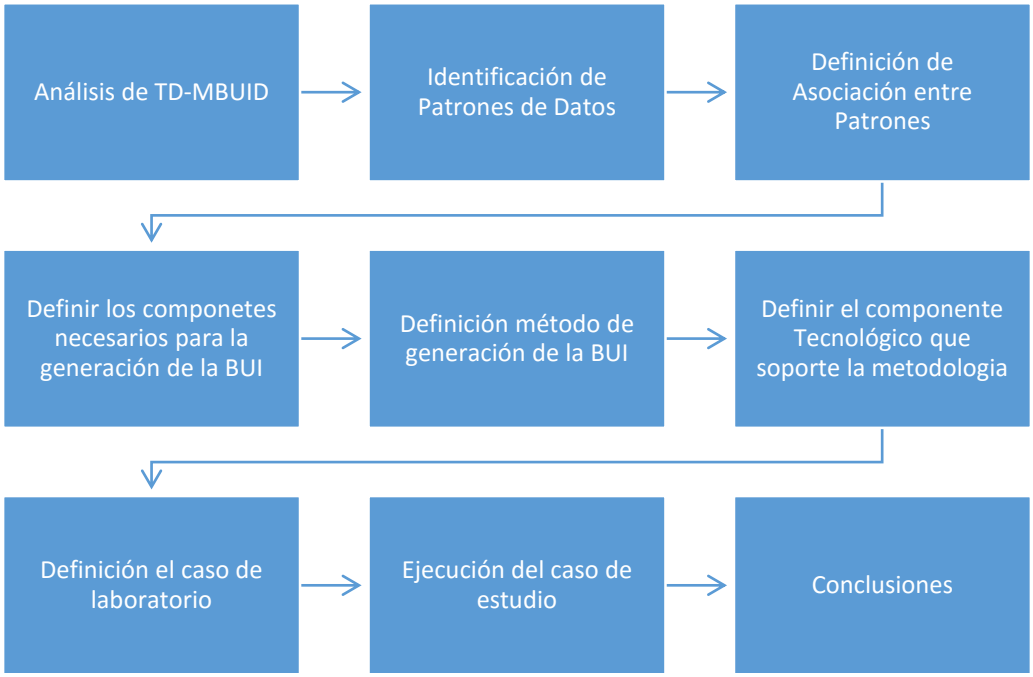

Figura 1 Resumen de la metodología usada.

Como se puede observar en la Figura 1, durante la ejecución de este proyecto es necesario estudiar y analizar la estructura del marco metodológico TD-MBUID, la cual es el fundamento teórico de esta tesis, enfocando este análisis específicamente en lo relacionado al modelado del dominio de negocio y al

modelado de la interfaz de usuario de negocio, esto permite conocer el proceso, actividades, tareas y artefactos que se deben utilizar para adaptar este trabajo en esta metodología, comprendiendo el grado en el cual se afecta el marco metodológico, conceptual y tecnológico en el cual se va a trabajar.

A continuación, se debe hacer un análisis de los patrones de datos y sus posibles plantillas de interfaz de usuario, por tal razón se iniciará con un estudio e identificación de patrones de datos, con el fin de definir una colección que sean comunes a diferentes contextos de negocios. Luego es necesario definir, con el apoyo de usuarios, las plantillas de presentaciones asociada a los patrones de datos y la forma de interactuar con la plantilla, definiendo el conjunto de plantillas de interfaz (incluida la interacción) que permitan la visualización de los datos del patrón.

El paso siguiente a realizar es definir el proceso necesario que permita a partir de los patrones y aplicando los fundamentos de TD-MBUID derivar la interfaz de usuario de negocio. En consecuencia, definir los elementos necesarios para lograr la generación automáticamente la interfaz de usuario de negocio, realizando la adaptación, modificación o definición de los elementos tecnológicos necesarios para la incorporación de este trabajo en el marco metodológico TD-MBUID.

Una vez definido el método de generación de la interfaz de usuario de negocios, la propuesta será validada a través de un caso de laboratorio que permita concluir su factibilidad.

Una vez finalizado el proceso antes mencionado se determina las conclusiones y aportes del proyecto.

CAPÍTULO SEGUNDO

2. MARCO DE REFERENCIA

En este capítulo se describe el marco de referencia, conformado por un marco teórico y un marco de antecedentes, en ellos se fundamenta el desarrollo de este trabajo. En una investigación nunca se parte de cero, sino que se parte de una base teórica y conceptual determinada. Ella guía todo el proceso, y en base a ella llegamos nuevamente al objetivo de toda investigación: generar un conocimiento válido y generalizable (Tamayo, 1999).

El marco Teórico constituye las teorías principales en las cuales se soporta esta tesis y el marco de antecedentes es un conjunto de proyectos o investigaciones que sirven de referencia para el desarrollo de este trabajo.

2.1 FUNDAMENTOS TEÓRICOS

El área principal en la que se desenvuelve este trabajo es el desarrollo de software, específicamente el desarrollo automático de la interfaz de usuario. Debido a que el desarrollo es una disciplina que pertenece a la ingeniería de software, empezaremos por definir esta última.

La ingeniería es definida por Hardy Cross como el arte de tomar una serie de decisiones importantes dado un conjunto de datos inexactos e incompletos con el fin de obtener, para un cierto problema, aquella entre las posibles soluciones, la que funcione de la manera más satisfactoria (Cross, 1952). En este sentido podemos decir que a la ingeniería de software le aplica la misma definición, con la salvedad de que las soluciones son productos relacionados al software. Barry W. Boehm, uno de los máximos referentes la define como “Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software” (Bohem, 1976).

2.1.1 Desarrollo de Software

La IEEE[2] define software como "el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación"; El desarrollo o construcción de software, normalmente se realiza a través de procesos que poseen unas reglas y actividades preestablecidas que buscan generar un producto que cumpla dos objetivos principales: el software debe realizar las tareas para las que fue diseñado, es decir, debe cumplir las funcionalidades especificadas por los requisitos del usuario; y también que el sistema sea manejable, fácil de aprender y fácil de comprender (Nielsen, 1993).

No obstante, El desarrollo de aplicaciones software comúnmente se realiza basado en la experiencia de los involucrados, lo que dificulta la consecución de estos objetivos; en este sentido (Simarro, 2005) afirma que este proceso tradicionalmente ha sido una labor artesanal y por lo tanto dependiente enormemente de la experiencia, él realiza un símil con un alfarero quien conoce las necesidades del destinatario de sus productos al situar asas o adecuarse a tamaños, formas o colores, y el ingeniero del software quien desarrolla su labor proporcionando productos de mayor o menor calidad, o más o menos próximos a los deseos del usuario final, dependiendo también de la experiencia que posea en tales desarrollos.

El componente responsable de lograr el segundo objetivo es la interfaz de usuario, esta ofrece en una aplicación la interacción entre el ser humano y la computadora, traduce del lenguaje binario de la máquina al lenguaje humano y viceversa (Jaquero, 2005) y por lo tanto constituyen la cara visible de las aplicaciones, siendo responsable de la percepción que tenga el cliente del software. (Constantine & Lockwood, 1999); al respecto (Granollers, 2004) afirma que al usuario no interesa la forma como se utilizan las funcionalidades de un programa, solamente está interesado en los resultados mostrados precisamente por dicha interfaz.

Es por esto que su desarrollo es una actividad de gran importancia e influencia en el diseño y tratamiento de un proyecto, el cual no debe ser abordado por cualquier profesional, como lo demuestra el experimento realizado por Gregg Bailey en el cual demostró que las interfaces desarrolladas por programadores tenían más errores y menor facilidad de uso que las realizadas por expertos de HCI. Lo que demuestra que la experiencia en la utilización y desarrollo de sistemas informáticos no es suficientes

[2] Institute of Electrical and Electronics Engineers

para el correcto diseño de la interfaz del usuario (Bailey, 1993). Se puede concluir de esto que la construcción de la interfaz, así como el software no puede ser una actividad artesanal.

Uno de los aspectos que dificulta el desarrollo de software complejo es la amplia separación conceptual existente entre el dominio de discurso del problema y el de la implementación (Frankel, 2004), para ayudar a disminuir esta brecha, ha surgido varios caminos, entre estos la ingeniería basado por modelos –MDE (de sus siglas en ingles), el cual engloba los enfoques que buscan la generación automática de código a través de modelos declarativos .

En este mismo sentido, el uso y definición de patrones ayuda a solucionar esa problemática, teniendo en cuenta lo indicado por (Kaisler, 2005), quien afirma que los programadores no construyen programas (incluyendo la interfaz de usuario) desde cero, sino que suelen emplear *patrones* apropiados, dentro del contexto de una problemática en particular y por los cuales constituyen soluciones probadas a problemas concretos, es decir, sirven como una guía, mecanismo o conocimiento que conlleva a realizar con éxito la satisfacción de una necesidad.

Por lo anterior, los patrones son un tema que se encuentra en auge en el mundo, ya que tratan de buscar en la realidad problemas comunes para abstraer la esencia del problema y de la solución, lo que permite obtener un catálogo de problemas-soluciones (Moreno, 2002).

2.1.2 MDE

El término *Model-Driven Engineering* (MDE) se usa normalmente para describir los enfoques de desarrollo de software, en los cuales a partir de la definición de modelos abstractos de sistemas software, se logran implementaciones concretas al aplicarles transformaciones sistemáticas (France & Rumpe, 2007), lo cual busca mejorar la coherencia de la solución implementada con la definición del problema.

La trasformación es el proceso que consiste en convertir un modelo en otro, ya sea grafico o textual (código fuente), estas operaciones son consideradas las más importantes a realizar sobre los modelos y una de las más críticas en MDE (Jouault & Kurtev, 2006).

En la actualidad, MDE se centra principalmente en el desarrollo de técnicas, métodos, procesos y herramientas de soporte que reduzcan efectivamente esta la brecha (Frankel, 2004).

Desarrollo de Software Dirigido por Modelos (MDSD)

Dentro del enfoque MDE, se plantea desde el punto de vista de la ingeniería de Software el MDSD, el cual cubre aspectos propios de esta Ingeniería la especificación de técnicas, métodos, herramientas y

lenguajes logrando una separación conceptual entre el espacio de modelado y el espacio de implementación. (Orozco, 2010).

MDSD busca que la generación de código se puede hacer de manera más o menos automática, por medio de transformaciones o traslaciones de modelos, de tal forma que los modelos se conviertan en una especie de "código fuente" del sistema a partir de los cuales la aplicación se genera.

Este enfoque pretende apoyar a los desarrolladores en su funciones, de tal manera que los liberes de tareas comunes y técnicas las cuales suelen ser una fuente de errores, se supone que, al aplicar sistemáticamente MDSD, la calidad de los sistemas de software, el grado de reutilización y, por tanto, implícitamente, la eficiencia de desarrollo, va a mejorar (Orozco, 2010).

En general, el MDSD es el proceso de generación de sistemas de software ejecutable a partir de modelos formales. Este proceso comienza a partir de modelos independientes de la computación (CIM), que se transforman en modelos independientes de la plataforma (PIM) para adaptarse a los modelos de plataforma específica (PSM) y, finalmente, dar como resultado el código fuente final de la aplicación (por ejemplo, Java) (Rech & Bunse, 2009).

2.1.3 Desarrollo de Interfaces de Usuario Basado en Modelos (MBUID)

El desarrollo de interfaces de usuario basado en modelos (MBUIDE – Model-Based User Interface Development Environment), es la vertiente del MDSD para el desarrollo de la interfaz de usuario; consiste en su especificación utilizando modelos declarativos que describen las distintas facetas y artefactos involucrados en el desarrollo de una interfaz de usuario, busca aumentar el nivel de abstracción usado en el diseño, permitiendo una generación automática parcial o total de la GUI. (Jaquero, 2005). Las tres principales características de los entornos MBUIDE son: soporte para la generación automática de interfaces de usuario, uso de métodos declarativos para la especificación de las interfaces y adopción de una metodología para soportar el desarrollo de la interfaz.

Los MBUIDE promueven el desarrollo iterativo de modelos declarativos por medio del uso de editores gráficos y de lenguajes de alto nivel, los cuales son una representación básica común para que las herramientas puedan razonar, y de este modo se facilite la construcción de herramientas que automaticen diversos aspectos del diseño de la interfaz.

Los *principales modelos declarativos* son: los modelos de datos, modelos de dominio, modelos de aplicación, modelos de tareas, modelos de diálogo, modelos de presentación (abstracta y concreta) y los

modelos de usuario. Cada uno de estos modelos representa distintos niveles de abstracción y granularidad en el proceso de desarrollo de la interfaz.

Gracias a esto MBUID ofrece mayor coherencia y reutilización en la construcción de nuevas interfaces a comparación de los métodos tradicionales en la construcción de interfaz de usuario (Szekely, Sukaviriya, Castells, Muthukumarasamy, & Salcher, 1995).

El uso de modelos declarativos de interfaces de usuario proporciona tres ventajas principales (Silva, 2002):

- Descripciones más abstractas de las interfaces de usuario.
- Facilitan la creación de métodos sistemáticos para el diseño e implementación de interfaces de usuario, puesto que aportan capacidades tales como el modelado en distintos niveles de abstracción, la posibilidad de un refinamiento incremental de los modelos y la reutilización de especificaciones.
- Crean el marco necesario para la automatización de los procesos de diseño e implementación de interfaces de usuario.

Dentro de los modelos declarativos centraremos la atención en el modelo de dominio, el cual describe los objetos de un dominio de aplicación. Especifica los datos que las personas utilizan, relacionados con las entidades del mundo real y las interacciones tal como son entendidas por los usuarios en relación con las acciones que es posible realizar sobre estas entidades (Orozco, 2010). Aunque El análisis del dominio tradicionalmente se usa los enfoques de desarrollo y no se considera en algo específico a considerar en el diseño de la interfaz de usuario, en esta propuesta se muestra como una fuente fundamental en el diseño de la GUI, ya que este modelo representa los datos que a través de ella se deben mostrar y manipular.

2.1.4 TD-MBUID (Task & Data – Model Based User Interface Development).

Dentro del enfoque MBUID, el Doctor William Joseph Giraldo Orozco en (Orozco, 2010) propone una aproximación metodológica para el desarrollo de Interfaces de Usuario Basada en Modelos, la cual se denomina TD-MBUID (*Task & Data – Model Based User Interface Development*), la cual se centra en combinar el diseño de las interfaces basadas en modelos de datos y de tareas,

Tradicionalmente, los sistemas MBUID proveen una separación en niveles de abstracción para los modelos de la interfaz de usuario. En el nivel abstracto, están todos los modelos declarativos con un alto nivel de

abstracción. Mientras que, en el nivel concreto, están los modelos ejecutables que permiten generar un *rendering* de la interfaz de usuario, a diferencia de esto, TD-MBUID provee una clasificación distinta de los niveles de abstracción de la interfaz de usuario. Los conceptos de abstracto y concreto, se relacionan exclusivamente con la interfaz de usuario y, tienen una connotación distinta a las anteriores especificaciones de sistemas MBUID. El nivel abstracto describe la información de una manera que es independiente de la modalidad. Mientras que, el nivel concreto describe la información de una manera que es independiente de la plataforma que soporta la interfaz de usuario.

En esta aproximación la interfaz se ha clasificado en dos tipos: la "interfaz de usuario de negocio", que incluye la información relacionada con la interacción a nivel de transacciones de negocio entre los usuarios de negocio y el negocio mismo, y la interfaz de usuario del sistema interactivo que comprende la interfaz abstracta, concreta y final.

La presentación de la interfaz de usuario de negocio está asociada a las tareas de dominio de alto nivel y se define a partir del modelo mental que el usuario tiene de los datos que están en el contexto de dichas actividades. El objetivo de esta interfaz es representar los formularios, generalmente en papel, que soportan la entrada de información a los procesos de negocio de una manera realista e independiente de la tecnología.

El desarrollo de la interfaz de usuario en TD-MBUID; se lleva a cabo en dos niveles de modelado, inicialmente, se analizan las tareas de alto nivel del negocio y los objetos del dominio que las soportan para entender los modelos mentales de los usuarios, y desarrollar la interfaz de usuario de negocio a partir de los datos que están siendo utilizados dentro del contexto de cada proceso. El objetivo del modelado de la interfaz de usuario de negocio es identificar los formularios, generalmente en papel, que soportan la entrada de información a los procesos de negocio de una manera realista e independiente de la tecnología; adicionalmente, se identifica la navegación que ocurre entre dichos formularios. Posteriormente, se analizan las tareas interactivas y los objetos del sistema que las soportan, para identificar el diálogo entre el usuario y el ordenador. Este diálogo se lleva a cabo mediante un conjunto de interfaces que son generadas a partir de una especificación abstracta, hasta obtener una especificación final ejecutable, en la Figura 2 se muestra el proceso descrito y encerrado en rojo las disciplinas involucradas en ese proyecto.

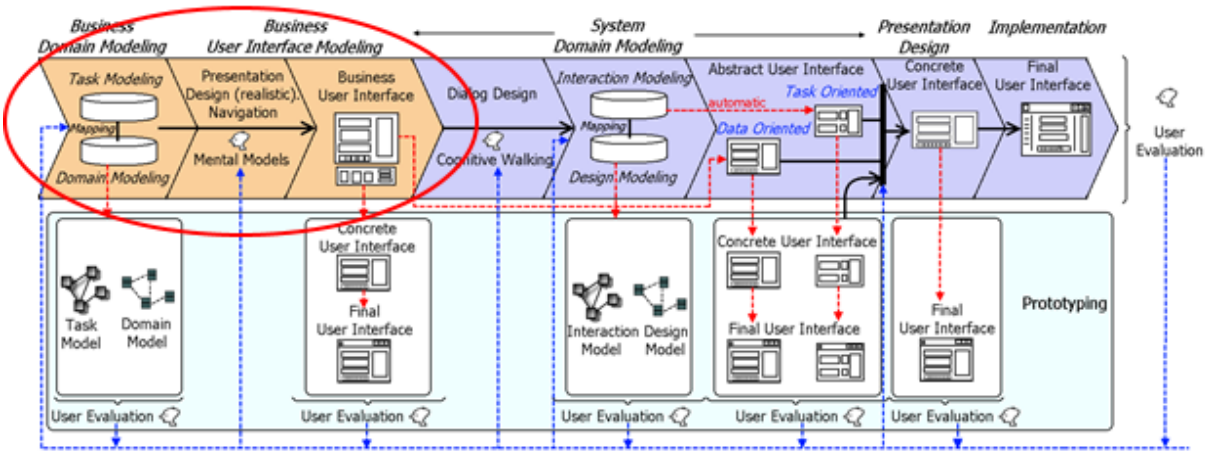

Figura 2 Flujo de desarrollo de la interfaz de usuario en TD-MBUID. (Orozco, 2010).

El presente trabajo para optar por el título de maestría, se centra dentro del modelado de negocio de la metodología TD-MBUID, y por lo tanto esta aproximación constituye la base teórica y práctica para la realización de este trabajo, razón por la cual será tratado a profundidad más adelante.

Como se describió anteriormente, la presentación de la interfaz de usuario de negocio está asociada a las tareas de dominio de alto nivel y se define a partir del modelo mental del usuario, la interpretación de este modelo no es una tarea sencilla para el desarrollador, quien normalmente la realiza basado en su experiencia, lo que genera un ámbito artesanal a la solución planteada. Para ayudar al desarrollador en esta tarea, y teniendo en cuenta que muchos negocios manejan datos similares, se propone hacer uso de patrones como fuente para plasmar este modelo.

2.1.5 Patrones

En área de la ingeniería de software, los patrones son un tema que se encuentra en auge en el mundo, ya que tratan de buscar en la realidad problemas comunes para abstraer la esencia del problema y de la solución y así obtener un catálogo de problemas-soluciones donde cada problema tiene un nombre y posibles soluciones. (Moreno, 2002). Un patrón puede verse como un conocimiento reutilizable, donde la su utilización reiterada es garantía de éxito para solucionar un problema común.

Un patrón recoge cómo pasar del qué al cómo, son una lingua franca (Erickson, 2000) que puede ser utilizada por todos aquellos que componen el equipo de desarrollo de un producto software, incluyendo al usuario, y, por otro, son una forma de documentación algo más precisa de la experiencia adquirida (Simarro, 2005).

Aunque el termino patrón es muy utilizado en el desarrollo de software, su origen está enmarcado en el ámbito de la disciplina arquitectónica, ya que su concepto fue introducido por el arquitecto Christopher Alexander, quien escribió varios libros sobre planificación urbana (Alexander, 1964, 1975, 1979; Alexander, Ishikawa, & Silverstein, 1977).

Alexander afirma que cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin tener que presentarse ni siquiera dos veces de la misma forma.

Aunque la fundamentación que Él platea se orienta a la arquitectura, estos conceptos se han utilizado en especial el desarrollo de software y actualmente en interacción persona - ordenador, ya que en ambas disciplinas se diseña artefactos para solucionar problemas, existiendo soluciones exitosas y recurrente que pueden ser tomadas como experiencia o conocimiento sensible a ser documentado.

En el desarrollo de software los patrones comenzaron a tener gran uso y éxito, gracias a la importancia y preeminencia del libro *Design Patterns: Elements of Reusable Object-Oriented Software* (Erich Gamma, Helm, Johnson, & Vlissides, 1995), cuyos autores son conocidos comúnmente como la pandilla de los cuatro (GoF). Éste libro recopila una serie de patrones de diseño orientados a objetos que ayudan a la construcción y posterior reúso de componentes de software. Los patrones ofrecen al desarrollador un conocimiento documentado que puede usar o adaptar para enfrentarse a muchos problemas que son frecuentes en su ámbito profesional, teniendo en cuenta que normalmente no construyen programas o soluciones desde cero, sino que suelen emplear *patrones* apropiados, dentro del contexto de una problemática en particular (Kaisler, 2005).

Asimismo, el uso de patrones de diseño se ha aplicado a IPO, por ejemplo (Erich Gamma et al., 1995; Noble, 1997) son propuestas de la aplicación del paradigmas orientados a objetos para solucionar problemas habituales que surgen de forma reiterada en el desarrollo de la GUI, asimismo, se han generado algunas propuesta de patrones que se enfocan en el diseño de la interfaz de usuario, como (Tidwell, 2010; Toxboe, 2014; Welie, 2008).

En términos generales, los patrones aportan experiencia, y lo hacen a través de la documentación de soluciones de diseño que llevan asociadas mejora en la calidad, es por esto, que su descripción es muy importante, ya que de ella depende la facilidad para entender y usar el patrón, debido a esto se han propuesto diversos formatos de especificación, uno de los más utilizados es el GoF Format definido por GoF (Erich Gamma et al., 1995).

Los patrones pueden clasificarse según el nivel de abstracción, según (Montero, López-Jaquero, & Molina, 2008) se clasifican en: Patrones de diseño, descrito en términos de construcciones de diseño como por ejemplo, objetos, clases, herencia, agregación y relaciones de uso. Patrones conceptuales, que se describe en términos de y conceptos de un dominio de aplicación (dominio del problema) y Patrones de programación, el cual está descrito en términos de construcciones de un lenguaje de programación. En (Eriksson & Penker, 1998), se presentan una clasificación que incluye además de los patrones de diseño y patrones arquitectónicos, los patrones de negocio, los cuales están enfocados a modelar los conceptos y recursos a nivel del negocio.

2.2 ESTADO DEL ARTE

En esta sección se presentarán los principales estudios encontrados en la literatura relacionados tanto con la generación automática de la interface de usuario como con el uso de patrones de datos. Estos antecedentes sirven como referencia para orientar y fundamentar la investigación.

2.2.1 UI generation from task, domain and user models: the DB-USE approach" (Tran, 2010)

Presenta un framework, un proceso metodológico, un metamodelo y un prototipo de aplicación llamado DB-USE. En este trabajo se propone un proceso que combina los modelos de tarea, de dominio y de usuario en conjunto para conducir el diseño de la interfaz de usuario del sistema de información y el código subyacente para la generación.

El modelo de tarea registra las tareas potenciales que los usuarios finales del sistema tienen que realizar para hacer su trabajo, de forma independiente a como las realizaría sobre una tecnología en particular, esta propuesta adapta el modelo de tareas CTTE, para poder definir las tareas específicas que usuario debe realizar sobre el sistema, en la Figura 3, se muestra el mapeo que realizaron del modelo CTTE al modelo usado por DB-USE

Task mapping	
Abstraction task	Action task
Interaction task (Control type)	Action task
Interaction task (Edit\Monitoring\Selection)	Operation task
Cooperation task	Action task
Application task	Operation task
User task	None

Figura 3 Mapeo modelo de tarea

Para el modelo de dominio, utilizan el modelo relacional de bases de datos, es usado para determinar los atributos de la interfaz de usuario concreta y definir las funciones más importantes de la aplicación sobre la base de datos.

Esta propuesta asocia los componentes de la tarea con los del modelado de dominio para generar la interfaz de usuario de acuerdo a las tareas y subtareas definidas, y definiendo en ellas los atributos que las tareas deben afectar de la base de datos, no obstante, no permite la reutilización de componentes de interfaz, ni de patrones de datos. En la Figura 4, se muestra el proceso definido en esta propuesta.

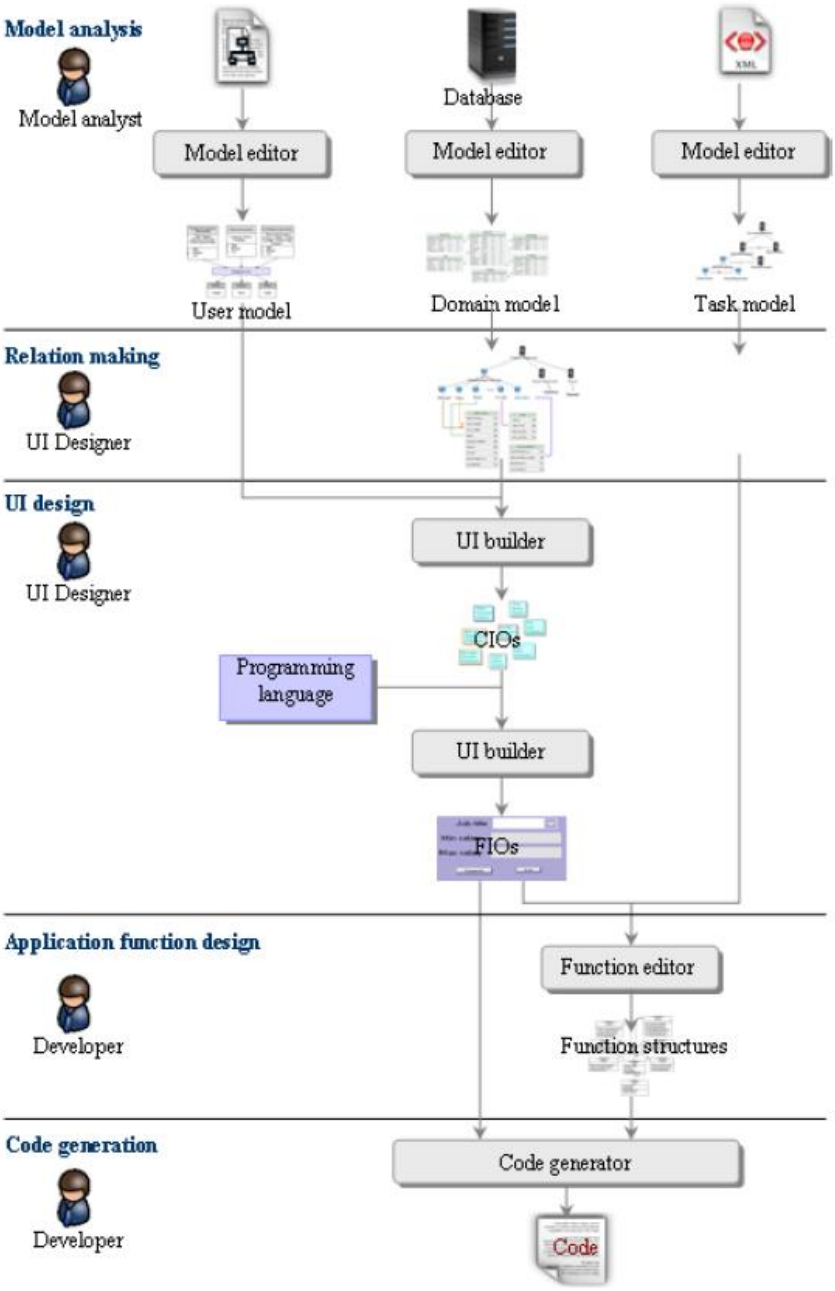

Figura 4 Proceso propuesto para DB-USE

2.2.2 Generative Pattern-Based Design of User Interfaces (Vanderdonckt & Simarro, 2010)

En este trabajo se propone un método para el desarrollo de interfaces gráficas de usuario basado en patrones generativos. Un patrón generativo contiene partes de interfaces de usuario diseñadas anteriormente expresados a través de modelos que son parcial o totalmente instanciados.

Estas partes pueden ser identificadas y re- aplicadas a un nuevo diseño creando instancias de las especificaciones contenida en los modelos ya definidos. El método consiste en los modelos típicos que se encuentra en el ciclo de desarrollo de la interfaz de usuario, como la tarea, dominio, interfaz abstracta de usuario, interfaz de usuario concreto, la interfaz de usuario final, modelo de contexto, y las asignaciones entre ellos. De tal manera que cualquier modelo prácticamente podría ser la fuente de un patrón y puede ser descrito, buscado, emparejado, recuperado, y ensambladas entre sí a fin de crear una nueva interfaz gráfica de usuario. Para este propósito, fue desarrollado una aplicación que administra los patrones generativos combinando lenguaje existente para la descripción de la interfaz de usuario (UsiXML - user interface extensible markup language) con conceptos que abordar los problemas naturales al momento de la descripción de patrones, debido a esto se introdujo PLML - Pattern Language Markup Language, para representar de manera uniforme los patrones interfaz de usuario. Una vez que se crean instancias de los patrones generativos, los modelos dan lugar a una ingeniería basada en el modelo soportado en la transformación de modelo a modelo y modelo a-código de compilación.

2.2.3 MultiPath (Limbourg, 2004)

MultiPath es un enfoque de desarrollo de la interfaz de usuario sobre la base del proyecto CAMALEÓN[3], que se centra en los aspectos técnicos de la adaptación a múltiples contextos y dispositivos. MultiPath se basa en el desarrollo transformacional para hacer un refinamiento progresivo de modelos abstractos en modelos concretos, hasta lograr obtener el código del programa. Este enfoque se realiza mediante varias vías de desarrollo (por ejemplo, ingeniería hacia adelante, ingeniería inversa, adaptación al contexto de uso) a fin de lograr el desarrollo mediante múltiples rutas. Para tal fin, MultiPath utiliza el lenguaje usiXML (Limbourg & Vanderdonckt, 2004) que define los distintos puntos de vista que pueden ser mantenidos en una interfaz de usuario de un sistema interactivo. Los distintos puntos de vista se estructuran jerárquicamente según su nivel de abstracción para describir las tareas del usuario, las clases de objetos, los aspectos de la presentación y del comportamiento de las interfaces de usuario, el contexto de uso, y un conjunto de *mapping* entre estas representaciones. MultiPath cuenta con una gran variedad de heurísticas de transformación para expresar múltiples caminos de desarrollo.

2.2.4 Just-UI (Molina, Belenguer, & Pastor, 2003)

Just-UI es una propuesta para la definición de patrones de interfaz de usuario que trabaja en términos del domino del problema. Just-UI permite la generación automática de interfaces de usuario de aplicaciones

[3] http://giove.isti.cnr.it/projects/cameleon.html

de gestión para diferentes plataformas a partir del modelado conceptual del sistema. La etapa de modelado conceptual es apoyada por OO-Method (Lopez, Insfr, Pelechano, Romero, & Merseguer, 1997). OO-Method es un método orientado a objetos para describir un esquema conceptual de sistemas de información por medio de diagramas compatibles con UML, y que se basa en el lenguaje formal OASIS (Lopez, Hayes, & Bear, 1992). Just-UI extiende OO-Method para explorar y navegar entre los datos (objetos de información) y capturar los requisitos de la interfaz de usuario. El objetivo final es modelar requisitos de interfaz de una manera abstracta y generar automáticamente la interfaz de usuario final para diferentes entornos de destino (Web, Windows, X11, UMTS, o PDA). Por lo tanto, la especificación obtenida a partir de Just-UI no depende de la plataforma, ya que las especificaciones no contienen detalles de diseño (Molina, Melia, & Pastor, 2003). Los analistas pueden crear especificaciones completas con muy poco esfuerzo sin necesidad de ser analistas expertos. Los patrones utilizados tienen aplicación directa en la ejecución final por medio de ventanas, formularios, páginas web o cualquier otra implementación.

2.2.5 JANUS (Balzert, 1996)

JANUS es un trabajo de la Universidad de Ruhr en Alemania. En esta propuesta a partir del Análisis Orientado a Objetos (OOA) (P. Coad & Yourdon, 1991), se genera parte de la interfaz de usuario asociada a las clases del modelo. A partir de este modelo empleado el cual incluye clases (con métodos, atributos) y relaciones (agregación y herencia), JANUS es capaz de generar, por ejemplo:

Un formulario por cada clase para representar una instancia de objeto, con controles para cada atributo y botones par cada método de la clase.

Un formulario por cada clase para mostrar conjuntos de objetos de una misma clase con una rejilla que contiene tantas columnas como atributos tiene la clase y botones para lanzar los métodos de la clase.

El método OOA proporciona un análisis del sistema, pero no captura requisitos de interfaz de usuario propiamente dicho, se apoya solo en el modelo de datos, sin incluir patrones de estos. Esta carencia de información acerca de los requisitos de la interfaz de usuario y la falta de patrones de datos e interfaz provocan que la interfaz generada por JANUS no sea muy refinada, que sólo contemple la parte de diálogo (parcialmente) de la interfaz y no se reutilicen componentes del modelado ni de la GUI.

2.2.6 Enhancing user interface design patterns with design rationale structures. (Janeiro, Barbosa, Springer, & Schill, 2009)

Este trabajo presenta un enfoque para representar los patrones de diseño de interfaz de usuario en una base de conocimientos que, presentando una estructuración entre patrones de diseño de interfaz de usuario y modelos de diseño lógico, permita ayudar a los diseñadores para buscar, comparar y reutilizar soluciones de diseño probadas en proyectos anteriores.

CAPÍTULO TERCERO

3. FUNDAMENTOS DE TD-MBUID

En este capítulo se busca introducir al marco de trabajo TD-MBUID, el cual como se indicó anteriormente es la base teórica y metodológica de esta propuesta, asimismo se presenta detalladamente el modelado de la interfaz de usuario de negocio, objetivo principal de este trabajo.

3.1 APROXIMACIÓN METODOLÓGICA TD-MBUID

TD-MBUID (*Task & Data – Model Based User Interface Development*) es una aproximación metodológica para el desarrollo de Interfaces de Usuario Basada en Modelos propuesta por el Doctor William Joseph Giraldo Orozco en (Orozco, 2010), la cual corresponde a un enfoque MBUID en el cual se centra en combinar el diseño de las interfaces basadas en modelos de datos y de tareas, asimismo se fundamenta en las propuestas de **Ventana Virtual** de Lauesen (Lauesen, 2005) la cual proporciona los fundamentos metodológicos para definir las disciplinas y las guías para el desarrollo completo de la interfaz de usuario y su integración con la funcionalidad; de la disciplina **Diseño de la usabilidad**: la disciplina denominada de Göransson (Göransson, Lif, & Gulliksen, 2003) que proporciona contenidos de métodos, artefactos y roles para el desarrollo de la interfaz de usuario integrables dentro del RUP y las bases para la usabilidad y por último de **usiXML,** lenguaje por Limbourg (Limbourg & Vanderdonckt, 2004) en donde se proporciona el soporte para la representación de los artefactos necesarios en la creación de modelos declarativos, a nivel abstracto y concreto, de la interfaz de usuario, así como la posibilidad del uso y creación de herramientas para soportar todo el proceso.

El desarrollo de la interfaz de usuario en TD-MBUID; se lleva a cabo en dos niveles de modelado, inicialmente, se analizan las tareas de alto nivel del negocio y los objetos del dominio que las soportan para entender los modelos mentales de los usuarios, y desarrollar la interfaz de usuario de negocio a partir de los datos que están siendo utilizados dentro del contexto de cada proceso. El objetivo del modelado de la interfaz de usuario de negocio es identificar los formularios, generalmente en papel, que soportan la entrada de información a los procesos de negocio de una manera realista e independiente de la

tecnología; adicionalmente, se identifica la navegación que ocurre entre dichos formularios. Posteriormente, se analizan las tareas interactivas y los objetos del sistema que las soportan, para identificar el diálogo entre el usuario y el ordenador. Este diálogo se lleva a cabo mediante un conjunto de interfaces que son generadas a partir de una especificación abstracta, hasta obtener una especificación final ejecutable.

3.1.1 Método de la ventana virtual

El método de desarrollo de la ventana virtual propone un método de diseño de interfaces de usuario sistemático, para que éstas sean fáciles de entender y apoyen al usuario de manera eficiente. Las ventanas virtuales son una realización gráfica muy temprana de la presentación de datos (Lauesen & Harning, 2001), que se realizan después del análisis de las tareas de dominio, pero antes del diseño del diálogo. Con el fin de diseñar, de una manera sistemática, este método recurre al mundo del programador, así como al mundo del especialista en IPO, pero eligiendo y adaptando sólo lo necesario. Estos son algunos de las consideraciones más importantes en las que se basa el método (Lauesen, 2005):

- Cómo medir la usabilidad
- Las pruebas de usabilidad y la evaluación heurística
- Diferentes tipos de prototipos
- Nivel de presentación de los datos
- Las leyes psicológicas que estudian la forma en que percibimos las pantallas
- El análisis de tareas y casos de uso
- El modelado de datos (desde el contexto del programador)
- La comprobación del diseño y el seguimiento de los problemas (desde el contexto del programador)
- Los modelos mentales de cómo los usuarios comprenden lo que no ven
- Las ventanas virtuales: la presentación de datos en pocas pantallas que cubren muchas de las tareas de manera eficiente
- El diseño de las funciones de la interfaz: agregar botones, menús, etc., en las pantallas en una forma estructurada y coherente

La Figura 5 presenta la organización de los artefactos del método de la ventana virtual desde el punto de vista que interesa en esta propuesta de desarrollo de la interfaz de usuario. Se presentan tres columnas para separar la información de las tareas que lleva a cabo el sistema, el usuario y, al mismo tiempo, los datos que soportan la interfaz de usuario. Esta clasificación es importante porque permite separar la

información que es relevante para los usuarios, de tal forma que, no solamente se evidencia la interactividad del sistema, sino también, el interés por la información relacionada con la usabilidad. Se observa como la información relacionada con la interfaz de usuario está formada por la columna "datos" (parte estructural) y por la columna "funciones del sistema" (parte dinámica). Por tanto, un prototipo de la interfaz de usuario es un componente del sistema que consta de unos datos de presentación soportados por un conjunto de funciones del sistema. Tanto el prototipo como la ventana virtual tienen un doble recuadro indicando que son artefactos que son visibles a los usuarios y que, por tanto, sirven para evaluar los diseños de la interfaz de usuario a medida que avanza el desarrollo.

Una ventana virtual es la presentación, orientada al usuario, de los datos persistentes del dominio del negocio (Lauesen, 2005). Durante su desarrollo, cada ventana virtual pasa por dos fases antes de llevarse a cabo un prototipo de la interfaz de usuario. En su primera fase, la ventana virtual puede ser hecha en papel para representar los datos y no contiene botones, menús y otras funciones. Desde el punto de vista de los sistemas MBUID esta representación se ubica en el nivel de presentación abstracta de la interfaz de usuario. En la segunda fase, la ventana virtual representa un *rendering* de la interfaz de usuario presentando la información de manera más realista para finalmente convertirse en pantallas en el ordenador, páginas Web, etc.

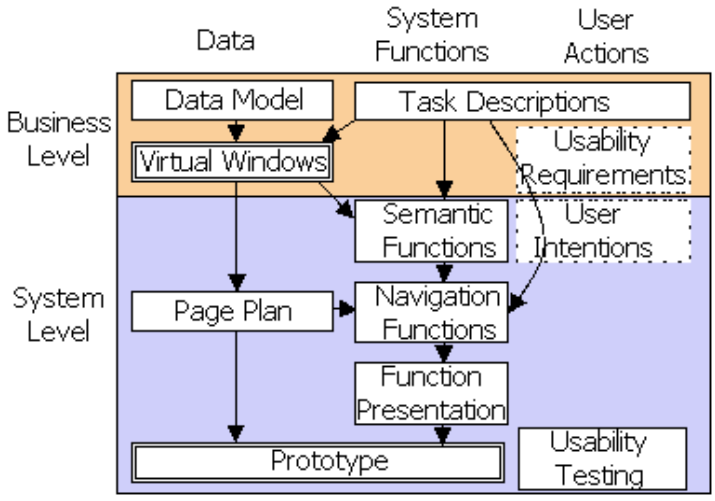

Figura 5 Organización de los artefactos del método de la ventana virtual dentro de la propuesta de desarrollo de la interfaz de usuario (Lauesen, 2005).

Como se observa en la Figura 5, existe un único artefacto "*Virtual Windows*" que es utilizado para describir la representación, tanto a nivel abstracto como concreto de la interfaz de usuario. Esto es el resultado de no utilizar un lenguaje para la descripción de las ventanas virtuales.

El modelado original de la ventana virtual en TD-MBUID se divide a nivel de negocio y a nivel del sistema informático, de acuerdo a sus dos fases de desarrollo. En su primera fase, la ventana virtual puede ser diseñada en papel y representa los datos del dominio. En la segunda fase, la ventana virtual representa un *rendering* de la interfaz de usuario de manera más realista.

La presentación de la interfaz de usuario de negocio está asociada a las tareas de dominio de alto nivel y se define a partir de los datos de dominio que están en el contexto de dichas actividades. La dinámica de la interfaz de usuario de negocio es la navegación que tiene lugar entre las distintas interfaces de negocio que soportan tareas de alto nivel. El modelo de diseño son los datos del sistema informático que están asociados a las interfaces abstractas y concretas. La interfaz abstracta está directamente relacionada con la actividad del sistema interactivo para representar tanto la presentación y el diálogo de una manera independiente de la modalidad del sistema. Finalmente, la interfaz de usuario final está relacionada con las pruebas de usabilidad que se efectúan a los usuarios del sistema.

3.2 DESARROLLO DE LA INTERFAZ DE USUARIO EN TD-MBUID

El desarrollo comienza con la definición de la *interfaz de negocio* a partir de los **datos del dominio** y continúa con la definición de la *interfaz de sistema* a partir de las **tareas interactivas**. Contrariamente a la mayoría de los enfoques de desarrollo, este método propone realizar la *interacción con el sistema* después de haber diseñado las *presentaciones* de los datos (las ventanas virtuales). La experiencia ha demostrado que esta secuencia ofrece mejores resultados para sistemas complejos (Lauesen, 2005).

Aunque TD-MBUID brinda la libertad al desarrollador de iniciar el desarrollo a partir de los datos o iniciarlo a partir de las tareas interactivas, en esta propuesta nos centraremos en únicamente en la primera opción.

En la Figura 6 se presenta el flujo de desarrollo de la interfaz de usuario en TD-MBUID; el cual se lleva a cabo en dos niveles de modelado. Inicialmente, se analizan las tareas de alto nivel del negocio y los objetos del dominio que las soportan. Estas actividades y objetos, aunque son independientes de la tecnología, permiten entender los modelos mentales de los usuarios para desarrollar la interfaz de usuario de negocio a partir de los datos que están siendo utilizados dentro del contexto de cada proceso. El objetivo del

modelado de la interfaz de usuario de negocio es identificar los formularios, generalmente en papel, que soportan la entrada de información a los procesos de negocio de una manera realista e independiente de la tecnología; adicionalmente, se identifica la navegación que ocurre entre dichos formularios. Posteriormente, se analizan las tareas interactivas y los objetos del sistema que las soportan, para identificar el diálogo entre el usuario y el ordenador. Este diálogo se lleva a cabo mediante un conjunto de interfaces que son generadas a partir de una especificación abstracta, hasta obtener una especificación final ejecutable.

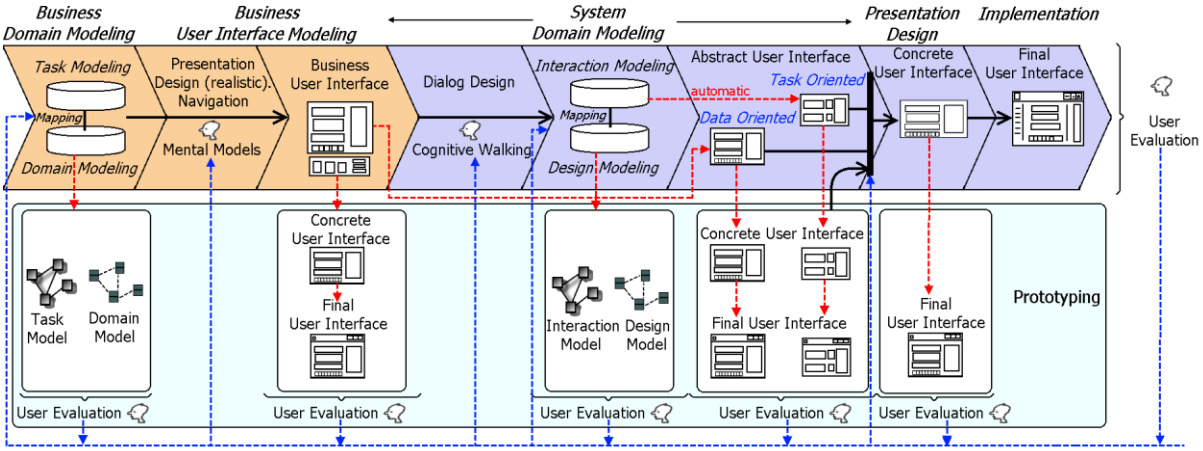

Figura 6 Flujo de desarrollo de la interfaz de usuario en TD-MBUID(Orozco, 2010).

Asimismo, en la Figura 6 se observar de manera vertical distintos flujos de desarrollo de la interfaz de usuario. De esta forma es posible evaluar los diseños a medida que evoluciona el ciclo de vida del proceso de desarrollo, de tal manera que a través de prototipado se permita facilitar a los desarrolladores y usuarios el análisis de los modelos de una forma apropiada mediante herramientas de *Runtime*.

En la fase de modelado de la interfaz abstracta, el desarrollador puede generar prototipos de interfaz siguiendo dos rutas distintas: la ruta basada en los datos (a partir de la interfaz de usuario de negocio) o la ruta basada en las tareas (a partir del modelo de interacción), como se indicó anteriormente nos centraremos en la primera ruta. Los prototipos son revisados, junto con los usuarios, para decidir cuál es el mejor diseño de interfaz abstracta.

A continuación, se describe cada uno de los pasos del desarrollo de la interfaz de usuario, como los menciona Orozco, 2010):

- *Diseño del dominio de negocio (Business Domain Design):* Se estudia el dominio de la aplicación para identificar junto con el usuario, las tareas que le sirven de apoyo y los datos que manipula. El resultado de este paso es una descripción de las tareas y los datos del dominio del negocio. La descripción de las tareas en este nivel define principalmente las intenciones o necesidades del usuario a muy alto nivel en el nivel del negocio. Se intenta entender el contexto del trabajo en general y el modelo de los datos que están asociados a ese contexto. Las actividades se especifican como si no existiera un tiempo exacto en el que tengan lugar, representando de esta forma tanto el pasado como el futuro del sistema.

- También se especifica la visión del sistema, los requerimientos de usabilidad y los perfiles de los usuarios. Para obtener estos resultados, se utilizan técnicas como entrevistas, observación de los usuarios, el estudio de los documentos existentes y los bosquejos de las interfaces en el ordenador.

- *Modelado de la interfaz de usuario de negocio (Business User Interface Modeling):* en este paso se lleva a cabo el diseño previo de las pantallas[4] que el usuario debe imaginar que están detrás de la pantalla física - en la parte posterior del sistema. Estas pantallas son las ventanas virtuales (Lauesen, 2005). Éstas se asemejan a las ventanas finales y tienen detalles gráficos con contenido de datos realistas, pero sin especificar *widgets* concretos (por ejemplo, 'botones'). El diseño de las pantallas se lleva a cabo a partir de los datos del modelo de dominio de negocio: el modelo de datos y las descripciones de las tareas.

- En esta etapa podrá también evaluarse el diseño. Dicha prueba se podrá hacer de muchas maneras; por ejemplo, aplicando pruebas de comprensibilidad (una prueba de usabilidad simplificada), se verifican los accesos a los elementos de las pantallas y a los datos que se requieren para llevar a cabo cada tarea. Este paso a menudo da lugar a cambios en las descripciones de las tareas y en el modelo de datos.

- *Diseño del dominio del sistema (System Domain Design):* en este paso se realiza el diseño interactivo, que muestra el modo en el que el usuario puede interactuar con el sistema. Este paso constituye el segundo nivel de modelado del dominio, pero esta vez a nivel de sistema. Se lleva a cabo para identificar y clasificar las tareas que son propias de los usuarios y las que serán llevadas a cabo por el ordenador. De este modo se crea un diseño conceptual que describe la estructura general de la interfaz de usuario. Se desarrollan escenarios de uso junto con los usuarios. Se identifican los principales componentes de la interfaz de usuario. Por lo general, se crean prototipos en papel para ilustrar posibles soluciones de diseño a un alto nivel.

  Se genera una interfaz de usuario abstracta que conserva el equilibrio entre las pantallas identificadas en el paso previo y las generadas, normalmente de forma automática, a partir del modelo de interacción. De esta forma se promueve un equilibrio entre el modelado, basado en

[4] Se usa el término pantalla para referirse al modelo mental que se forma el usuario respecto de los datos que manipula. No se quiere utilizar el término ventana como componente concreto de interfaz de usuario.

los datos y basado en las tareas, de la interfaz de usuario. En este paso de diseño se lleva a cabo un *mapping* entre las actividades y los datos para definir los tipos de controles de la interfaz de usuario que estarán soportando la interacción. En este *mapping* se determinan aspectos como la cardinalidad y la funcionalidad que soportará el acceso o actualización de la información sobre la interfaz de usuario y el aspecto de cada control según las acciones que soporta.

- *Diseño de la presentación (Presentation Design):* en este paso se crea la interfaz de usuario concreta, que se especifica para soportar una modalidad específica dentro del sistema. Esta interfaz concreta se genera inicialmente de manera automática a partir de la interfaz abstracta. Se tendrán en cuenta reglas de diseño y ergonomía y requerimientos de usabilidad. A continuación, se combinan las ventanas con la intención de reducir su número, promover la reutilización de las mismas y soportar adecuadamente los modelos mentales de los usuarios. Posteriormente, se seleccionan los controles que necesitará el usuario para llevar a cabo sus tareas. Éstas serán soportadas por las ventanas de la interfaz de usuario concreta. Se decide, así mismo, el modo de llevar a cabo las acciones sobre la interfaz identificadas en el paso anterior; por ejemplo, mediante botones, menús, arrastrar y soltar, etc. Una vez definidas las ventanas y los controles, se procede a la asignación de las funciones de navegación según sea el modelo de navegación general del sistema. Se pueden utilizar diagramas de estado para definir las funciones necesarias de navegación, el menú de navegación, la información y la funcionalidad que simula el sistema real.

- *Implementación (Implementation)*: en este paso se lleva a cabo la programación del modo habitual. Durante la implementación se deben buscar errores del programa y problemas de interacción entre los distintos programas.

- *Prototipado y evaluación (Prototype & Evaluation)*: en este paso se evalúa la usabilidad de las soluciones diseñadas y se comparan con las metas de usabilidad. Las evaluaciones se pueden realizar sobre bocetos preliminares, así como sobre prototipos totalmente interactivos y el sistema final. Para ello, se combinan las pantallas y las funciones en un prototipo, al que se pueden añadir textos de ayuda y mensajes de error. Después de aplicar las pruebas de usabilidad al prototipo, se evalúan los problemas y se corrigen en la medida de lo posible. En este momento se puede llevar a cabo una evaluación heurística para apuntar a los posibles problemas. Sin embargo, se recomienda que se lleven a cabo evaluaciones desde las etapas anteriores. De esta forma, aunque en esta etapa se mantendrán algunos problemas, si se ha hecho un buen trabajo de diseño en los pasos anteriores, los problemas de usabilidad tienden a ser fáciles de corregir sin que conlleve una reestructuración importante de las pantallas.

Esta propuesta se centra la opción que ofrece TD-MBUID para el desarrollo de la interfaz de usuario a partir de los datos, incluyendo a esta aproximación los componentes metodológicos y tecnológicos necesario para la definición y utilización de patrones de datos que permita facilitar la identificación y definición del modelo de dominio (modelo de datos), definiendo para cada patrón un conjunto de variabilidades (adaptaciones del patrón según determinado contexto) con sus respectivas formas de

presentaciones (definidas junto con los usuario) y a partir de esto generar de manera casi inmediata los prototipos de la interfaz de usuario de negocios.

3.3 LA INTERFAZ DE USUARIO DE NEGOCIO

El modelado de la interfaz de usuario, como se mencionó anteriormente, es una adaptación del método de diseño de interfaces virtuales definido en (Lauesen, 2005), aunque en TD-MBUID se ha decidido cambiar el término *ventana virtual* por el de *interfaz de usuario de negocio* o en su forma resumida, *ventana de contexto*. Una ventana de contexto es una presentación de los datos persistentes, orientada al usuario. Se denomina ventana de contexto porque es la interfaz que soporta el contexto de uso relacionado con un grupo de actividades y que posee la información necesaria para ejecutar dichas actividades. Como se indica en la Figura 7, El diseño de la BUI se lleva a cabo a partir de los datos del modelo de dominio de negocio: el modelo de datos y las descripciones de las tareas, a partir de estos y basados en el modelo mental del usuario se realiza el diseño de ventanas que permitan la presentación realista de los datos del dominio que soportan las tareas, generando así un conjunto de ventanas de contexto que conforman la interfaz de usuario de negocio.

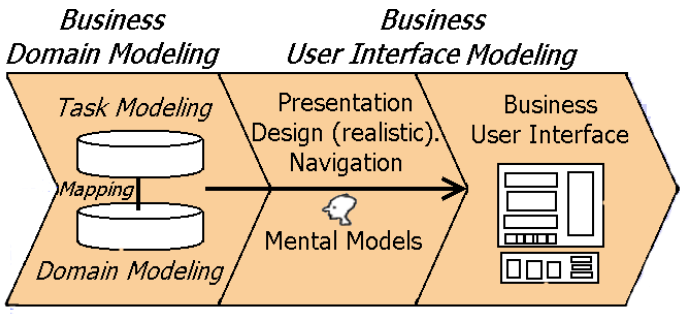

Figura 7 Flujo de desarrollo de la interfaz de usuario de negocio(Orozco, 2010).

El objetivo es, si es posible, diseñar un número mínimo y óptimo de ventanas de contexto para cada tarea de alto nivel, en el modelo de dominio, que brinden apoyo al conjunto de sus tareas de menor granularidad y no sólo a cada una de manera aislada, es decir agrupando las tareas pequeñas y relacionadas que se llevan a cabo en la misma sesión de trabajo. De tal manera que la interfaz de usuario de una tarea de inter-acción estará compuesta por una o por un conjunto de ventanas de contexto que contengan toda la información necesaria para la ejecución de la totalidad de sus tareas internas.

La ventana de contexto para una tarea podrá ser reutilizada como un patrón para la interfaz de usuario durante el diseño de la interacción entre el usuario y el sistema que tiene lugar dentro de cada una de las tareas de menor granularidad.

Para diseñar la interfaz de usuario de negocio y promover la reutilización se aconseja identificar, con ayuda de los usuarios, las tareas más importantes y frecuentes del sistema y los datos que necesitarían para realizar dichas tareas. Luego, se deben agrupar los datos comunes dentro de unas pocas ventanas. Si una tarea requiere una ventana de contexto ya identificada, simplemente se reutiliza dicha ventana.

Por lo tanto, se hace primero un plan de lo que debería estar en cada ventana, y luego se hace un diseño gráfico detallado de las mismas, en esta fase se preparan prototipos o formularios en papel para describir este tipo de ventanas, en las que se muestran los datos, pero que aún no contienen botones, menús u otras funciones. Por último, se evalúa a los usuarios para determinar su nivel de comprensión y aceptación de dichas ventanas, y se comparan con la descripción de las tareas y el modelo de datos para que todo esté cubierto. Este proceso se repite hasta abarcar todas las tareas del sistema.

Para que sea más eficaz el diseño de las ventanas de contexto y la forma de combinarlas cuando se soporta una tarea, se deben seguir una serie de normas. Algunas de estar normas son las siguientes (Lauesen, 2005):

- El número de ventanas debe ser el menor posible.
- Evitar que el usuario pueda ver o editar el mismo dato en varias ventanas.
- Una ventana debe estar estructurada en torno a un objeto principal y sus objetos relacionados.
- La ventana debe caber en la pantalla.
- La ventana debe ofrecer una visión general de muchos datos (las personas entienden mejor cuando las cosas están dentro de un contexto).
- La ventana debe tener nombre de cosas y no de acciones para no limitar el uso de dicha ventana a una tarea. Se propone usar nombres en plural para indicar listados de entidades.

3.4 DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO

En esta sección se describe el proceso específico para el modelado de la interfaz de usuario de negocio de acuerdo a la metodología TD-MBUID.

En la Figura 8, se observa el desarrollo de la metodología del desarrollo de la Interfaz de usuario de negocio propuesta en (Orozco, 2010), en ella se puede observar que con base a la captura de los modelos mentales

de las tareas y los datos que tiene el usuario, se realiza el diseño inicial de la interfaz de usuario de negocio, la cual representa un diseño inicial de la interfaz de usuario que podría contener la forma (Layout, esqueleto) y la semántica de los datos.

También se puede apreciar que todas las tareas de diseño son supervisadas por el equipo desarrolladores, es decir todas las tareas de diseño son realizadas y supervisadas por los desarrolladores, al mismo tiempo van realizando los diferentes componentes (Paper Form, Data Form, Business Domain Data) para que sean evaluados y construidos con base a las especificaciones del usuario y en forma iterativa (Prototipado).

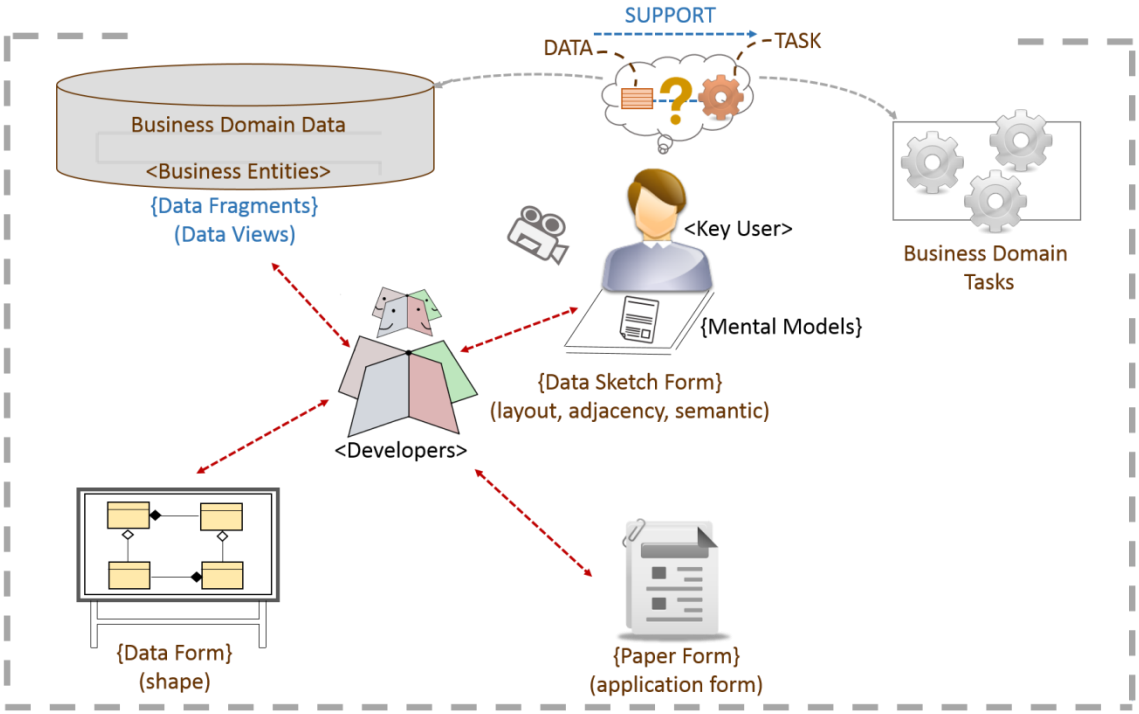

Figura 8 Desarrollo de la Interfaz de Usuario de Negocio(Orozco, 2010)

3.4.1 componentes de la interfaz de usuario de negocio

Como se mencionó en la sección anterior los componentes que conforman el desarrollo de la interfaz de usuario de negocio son el Business Domain Data, el Paper Form y el Data Form, a continuación, se describe cada uno de ellos.

3.4.1.1 Business Domain Data

Este componente está conformado por las entidades de negocio, que representan los datos que soportan las tareas del negocio, estos son capturados por el desarrollador a partir de la interpretación de los

modelos mentales que el usuario tiene de cada tarea del dominio, los modelos mentales es la forma como el individuo estructura, organiza y entiende la información. El usuario es quien conoce los datos que necesita, sin tener claridad de que ellos conforman o forman parte de una o varias entidades de negocio, esto último es habilidad y responsabilidad del desarrollador quien es el experto en modelado de datos y por lo tanto conoce e identifica estos conceptos. Este componente, aunque se basa en el modelo mental del usuario, se determina desde la perspectiva del desarrollador.

3.4.1.2 **Paper Form**

En TD-MBUID la presentación de la interfaz de usuario está dada por la forma, la cual es capturada por el Paper Form, el cual describe la estructura de la presentación, ligada a la semántica de los datos, muestra un diseño visual de manera realista de los Business Domain data que soportan la tarea, la renderización de este componente constituye la interfaz de usuario de negocio. Este componente se determina desde la perspectiva del usuario, ya que se busca representar solo las entidades y la porción de los datos que el usuario necesita de ellas para realizar la tarea.

3.4.1.3 **Data Form**

Por cada Paper Form se debe diseñar una estructura de datos llamada Data Form la cual describa la forma y que actúa como la persistencia de los datos del Paper Form, es decir, le brinda el almacenamiento de los datos necesarios para el uso de la interfaz. Este componente se explicará con mayor profundidad en el siguiente capítulo.

3.5 DATAFORM

Este componente es el encargado de darle persistencia a la forma (interfaz de usuario de negocio) y es una parte fundamental en el desarrollo de interfaz de usuario de negocio TD-MBUID. Debido a la importancia de los datos en el desarrollo de la interfaz de usuario en TD-MBUID, se requiere un componente donde se especifiquen las relaciones de contención y adyacencia entre los datos y los elementos de la forma involucrados en la interfaz, es decir un modelo que sea independiente al modelo de dominio, ya que el dominio se compone de toda la información del negocio y de datos que quizás no sean necesarios en la interfaz de usuario, este modelo se define con el nombre de DataForm (Datos y Forma), en él se captura la semántica de los datos y la semántica de la forma.

El DataForm se convierte en una pieza clave en el modelado de la interfaz de usuario ya que podemos tener fragmentos de los datos que serán de interés para el usuario y serán mostrados en la interfaz, ya

que es un modelo de información de contexto que representa los datos persistentes en la interfaz que serán mostrados al usuario y es muy útil para la especificación de la información que será visualizada y manipulada en la interfaz de usuario independiente de la manera en cómo es utilizada en el dominio.

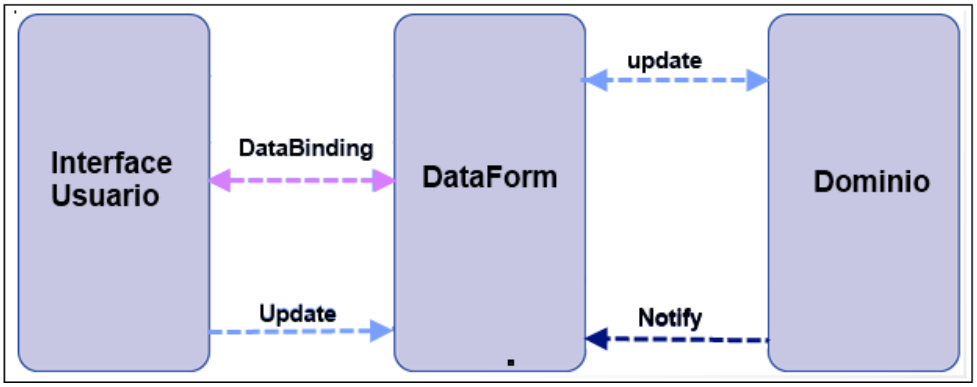

Figura 9 Esquema de Comunicación del Data Form.

Como se puede observar en Figura 9, el DataForm es el intermediario entre la interfaz y el modelo de dominio y se centra en recibir peticiones o notificaciones de la interfaz y de obtener los datos del dominio y generar actualizaciones en la interfaz de acuerdo a la especificación de DataForm que se haya realizado con anterioridad.(Muñoz & Orozco, 2014)

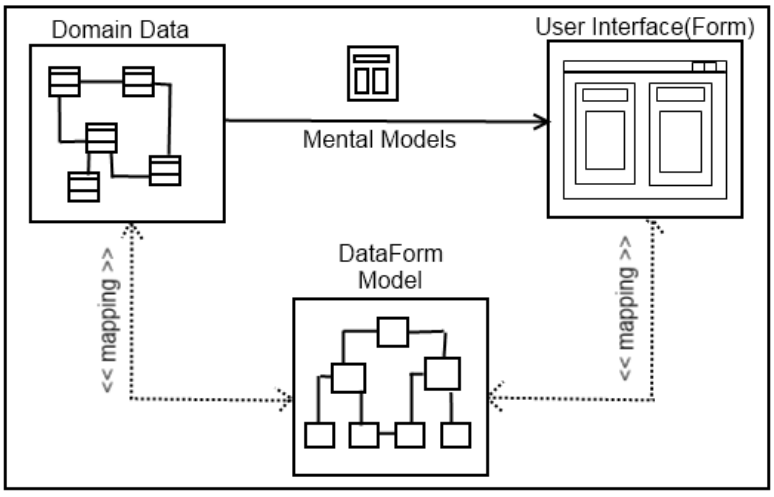

Figura 10 Componentes intervienen en la creación del Modelo Data Form(Muñoz & Orozco, 2014)

En la Figura 10 Componentes intervienen en la creación del Modelo Data Form(Muñoz & Orozco, 2014)Figura 10, se observa los componentes que intervienen en la creación de un modelo de DataForm: *Domain data* es el componente donde se encuentra todos los datos del negocio la cual es capturada por el diseñador o desarrollador, El *User Interface*, representan la interfaz en papel de acuerdo a la visión que el usuario tenga del negocio(Mental Models), de cómo percibe y estructura la información; y por otro lado el componente DataForm que se crea haciendo un proceso de mapeo (mapping) haciendo uso del lenguaje del DataForm de acuerdo a la interfaz de usuario y los datos provenientes del dominio que serán visibles en la interfaz de usuario, en la Figura 11 se puede observar un diagrama del modelo DataForm, el lenguaje del DataForm será expuesto en el capítulo 7.3 Lenguaje Data Form.

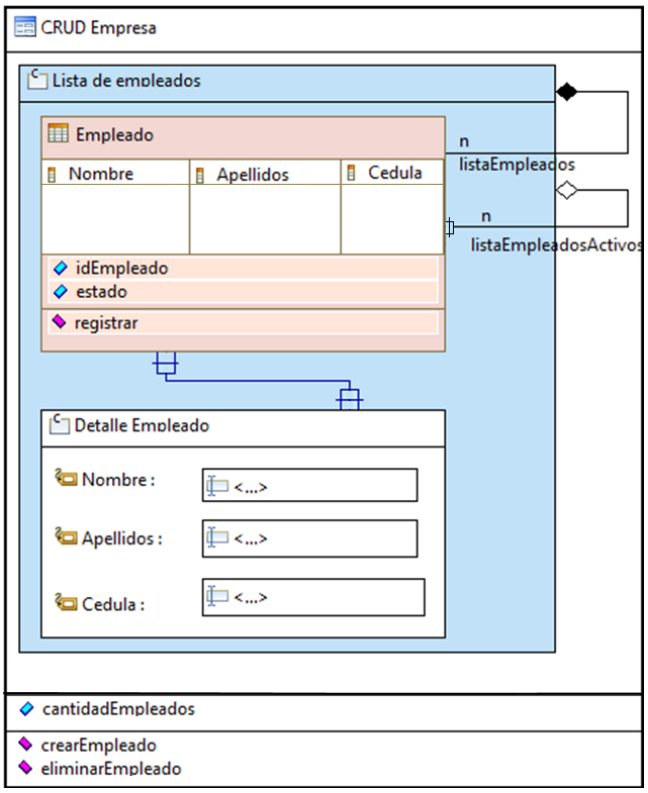

Figura 11 Diagrama de un Modelo Data Form

La generación automática de la interfaz de usuario a partir del modelo Data Form se rige por los principios del patrón de Diseño model – view –viewmodel que desacoplar el código de la interfaz de usuario del resto del código de la aplicación. En la siguiente sección se indicará cómo adoptaron este patrón en el desarrollo de la interfaz de usuario a partir de este modelo.

3.5.1 Patrón model – view - view Model

Para el desarrollo de la interfaz de usuario basado en el modelo DataForm adoptaron el patrón MVVM (Model View ViewModel). Este patrón de diseño propuesto por Microsoft, proporcionar una separación entre los controles de la interfaz de usuario y su lógica. EL patrón está compuesto por tres elementos principales: el modelo, la vista y el modelo de vista, en la Figura 12 se presenta la relación entre estos componentes(Microsoft, 2012).

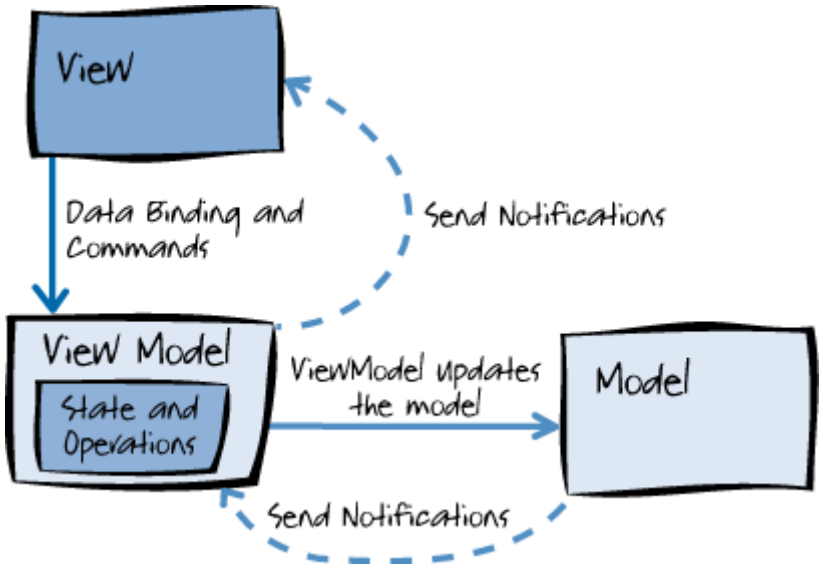

Figura 12 Estructura del Patrón MVVM(Microsoft, 2012)

Cada componente del patrón tiene una función y responsabilidad diferente, a continuación, se describen cada uno de ellos:

Modelo (La capa de modelo) incluye todo el código que implementa la lógica principal de la aplicación y define los tipos requeridos para modelar el dominio de la aplicación. Incluye el modelo de datos junto con las validaciones del negocio y de la lógica. Esta capa es completamente independiente de las capas de vista y modelo de vista.

View (La capa de vista): La vista es responsable de definir la estructura, el diseño y el aspecto de lo que el usuario ve en la pantalla. La vista no contiene la lógica de negocio, en ella se definen la conexión entre componentes específicos de la interfaz de usuario y diversos miembros de modelo lo que permite que, al realizarse un evento en la vista, como un clic en un botón, se ejecuta el código del modelo de la vista.

ViewModel (La capa de modelo de vista) El modelo de vista actúa como intermediario entre la vista y el modelo, es responsable de manejar la lógica de la vista. Típicamente, el modelo de vista interactúa con el modelo por invocaciones de métodos de las clases del modelo. El modelo de vista a continuación ofrece datos del modelo en una forma en que se puedan utilizar fácilmente por la vista. El modelo de vista recupera los datos del modelo y luego hace que los datos sean disponibles a la vista. El modelo de vista también puede definir miembros para realizar un seguimiento de los datos que son relevantes para la interfaz de usuario, pero no para el modelo, como el orden de visualización de una lista de elementos.

En el proceso de desarrollo de la interfaz de usuario es común que del modelo mental que tiene un usuario se generen diferentes modelos de los datos que son requeridos en la interfaz, esto debido al conocimiento y subjetividad del desarrollador, La naturaleza del patrón Model - View - ViewModel soporta esta situación. Este patrón soporta que a partir de un formulario en papel se puedan obtener diferentes modelos de vista (ViewModel) para una sola interfaz. La adopción de este patrón buscó que a partir de un solo modelo (DataForm) se pueda generar la interfaz (View) y el modelo de la vista (ViewModel).

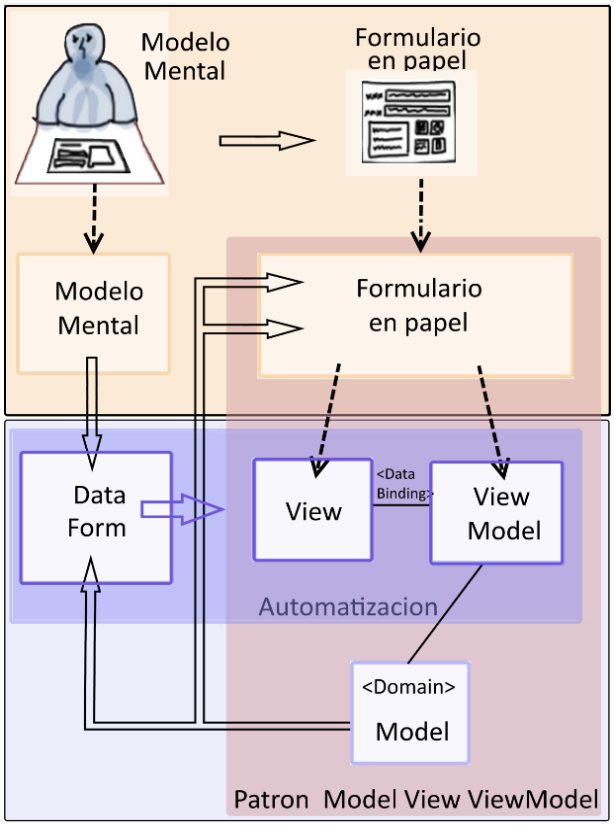

Figura 13 Adopcion del patron(Muñoz & Orozco, 2014)

El mapping entre el patrón MVVM (Model View ViewModel) y los elementos de la metodología de desarrollo de la interfaz de usuario de negocio (descrito en la sección anterior) se presenta en la Figura 13. En la metodología TD-MBUID la presentación de la interfaz de usuario inicia desde los modelos mentales de los cuales se deriva los formularos en papel, luego continua con el prototipado de las interfaces de usuario, aquí es donde la herramienta apoyaría al desarrollador o diseñador en la generación automática de estos prototipos de manera rápida a partir de los modelos DataForm.

CAPÍTULO CUARTO

4. FUNDAMENTOS DE PATRONES

En este capítulo se introduce a la teoría general del concepto de patrones, se presenta también la colección de patrones propuesta en esta tesis, la cual asocia por cada patrón opciones para la presentación de sus datos.

Se ha decidido hacer uso de patrones para plasmar el modelo mental que el usuario tiene de sus datos y para generar una propuesta rápida de interfaz de usuario de negocio debido a que su uso logra ocultar al analista gran parte de la complejidad subyacente.

4.1 DEFINICIÓN

Un patrón es un modelo que representa una solución recurrente a un dominio de problema específico, es decir, describe y especifica cómo se puede solucionar un problema en particular.

En relación a este tema podemos encontrar muchas definiciones:

- Un patrón es un fragmento de modelo que es profundo y recurrente. (Blaha, 2010)
- Un patrón resuelve un problema en un contexto. (Alexander, 1979)
- Un patrón de arquitectura de software describe un problema de diseño recurrente particular que surge en contextos específicos de diseño, y presenta un esquema genérico eficaz y probado para su solución.(Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 2013)
- Un patrón es una plantilla de un ejemplo digno de emular, de algo observado en la realidad. Es una plantilla hacia una solución, no una solución como tal. (Peter Coad & Mayfield, 1994)
- Un patrón es conocimiento recopilado sobre una determinada actividad (Hay, 2010).
- Un patrón ofrece una solución probada a un problema común documentado individualmente en un formato consistente; por lo general forma parte de una gran colección (Erl, 2009).
- Un patrón recoge cómo pasar del que al cómo, es una lingua franca (Erickson, 2000) que puede ser utilizada por todos aquellos que componen el equipo de desarrollo de un producto software, incluyendo al usuario, así mismo, son una forma de documentación algo más precisa de la experiencia adquirida (Simarro, 2005).

- Cada patrón es una regla compuesta por tres partes, que expresan una relación entre un cierto contexto, un problema y una solución (Alexander, 1979):

Como **elemento en el mundo**, cada patrón es una relación entre un cierto contexto, un cierto sistema de fuerzas que ocurren repetidamente en ese contexto, y una cierta configuración espacial que permite resolver las fuerzas por sí mismas.

Como **elemento del lenguaje**, un patrón es una instrucción que muestra cómo la configuración espacial debe ser usada una y otra vez, para resolver el sistema de fuerzas dado, siempre que el contexto sea relevante.

El patrón es, en resumen, al mismo tiempo una entidad, que ocurre en el mundo y una regla que nos dice cómo crear esa entidad, y cuándo debemos crearla. Es a la vez un proceso y una entidad: la descripción de una entidad que está viva y la descripción del proceso que genera esta entidad.

De acuerdo a las definiciones mencionadas, podemos concluir aspectos comunes entre los diferentes autores: primero, que un patrón ofrece una forma de solucionar un problema común (repetitivo) en particular; segundo, que representa un conocimiento adquirido por la experiencia y tercero, que esta solución debe estar documentada de tal manera que sirva de guía para aplicar la solución planteada.

Por esto, el proceso de identificación o especificación de patrones requiere de grandes dosis de observación, para encontrar semejanzas y de abstracción, para descartar detalles irrelevantes y describir con precisión la esencia del problema. De este modo, el proceso de identificación de patrones puede verse como un tipo de ciencia empírica y de abstracción, en el cual a partir de los resultados se intenta obtener leyes que los rijan. En aras de disminuir la complejidad y duración de este proceso, un desarrollador (el caso de esta tesis), puede soportarse en patrones que han sido previamente identificados por autores reconocidos en este tema, como (Blaha, 2010; Eriksson & Penker, 1998; Hay, 2010; Silverston & Agnew, 2009; Toxboe, 2014; Vanderdonckt & Simarro, 2010; Welie, 2008), los cuales presentan colecciones de patrones de diseño, presentación y conceptuales.

Según (Moreno, 2002), el uso de patrones supone una ciencia de carácter estadístico y empírico, puesto que los patrones, por sí mismos y de modo aislado, no resuelven todos los problemas posibles; sin

embargo, una colección de patrones cuidadosamente escogidos puede caracterizar del orden de 80% de los pares problemas/soluciones para un dominio dado.

Dentro de esta investigación se pretende proponer un catálogo de patrones de negocios con posibles variabilidades y asociados a formas de presentación, que permita solucionar problemas de captura del modelo mental que los usuarios tienen de los datos de su contexto de negocio.

4.2 CUALIDADES DE LOS PATRONES

Un patrón bien descrito debe exhibir las siguientes cualidades deseables, según Lea (Lea, 1994):

Encapsulación y abstracción: Cada patrón encapsula un problema bien definido y su solución en un dominio particular. Los patrones deben proporcionar fronteras claras que ayuden a separar claramente el espacio del problema del espacio de la solución parcelando el patrón en fragmentos interconectados.

Apertura y Variabilidad: Los patrones deberían ser abiertos para soportar extensiones o parametrización por parte de otros patrones de modo que puedan trabajar juntos para resolver problemas mayores. La solución de un patrón debe ser capaz de ser realizada mediante diferentes implementaciones.

Generabilidad y Composicionabilidad: Un patrón una vez aplicado genera un contexto resultante que encaja con el contexto inicial de uno o más patrones en un lenguaje de patrones. Los subsiguientes patrones se aplican para, poco a poco, alcanzar la solución completa.

Equilibrio: Por último, cada patrón debe realizar algún tipo de balanceo entre las fuerzas y restricciones involucradas. Puede deberse a que hay invariantes o heurísticos que son empleados para minimizar el conflicto en el espacio de la solución. Los invariantes a menudo tipifican problemas fundamentales que pueden ser resueltos mediante un principio de resolución para el dominio particular.

4.3 CLASIFICACIÓN DE PATRONES.

En la teoría relacionada con patrones se encuentran diversas formas de clasificarlos, por ejemplo, de acuerdo a sus características (su finalidad o su nivel de abstracción). En el contexto del desarrollo de software se encuentra la taxonomía propuesta por (Sarver, 2000):

4.3.1 Patrones arquitectónicos.

Expresan estructuras de organización de sistemas de información. Proporcionan un conjunto de subsistemas predefinidos donde se especifican sus responsabilidades y se incluyen reglas y guías para organizar las relaciones entre los subsistemas, por ejemplo, la arquitectura tres capas o cliente/servidor.

4.3.2 Patrones de diseño.

Un patrón de diseño describe una estructura recurrente de componentes que se comunican; resuelve un problema general de diseño en un contexto particular y ofrece soluciones descritas, independientes de cualquier lenguaje de programación. Para un diseñador de sistemas, estos son una fuente de conocimiento que no debe ser despreciada, ya que contienen la experiencia de otros diseñadores de sistemas sobre problemas que aparecen una y otra vez.

4.3.3 Patrones de programación, Idioms.

Un patrón de programación o idiom es un patrón de bajo nivel, específico para un lenguaje de programación. Describe cómo implementar aspectos particulares de componentes o sus relaciones entre ellos usando las características particulares del lenguaje de programación específico y por lo tanto un patrón idiom no puede ser reusado en otro lenguaje.

(Riehle, Z, & llighoven, 1996), proponen la siguiente clasificación:

4.3.4 Patrones conceptuales.

Es un patrón que se describe en términos y conceptos de un dominio de aplicación (dominio del problema).

4.3.5 Patrones de diseño.

Un patrón de diseño es un patrón descrito en términos de construcciones de diseño como, por ejemplo, objetos, clases, herencia, agregación y relaciones de uso.

4.3.6 Patrones de programación.

Descritos en términos de construcciones de un lenguaje de programación.

En (Eriksson & Penker, 1998), presentan una clasificación que incluye además de los patrones de diseño y patrones arquitectónicos, un nuevo concepto:

4.3.7 Patrones de negocio.

Direccionan la solución a problemas en el dominio del negocio -situaciones de análisis-: cómo modelar y estructurar recursos y conceptos, que incluyen facturas, organización e información. También se ocupan de cómo organizar y relacionar los procesos, reglas de negocio, visiones corporativas y objetivos.

4.3.8 Patrones de modelado de datos.

Son guías reutilizables que proporcionan una solución a problemas de modelado de conceptos o temas frecuente o universales que se presentan en el modelado de datos. Buscan ser una plantilla que sirva como una guía para el desarrollo de modelos de datos.(Silverston & Agnew, 2009)

4.4 ADOPCIÓN DE LOS PATRONES A APLICAR EN ESTA TESIS

Una vez establecido el concepto de patrón y presentadas sus potenciales clasificaciones, se procede a adoptar los patrones a utilizar de esta tesis a partir de dos categorías:

> Categoría de Patrón de negocio: porque están enfocados a términos de dominio de negocios, buscan modelar los conceptos y recursos a nivel del negocio.
>
> Categoría de patrón de Modelado de datos: están enfocados a representar soluciones de cómo se deben modelar conceptos, representan las entidades y relaciones que lo conforman.

De las categorías anteriores se propone clasificarlas a ambas como una categoría única denominada ***categoría de patrón de negocio,*** esto a razón del nivel de abstracción en que se enfoca esta tesis, el modelado de negocio, en donde se describen los conceptos y datos que soportan las tareas naturales del negocio independiente de la tecnología. En este sentido, con estos patrones se busca especificar soluciones de modelamiento que faciliten interpretar el modelo mental que el usuario tiene de sus datos

ayudando así la realización del *Diseño del dominio de negocio* y *del Modelado de la interfaz de usuario de negocio*.

4.5 CATÁLOGO DE PATRONES

Un catálogo es un conjunto de patrones relacionados entre sí y clasificados por algún criterio, los cuales pueden ser utilizados conjuntamente o de manera independiente. La finalidad de clasificar los patrones es organizarlos en conjuntos con características similares, es decir, grupos que tienen en común un cumulo de propiedades. De acuerdo a los criterios definidos para agrupar los patrones, se pueden definir esquemas de clasificación con más de una dimensión. La finalidad de esta clasificación es facilitar la búsqueda de los patrones.

En la literatura se pueden encontrar los siguientes criterios utilizados para clasificar diferentes tipos de patrones(Duda, Hart, & Stork, 2000):

Por su dominio. Dentro del campo de la ingeniería del software, los patrones no están restringidos a resolver problemas de diseño o de arquitectura software. Considerando el problema del diseño, se han propuesto gran cantidad de patrones en diferentes áreas, como pueden ser los de sistemas distribuidos, concurrentes o paralelos (Schmidt, Stal, Rohnert, & Buschmann, 2013) , los de presentaciones electrónicas (Roth & Schulz, 2015)

Por su paradigma. Cada paradigma tiene sus propios patrones. Un patrón de diseño imperativo (uno de los paradigmas de la programación) podría dirigir un problema similar que un patrón de diseño orientado a objetos, pero la solución se describiría en términos de cada paradigma. Un ejemplo de este tipo sería el patrón Decorator (E. Gamma, Helm, Johnson, & Vlissides, 1994) que es presentado según el paradigma de la orientación a objetos y el patrón Navigational Context (Rossi, Schwabe, & Garrido, 1997) que es el mismo patrón pero adaptado al paradigma del hipertexto.

Por su granularidad. Los patrones también pueden ser clasificados dependiendo del nivel al cual se dirijan. Un ejemplo de esta clasificación es la descrita en (Buschmann et al., 2013), donde en el nivel más bajo se encuentran los patrones específicos del lenguaje de programación, en el nivel medio se ubican los de diseño y en el nivel superior los patrones referentes a la arquitectura del sistema.

Por su nivel de abstracción. De una manera similar a la anterior, los patrones software se pueden clasificar de acuerdo al contexto o nivel de detalle en el que se represente una solución por ejemplo patrones conceptuales, patrones de diseño y patrones de programación, que estarían en el nivel más bajo de abstracción, cuyas soluciones son descritas mediante construcciones propias de los lenguajes de programación.

Por su propósito. El propósito representa qué tipo de problemas resuelve el patrón. Este criterio fue usado inicialmente por (E. Gamma et al., 1994), quienes establecieron las siguientes categorías: creación, estructurales y de comportamiento.

En este trabajo se propone un catálogo de patrones organizado de acuerdo a los clasificadores que ofrece la taxonomía de Zachman (Sowa & Zachman, 1992), en el capítulo 6.2 se expone el catálogo propuesto.

CAPÍTULO QUINTO

5. PROCESO METODOLÓGICO PARA LA DEFINICIÓN DEL MÉTODO PROPUESTO

En este capítulo se aplican los fundamentos descritos en los CAPÍTULOS 3 y 4 para definir el método de desarrollo de la interfaz que se trata en esta tesis, en forma de ejemplos se presenta paso a paso las tareas realizadas tomando como base el proceso TD-MBUID y el lenguaje DataForm. A partir de las tareas realizadas se construye el método propuesto y se definen los artefactos requeridos para aplicar dicho método.

La definición del método de desarrollo se centra en dos ejes principales, el primero con la identificación y definición de patrones de negocios y el segundo la aplicación de los patrones en el desarrollo de la interfaz de acuerdo al Marco Metodológico TD-MBUID quien orienta el proceso metodológico.

El desarrollo de la interfaz de usuario de negocio se fundamenta en el nivel de negocio de TD-MBUID y en su componente DataForm expuesto en la sección 3.5, asimismo en esta tesis, el diseño de la UI se realiza centrado en los datos, es por esto que se define o propone el uso de patrones de negocios, los cuales son descritos en la sección 6.2.

El proceso de concepción y elaboración del método propuesto tiene como resultado dos componentes principales: el *marco metodológico* representado por las propuestas metodológica y el *marco tecnológico* representado por las herramientas de soporte, para el marco conceptual se adopta el lenguaje DataForm, el cual es descrito en la sección 7.3.

5.1 DEFINICIÓN DE PATRONES

Como se ha mencionado, un patrón es un modelo que representa una solución recurrente a un dominio de problema específico, es decir, describe y especifica cómo se puede solucionar un problema en particular ya sea de modelado, implementación u otro nivel de abstracción(Schmidt et al., 2013).

De acuerdo al marco teórico expuesto en el capítulo 2, los patrones que se buscan en este trabajo son los relacionados a soluciones de modelado de datos y a partir de estos definir las plantillas de presentación (prototipo de interfaz) que soporte dichos datos.

5.1.1 Identificación de patrones

El proceso de identificación o especificación de patrones requiere de grandes dosis de observación y experiencia que permita encontrar semejanzas en diferentes soluciones de un mismo problema y a partir de esto lograr la abstracción de la esencia de la solución, para descartar detalles irrelevantes y describir con precisión la naturaleza del problema y de cómo solucionarlo. De este modo, el proceso de identificación de patrones puede verse como un tipo de ciencia empírica y de abstracción que depende de la experiencia obtenida en el desarrollo de proyectos (Jaquero, 2005), la cual permite encontrar conceptos comunes a múltiples contextos de problemas, formando a través de estos nuevos patrones de negocios.

Un desarrollador que inicia un proceso de identificación de patrones, en aras de disminuir la complejidad y duración de dicho proceso, puede soportarse en patrones o colecciones existentes definidos por autores de reconocida experiencia y conocimiento. En consecuencia, para la identificación de los patrones en este trabajo se utiliza como referencia la revisión del estado del arte, encontrando colecciones de patrones que, aunque tienen una clasificación distinta tienen en común que son orientadas a soluciones de modelado de datos. Para el proceso de identificación de patrones se encontraron las siguientes colecciones:

- Data Model Patterns: A Metadata Map: A Metadata Map *(Hay, 2010)*
- Patterns of Data Modeling (Blaha, 2010)
- The Data Model Resource Book: Universal Patterns for Data Modeling (Silverston & Agnew, 2009)
- Business Modeling With UML: Business Patterns at Work (Eriksson & Penker, 1998)

Como se puede observar en estas colecciones existen tres formas de clasificación de patrones; no obstante, todas están enfocadas a solucionar problemas de modelado de datos o conceptos, que es el interés en esta tesis. Es por esta razón que estas fuentes fueron seleccionadas y adaptadas para extraer los patrones relacionados a descripciones de conceptos de negocios que más adelante se describen en la sección 6.2.

Un ejemplo de patrón se presenta en la Figura 14, para este caso se trata del patrón Ubicación Geográfica cuya fuente es el libro Data Model Patterns: Conventions of Thought.

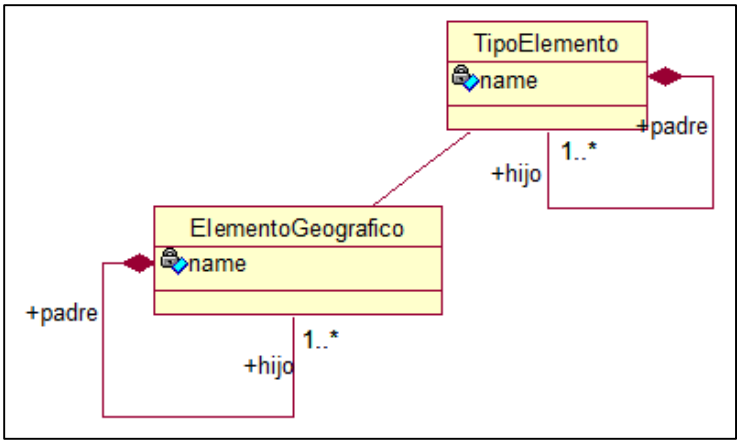

Figura 14 Patrón Ubicación Geográfica

Una vez identificadas las fuentes y seleccionados los patrones, se procede a identificar las variabilidades de interés para esta tesis.

5.1.2 Identificación de variabilidades

Un patrón describe la solución a un problema específico, pero puede ser adaptado, modificado o extendido según contextos particulares del mismo problema (Schmidt et al., 2013), en este sentido una cualidad importante que deben tener es la *Apertura y Variabilidad,* es decir, los patrones deberían ser abiertos para soportar extensiones o parametrización por parte de otros patrones de modo que puedan trabajar juntos para resolver problemas mayores. La solución de un patrón debe ser capaz de ser realizada mediante diferentes implementaciones(Lea, 1994).

Una vez identificado el patrón, se analiza su aplicación en diferentes contextos para determinar su variabilidad, es decir, se analiza los cambios a nivel de datos o relaciones que requiere el patrón para satisfacer las especificidades de cada contexto diferente, pero sin perder la esencia de la estructura del patrón en sí.

Para ilustrar este proceso se toma como base el patrón de Ubicación Geográfica ilustrado en la Figura 14, el cual tiene por objetivo modelar cualquier ubicación geopolítica en la tierra independiente de su constitución, ya que según la región o país la forma de división política cambia, este patrón puede generar producciones distintas según el contexto (variabilidades), por ejemplo, la ubicación en Colombia está dividida en 3 niveles de jerarquía (Figura 15.a) mientras que para definir una ubicación en España se requiere cuatro conceptos (Figura 15.b).

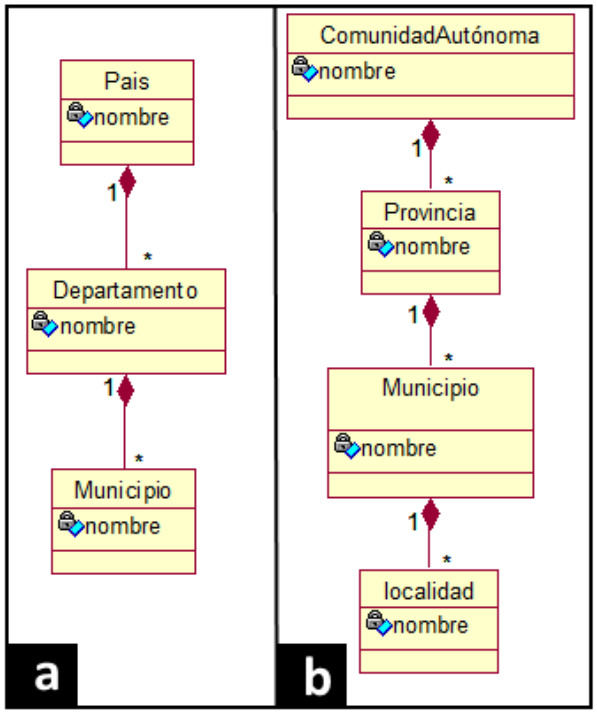

Figura 15 Ejemplo Variabilidades del patrón Ubicación Geográfica

Una vez definidas las variabilidades se procede a determinar con la ayuda de usuarios las plantillas de presentación, en la siguiente sección se describe el proceso utilizado.

5.1.3 Definición de presentaciones para los patrones de negocios

Una vez definidos los patrones y definidas las variabilidades a considerar por cada patrón que definen la estructura de los datos que se manipularán en la aplicación, se procede a definir la plantilla de presentación por medio de la cual un usuario podría capturar dichos datos. Esta tarea se debe realizar con el apoyo de usuarios, debido a que la forma y los datos presente en la presentación dependen de su modelo mental.

El método utilizado para definir las plantillas de presentación asociadas a cada patrón de negocio y sus respectivas variabilidades, ver Figura 16, se compone de tres pasos: 1) seleccionar el patrón de negocio; 2) Definir el formulario de papel, 3) definir la interacción

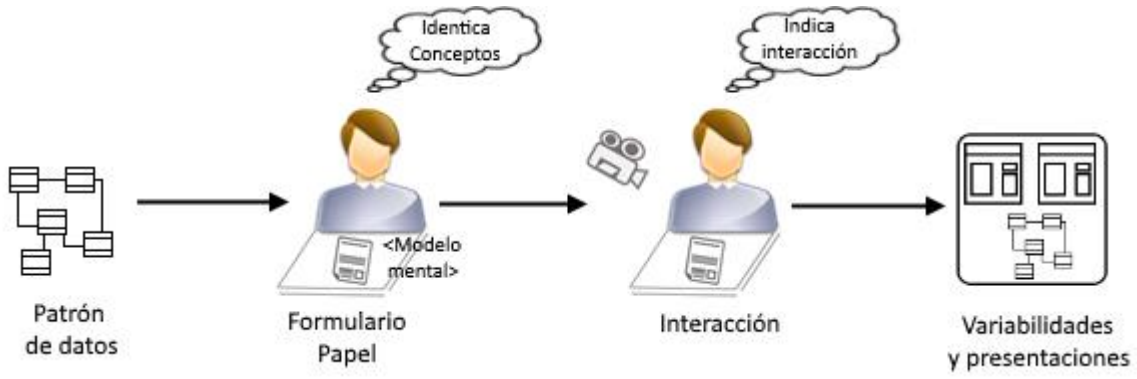

Figura 16 Proceso Identificacion de bosquejo de presentación de los patrones

Para ilustrar el proceso de definir las presentaciones de los patrones se selecciona como ejemplo el patrón Dirección (Figura 17) con su especialidad Dirección Postal, la cual se puede identificar por el color verde en la Figura 17.

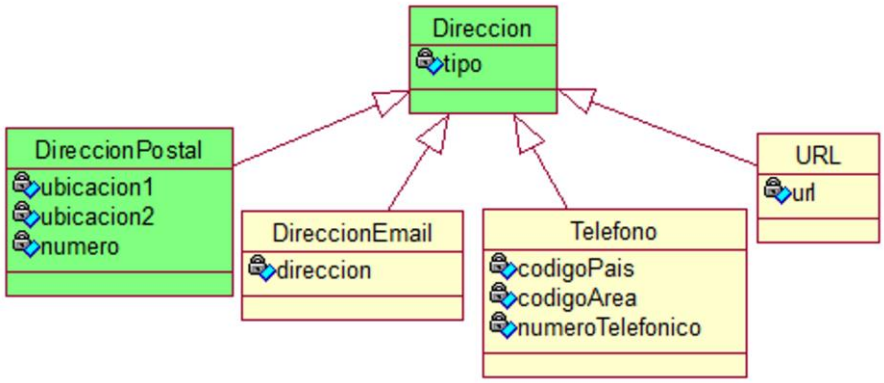

Figura 17 Patrón Dirección Variación Dirección Postal

Una vez definidas la variación del patrón se procede a definir la plantilla de presentación para visualizar los datos especificados en el patrón.

5.1.3.1 **Definición de Plantillas de Presentación**

Como ya se mencionó en la sección 3.4, la interfaz de usuario de negocio se realiza a partir del modelo mental del usuario quien conoce cuáles son los datos que necesita y como desea verlos e interactuar con ellos. En consecuencia, la definición del patrón o plantilla de presentación se define con ayuda de usuarios para comprender cuál debe ser la forma de la interfaz.

Siguiendo los fundamentos de TD-MBUID, la presentación de la interfaz de usuario de negocio está dada por los datos y la forma que es capturada por el Paper Form (Formulario en Papel). El Paper Form describe la estructura de la presentación, ligada a la semántica de los datos, y muestra un diseño visual de manera realista relacionado con los datos (Business Domain data) que soportan la tarea. En consecuencia, por cada variabilidad del patrón un diseñador visual interactúa con el usuario y genera una plantilla de presentación para visualizar los datos asociados al patrón y la interacción del prototipo de interfaz.

Para describir el proceso, tomaremos de ejemplo la tarea de definir una dirección postal para enviar una encomienda. A continuación, se presentan las acciones del método de creación de las plantillas de presentación aplicado a dos usuarios:

1. Describir una dirección postal, es decir, indicar cuáles son los datos y las relaciones entre esos datos que se necesitan para definir una dirección.

   Usuario 1: “la dirección, la ciudad, departamento y país”

   Usuario 2: “Una dirección está dada por una calle, carrera, numero de cada o apartamento y el nombre del barrio o conjunto residencial”.

2. Diseñar un formulario para escribir una dirección a partir de la información antes identificada.

   En este paso se lleva a cabo el diseño previo de las pantallas[5] que el usuario debe imaginar que están detrás de la pantalla física - en la parte posterior del sistema. Este diseño se lleva a cabo a partir de los datos del patrón y de los conceptos que el usuario identificó. A continuación, ilustraremos las plantillas generadas por los 2 usuarios.

   Usuario 1: El formulario resultante del proceso anterior se presenta en la Figura 18 mediante un Mockups[6].

[5] Se usa el término pantalla para referirse al modelo mental que se forma el usuario respecto de los datos que manipula. No se quiere utilizar el término ventana como componente concreto de interfaz de usuario.

[6] Modelo a escala o tamaño real de un diseño o un dispositivo, utilizado para la demostración, evaluación del diseño, promoción, y para otros fines. Un Mockup es un prototipo si proporciona al menos una parte de la funcionalidad de un sistema y permite pruebas del diseño

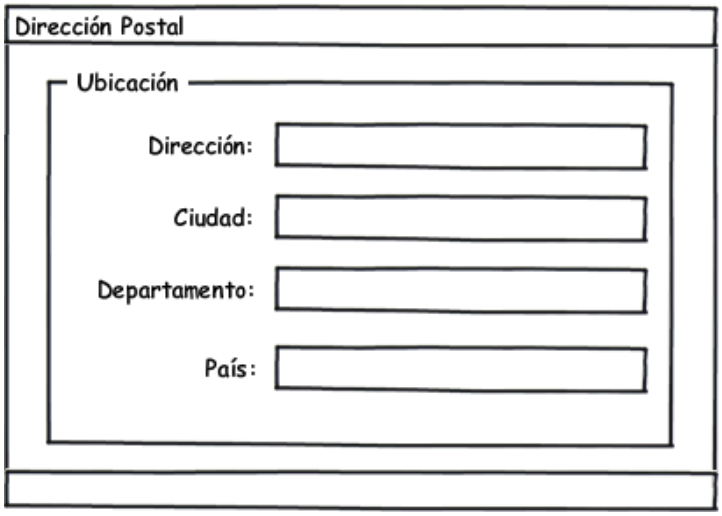

Figura 18 Mockups del formulario para una direccion postal definido por un usuario 1.

Usuario 2: El Mockups basado en el formulario definido por el usuario para soportar los conceptos que identificó para definir una dirección postal se muestra en la Figura 18 Mockups del formulario para una direccion postal definido por un usuario 1.Figura 19.

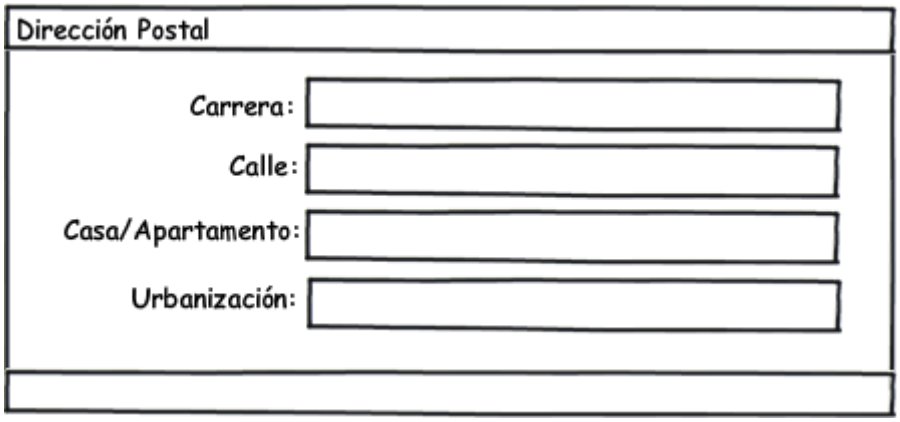

Figura 19 Mockups del formulario para una dirección postal definido por un usuario 3

3. Definir la interacción: Los usuarios dicen en voz alta como les gustaría interactuar con el formulario, es decir, qué acciones les gustaría que realizará la interfaz mientras ellos ingresan una dirección.

Aunque el objetivo del modelado de la interfaz de usuario de negocio es preparar prototipos, normalmente en papel, en las que se muestran los datos, pero que aún no contienen botones, menús u otras funciones, en esta propuesta se pretende facilitar las evaluaciones de las interfaces.

Estas evaluaciones pueden ser fructíferas si en los prototipos de las ventanas se ofrecen datos reales, detalles gráficos y de interacción que permitan al usuario validar si su modelo mental fue capturado de la manera adecuada, por esta razón en este paso se pretende identificar lo que el usuario espera que el sistema a medida que interactúa con él.

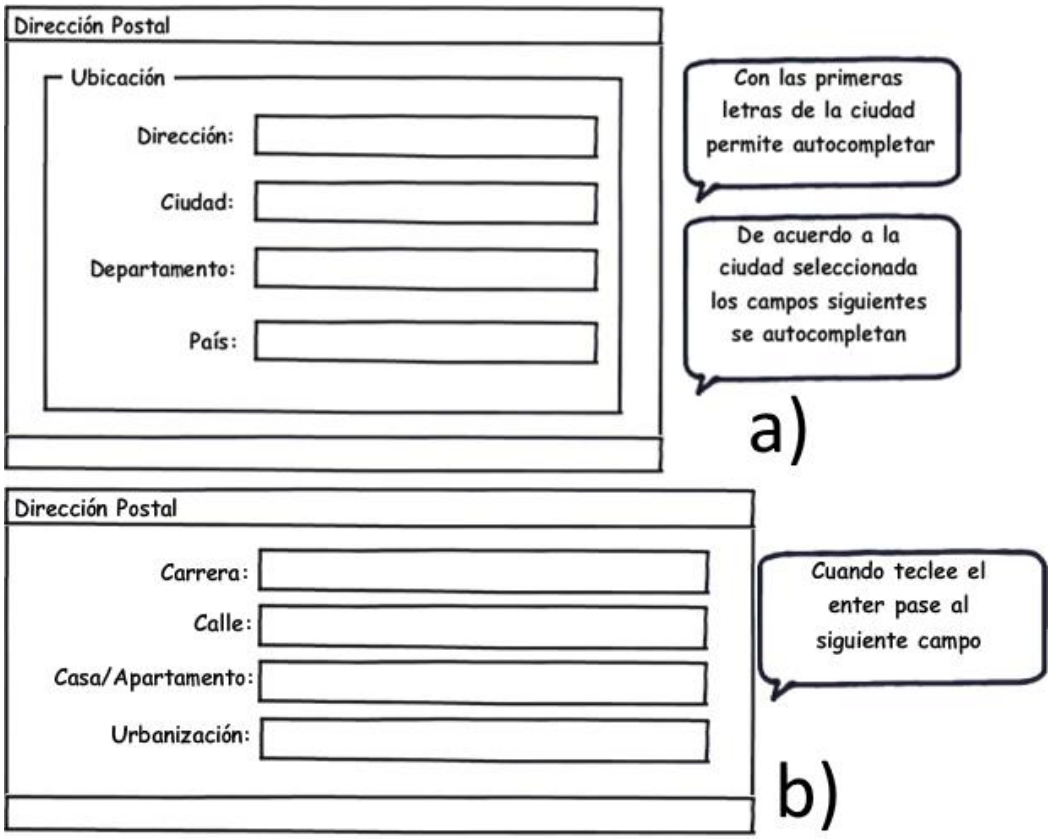

Figura 20 Descripcion interaccion deseada por los usuarios.

Para lograr esto, se le pide al usuario que interactúe con el formulario que definió y que exprese que le gustaría que el sistema realizara cuando lo está diligenciando, este paso genera un modelo simple de la interacción que permite generar un realismo mayor cercano a la interfaz final que el usuario se imagina que tendrá el sistema. En la Figura 20 se presenta la descripción de la interacción que cada uno de los usuarios que ejemplifican el proceso dan a sus respectivas plantillas de presentación (usuario 1 Figura 20.a y usuario 2 Figura 20.b).

Al finar este conjunto de paso, se analizan las plantillas generadas para identificar las variabilidades del patrón y sus respectivas presentaciones, es decir, se genera un conjunto de variabilidades con su respectivo subconjunto de prototipos de interfaz para cada patrón de negocio, por lo tanto, la variabilidad

del patrón se determina desde la perspectiva del desarrollador y las plantillas de presentaciones de cada variabilidad desde la perspectiva del usuario.

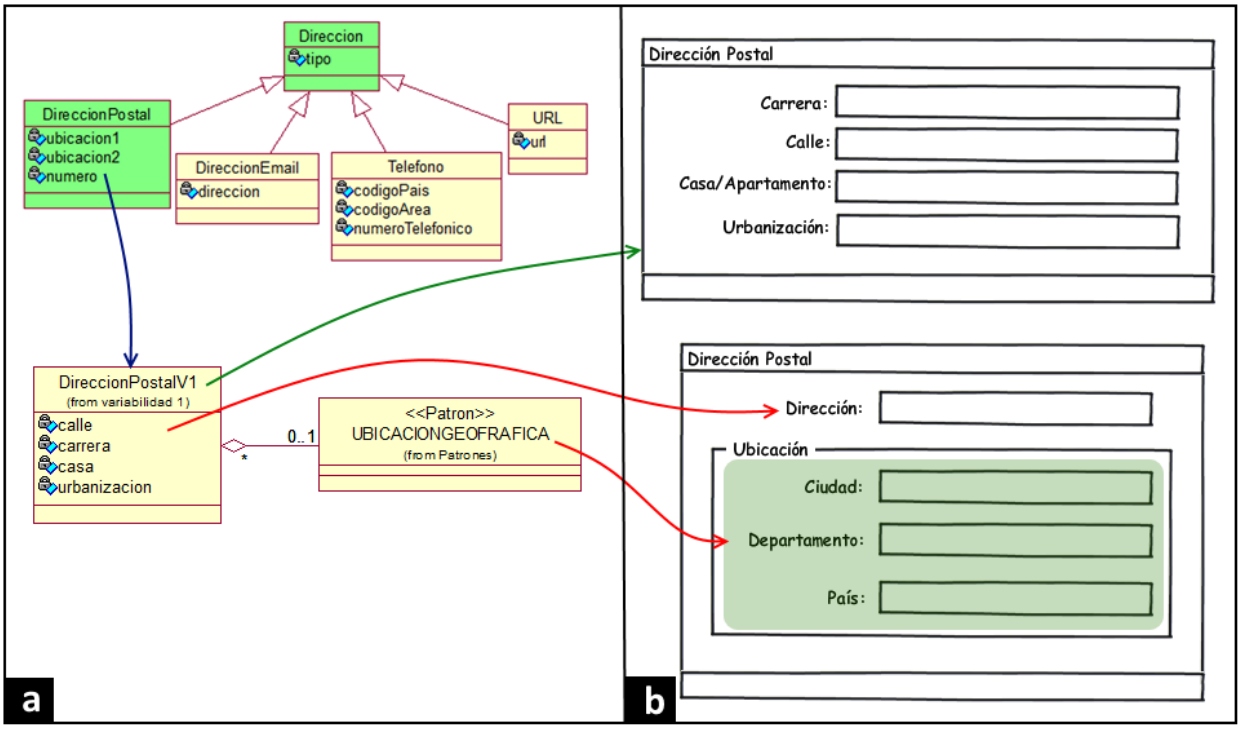

Figura 21 Variabilidad y presentaciones identificadas del patrón Dirección

Como se describió a lo largo de esta sección, se generaron dos plantillas de interfaz para la *dirección Postal* del patrón *Dirección* (Figura 18 y Figura 19). En la Figura 21.a, se presenta en la parte superior el patrón y en la parte inferior la variación del patrón a nivel de datos, en ella se puede observar que se incluye una asociación con cardinalidad cero a uno con el patrón *Ubicación Geográfica*, esto debido a que el desarrollador identifica con la propuesta del usuario 1 que se puede requerir al especificar una dirección. En la Figura 21.b se presentan las dos plantillas de presentación generadas por los usuarios asociadas a esta variabilidad, en la presentación mostrada en la parte inferior se puede observar (sombreado en verde claro) los datos que conforman la ubicación geográfica, por lo tanto se podría reutilizar una ventana de contexto del patrón ubicación geográfica.

5.1.4 Definición de la combinación entre patrones

Una vez definidas las variabilidades y presentaciones de cada patrón se examinan cada una de ellas para consolidar cuáles patrones y/o variabilidades se relacionan entre sí y por lo tanto qué presentaciones se

pueden reutilizar en el formulario de otro patrón. Así como, el caso expuesto entre dirección y ubicación geográfica, esto es fundamental para definir y restringir cuales combinaciones serán soportadas y evitar relaciones incoherentes entre los patrones.

Las combinaciones entre patrones surgen como resultado del proceso anterior y de la documentación de la fuente del patrón, en la cual indica con qué otros patrones se pueden relacionar. A partir de esta información se define de manera manual cuál es la combinación entre patrones de cada variabilidad, es decir, se indica cuáles patrones están directamente combinados y qué variabilidades de dicho patrón y presentaciones se puede usar, en la Tabla 1 se puede visualizar un ejemplo de lo indicado, las posibles combinaciones que soporta el patrón dirección Postal, en ella se puede distinguir a través de los distintos colores 4 variabilidades, asimismo que la variabilidad 2 y 4 se pueden combinar con una variabilidad del patrón Ubicación Geográfica.

Patrón Dirección	Variabilidad 1 Dirección Postal	Presentación (1 solo widget)			
		Presentación 2 (todos los campos)			
	Variabilidad 2 Dirección Postal con ubicación	Presentación 1 (solo widget)	PATRON Ubicación Geográfica	Variabilidad 1 Colombia	Presentación 1 (solo widget)
		Presentación 2 (todos los campos)			Presentación 2 (todos los campos)
	Variabilidad 3 Dirección Postal Europea	Presentación 1 solo widget			
		Presentación 2 todos los campos			
	Variabilidad 4 Dirección PostalEuropea con ubicación	Presentación 1 solo widget	PATRON Ubicación Geográfica	Variabilidad 2 Argentina	Presentación 1 (Todos los campos
		Presentación 2 todos los campos			Presentación 2 (1 campo)

Tabla 1 Ejemplo de combinaciones soportadas por el patron Dirección

5.1.5 Instancia de patrones.

Un patrón es una plantilla hacia una solución, no una solución como tal (Peter Coad & Mayfield, 1994), en consecuencia para implementar la solución que propone se deben crear instancias del patrón. Una instancia en el paradigma orientado a objetos, constituye la creación de objetos a partir de una clase, que permite definir el valor de los atributos y utilizar los métodos que en ella se definen.

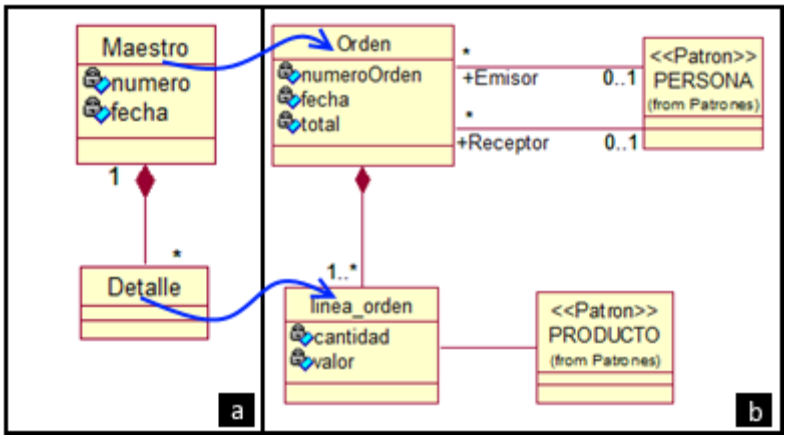

Figura 22 Instancia del Patrón

Los patrones son una especie de plantilla la cual se instancia para crear soluciones. Cada variabilidad definida constituye una instancia o producción que conserva las reglas y estructura definidas por el patrón, por ejemplo, en la Figura 22.a se presenta el patrón Maestro detalle, a partir del cual se define la variabilidad orden detalle (Figura 22.b), se puede observar a través de las líneas azules que se conserva la estructura base del patrón, aunque se adicionan elementos propios del contexto especifico de una orden comercial (como lo son Facturas, cotizaciones).

5.2 DEFINICIÓN DE DATAFORM PARA CADA PATRÓN

El DataForm dentro de TD-MBUID es un componente fundamental en el modelado de la interfaz de usuario, su importancia radica en el hecho de que separa los datos del dominio de los datos de la interfaz, puesto que podemos tener fragmentos de los datos del dominio que son de interés para el usuario y por lo tanto serán mostrados en la interfaz.

El DataForm representa la información que será visualizada y manipulada en la interfaz de usuario independiente de la manera en cómo es modelada en el dominio. Por lo tanto, un mismo dominio puede soportar múltiples presentaciones, siendo el DataForm quien funciona como intermedio entre estos componentes, además permite mantener una correcta relación e integridad entra los datos que se muestran provenientes del dominio de la aplicación (ver Figura 23).

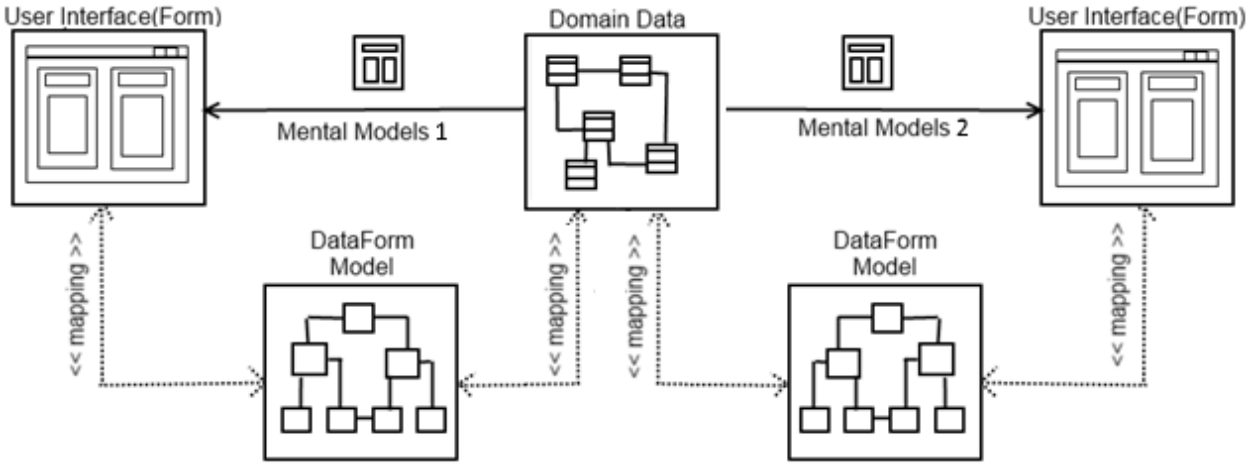

Figura 23 Soporte DataForm a múltiples Presentaciones de un domino

Esta facilidad es precisamente lo que se requiere para relacionar la variabilidad en datos de un patrón (viéndolo como el domain) y las distintas presentaciones que puede soportar (representando la Form), asimismo, permite adicionar las funcionalidades o porciones de lógica que permiten implementar la interacción que el usuario desea realizar en los formularios.

Otro factor importante en la adopción del DataForm es que su definición se realiza a partir de los formularios en papel definidos por el usuario (modelo mental). En consecuencia, basados en las variabilidades de los patrones y sus plantillas de presentación se define el modelo DataForm, el cual (basados en el patrón MVVM) permite generar de manera automática la view y el view model. En la Figura 24 se puede observar una view generada a partir del DataForm en donde se puede observar el mapeo entre sus elementos.

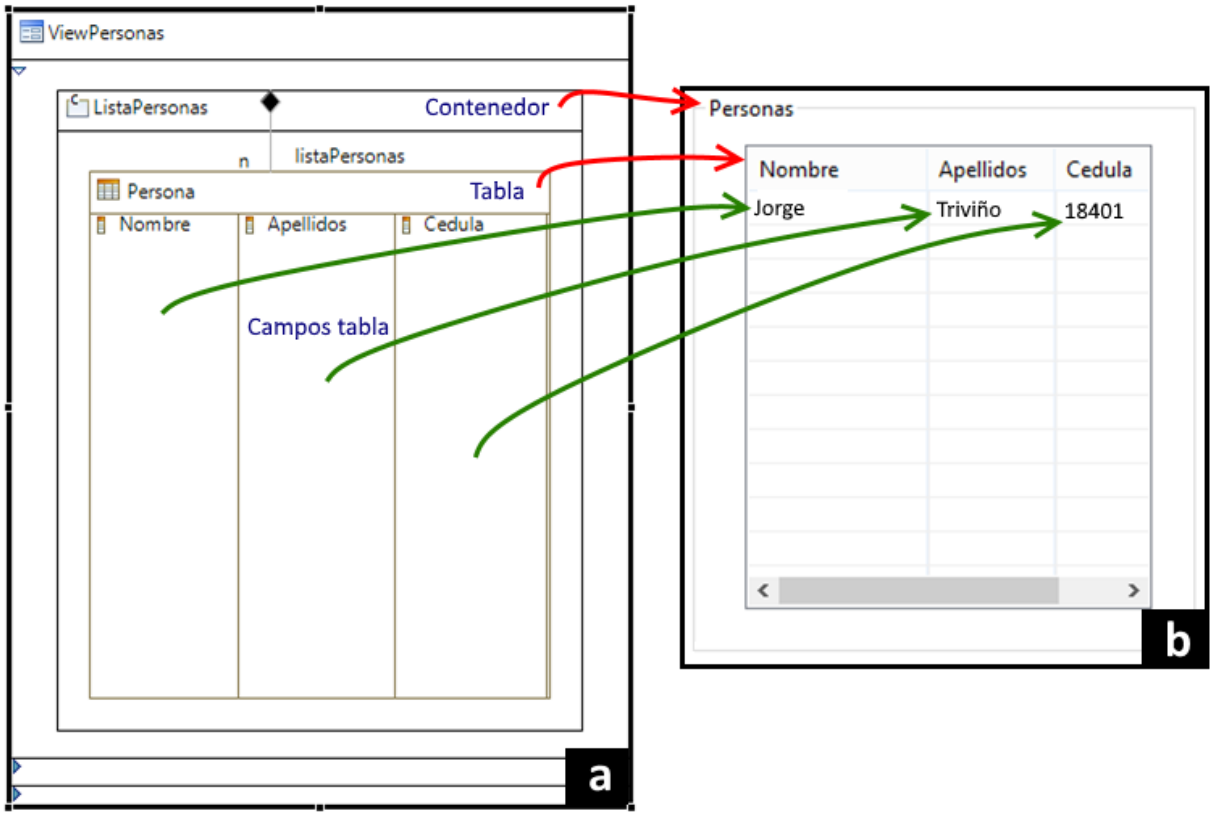

Figura 24 View generada a partir del DataForm Model

Una vez definido los patrones de negocios, su conjunto de variabilidades en datos y sus plantillas de presentación, se continua con la construcción de un modelo DataForm por cada una de las presentaciones del patrón donde se especifica los datos y la forma, así como la interacción, es decir, si una variabilidad de un patrón tiene 2 propuestas cada una de ellas se plasma en un DataForm.

Como se menciona en la sección 3.4.1.3, El DataForm se compone de tres elementos*: Domain data* que representa todos los datos del negocio la cual es capturada por el diseñador o desarrollador (equivalente al patrón de datos), El *User Interface (Form)*, representa la forma de la interfaz de acuerdo a la visión que el usuario de cómo percibe y estructura la información (equivalente a la plantilla de presentación); y por otro lado el componente DataForm quien representa los datos de la forma, es decir, los datos que se requiere para soportar la interfaz, brinda la persistencia de los datos que se muestran o manipulan en la interfaz gráfica, este último se construye a partir de un proceso de mapping.

En la Figura 25 se puede observar cómo se realiza el proceso para definir el modelo DataForm a partir de una variabilidad de un patrón de negocio (variabilidad en datos y su presentación), en donde el patrón

corresponde al componente del domain data, ya que representa los datos a nivel de negocio, la plantilla de presentación (Mockups) representa el modelo mental capturado del usuario y el modelo DataForm que se construye a partir de estos dos elementos, y a partir de este último se genera la view y el viewmodel que permiten la generación automática del prototipo de interfaz.

Para este ejemplo se toma como base la variabilidad del patrón Dirección Postal y una de las presentaciones mostradas en la sección anterior (Figura 19 y Figura 20.b)

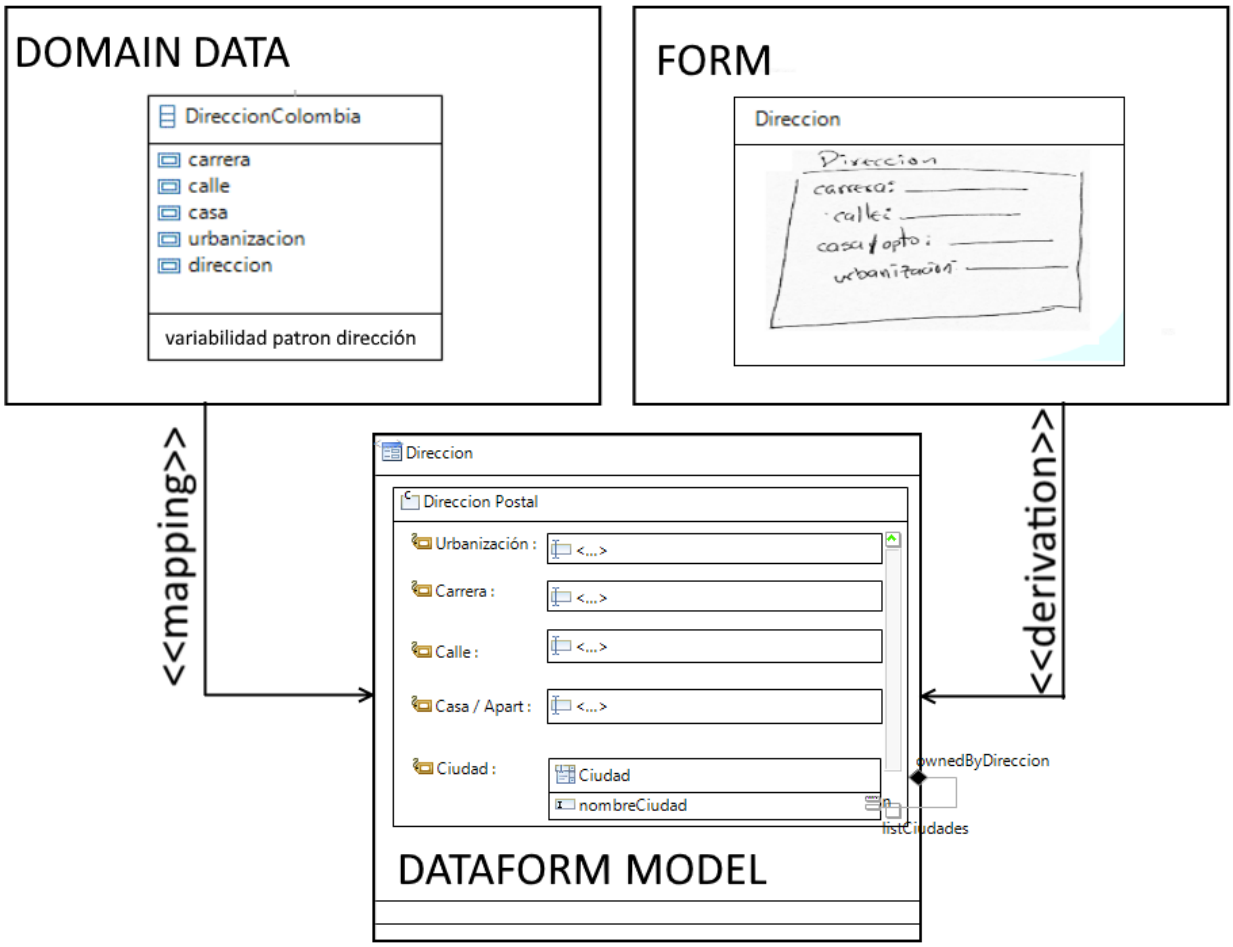

Figura 25 Proceso de contrucción del DataForm asociadoa un patron.

En conclusión, se debe realizar, tal como se puede apreciar en la Figura 23, un domain por cada variabilidad en datos de un patrón y un DataForm por cada presentación definida para esa variabilidad.

5.2.1 Construir del Domain de cada patrón

Este paso consiste en crear el componente *domain* del DataForm, como se indicó en la Figura 25, la variabilidad en datos de un patrón constituye este elemento. Por cada variabilidad se construye un modelo de dominio conformado por las clases, atributos y relaciones especificadas en ella. El dominio se construye en la herramienta toolDataForm a partir de diagramas de clases o producciones que representen este diagrama.

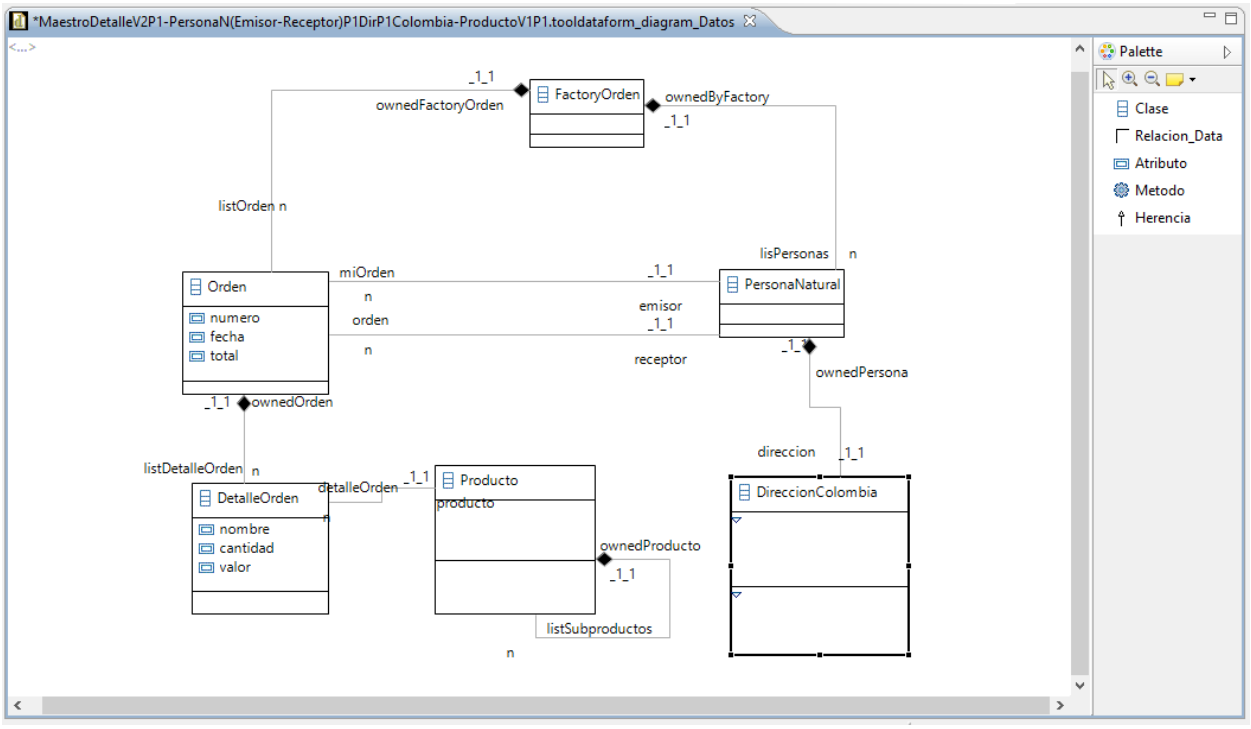

Figura 26 Domain creado para la variabilidad Factura con emisor y receptor.

En la Figura 26 se muestra un ejemplo de un domain creado, en ella se observa que del patrón se debe crear un Factory, el cual encapsula los elementos que lo componen y permite crear a partir de ellos producciones de instancias de las clases que lo conforman.

5.2.2 Construcción de los modelos DataForm basada en los Patrones

La elaboración del modelo DataForm se realiza haciendo uso de la herramienta toolDataForm, la cual fue definida por (Muñoz & Orozco, 2014) para brindar el soporte tecnológico al lenguaje DataForm, como se indicó por cada variabilidad del patrón se debe crear un modelo, en la Figura 27.a se pueden observar en el explorador del proyecto los distintos modelos DataForm (toolDataForm.diagram), en la Figura 27.b un ejemplo del modelo para la variabilidad factura con emisor y receptor del patrón orden detalle.

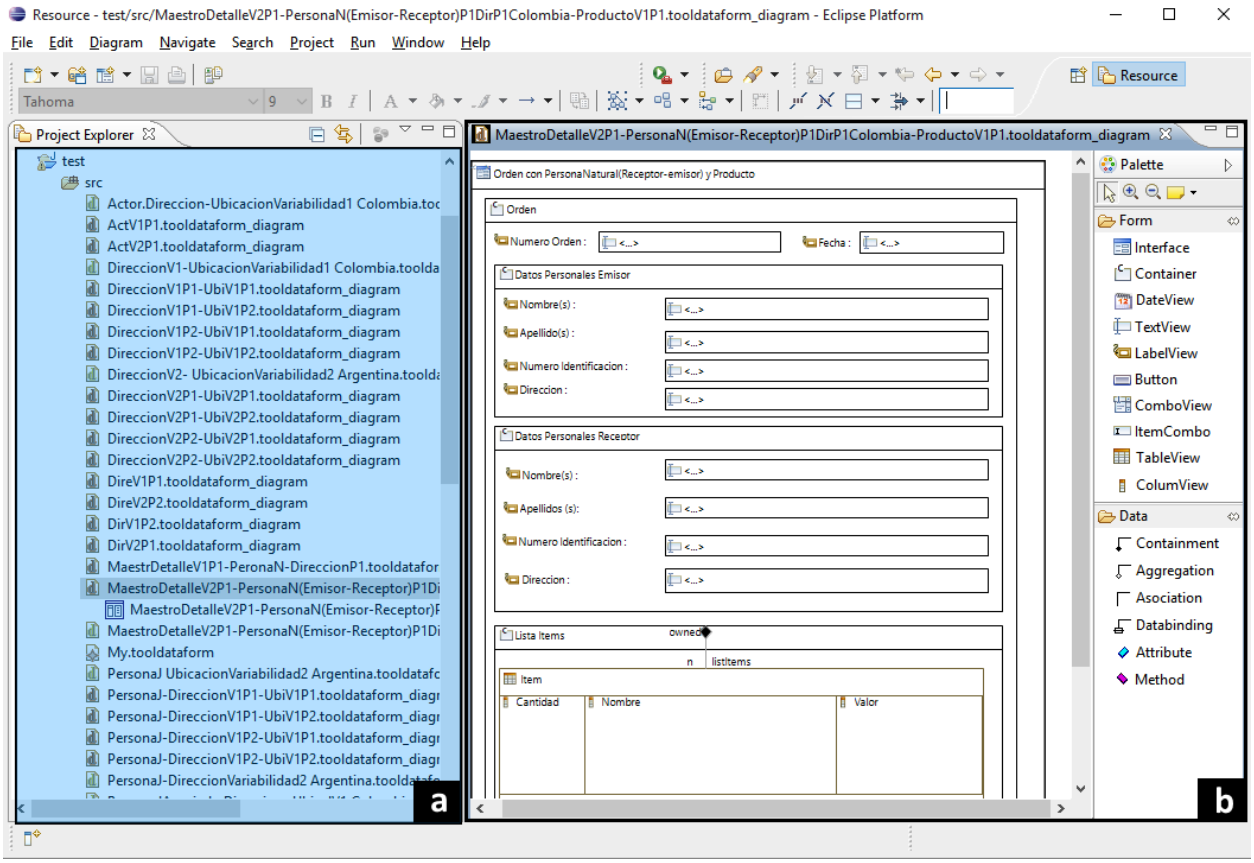

Figura 27 Ejemplo Modelo DataForm para el patrón Orden-Detalle variabilidad Factura desarrollado en toolDataForm

La combinación de patrones identificada en la tarea descrita en la sección 5.1.4, debe reflejarse al implementar un modelos DataForm de un patrón que tiene relación con otro, es decir, se debe tener en cuenta la presentación del patrón relacionado. Por ejemplo en la Figura 28 se puede observar que para la definición del DataForm de Dirección se tienen en cuenta las dos presentaciones de la variabilidad *Ubicación Colombiana* dando como resultado dos modelos diferentes (Figura 28.a) y b).

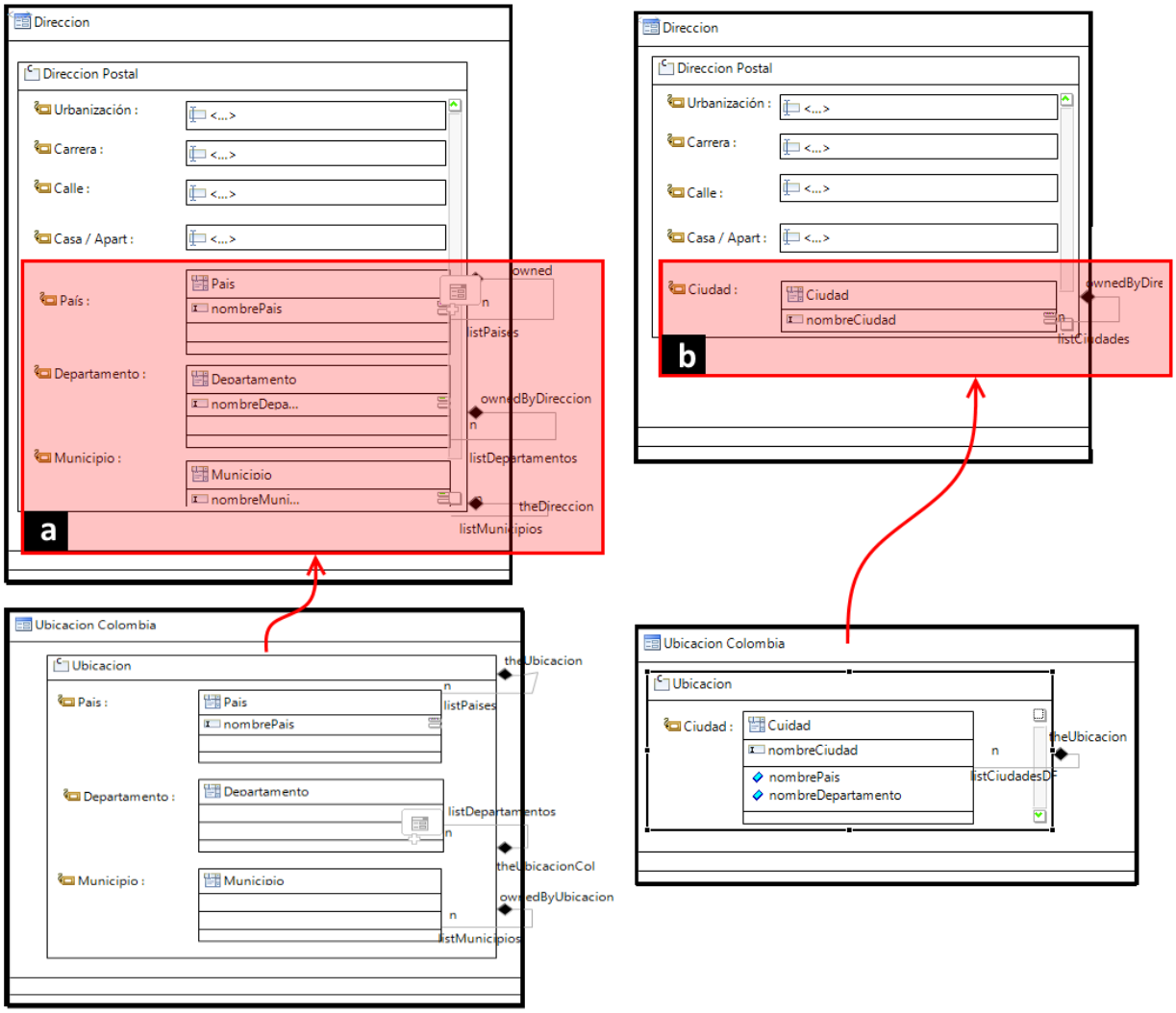

Figura 28 Patron Direccion relacionado con dos presentaciones diferentes del patron Ubicacion

5.2.3 Implementación de la Interacción

En la plantilla de presentación se captura la interacción que el usuario desea que tenga la interfaz gráfica, por lo tanto, una vez construido el DataForm se implementa a través de él la interacción haciendo uso del elemento *Method (ver* Figura 29.*a)*, el cual permite incluir bloques de lógica de programación en el DataForm, que implementen la funcionalidad requerida en el comportamiento de algún elemento, por ejemplo, un listado municipios se filtre por el valor de un departamento.

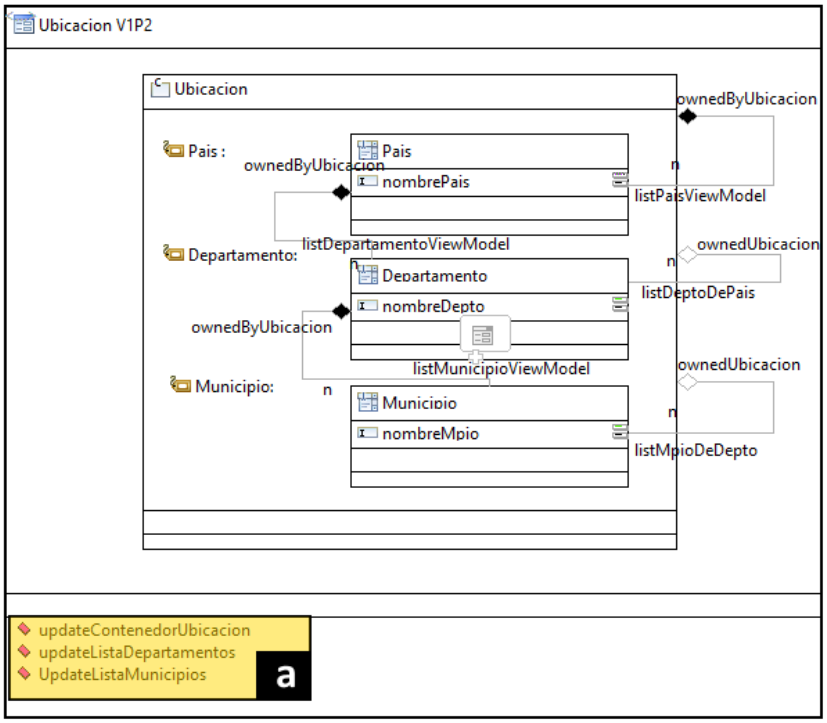

Figura 29 Definición de la interacción de los elementos del DataForm

Definidos el domain, el DataForm y el modelo de interacción (implementado directamente en el DataForm), como se expuso en la sección 3.5.1, se puede derivar de ellos el prototipo de interfaz, automatizando su generación haciendo uso del patrón MVVM.

5.3 CONFORMACIÓN DE LA TRIADA QUE CONFORMA LA VARIABILIDAD DE UN PATRÓN

La generación de los prototipos de la interfaz de usuario de negocios, como se expuso en la sección anterior, se puede automatizar a parir del DataForm y del domain data (patrón de datos). Para el desarrollo de la Interfaz basado en el modelo DataForm se adopta el patrón MVVM (ver sección 3.5.1), en la Figura 30 se muestre el proceso de automatización de la BUI a partir de dicho patrón.

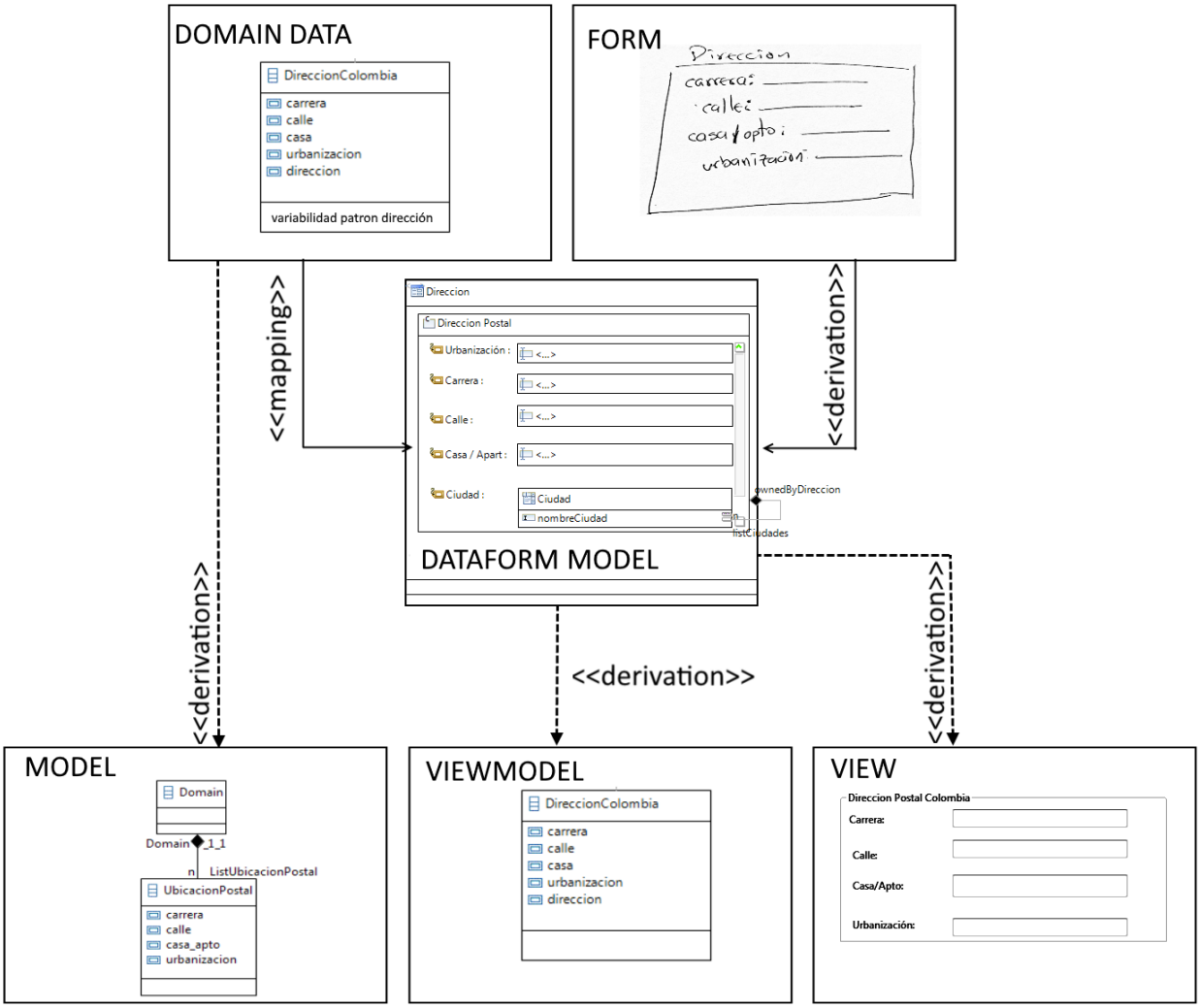

Figura 30 Proceso automatización Prototipos BUI

En consecuencia, si cada variabilidad se conforma de una triada compuesta del patrón de datos, plantilla de presentación y el modelo de interacción (ver Figura 31), las dos últimas representadas con el DataForm, se puede derivar automáticamente el view, el view model y model, facilitando la generación de prototipos funcionales de la interfaz funcionales que permitan una mejor validación de manera temprana, en la Figura 32 se muestra la conformación conceptual de la triada y el soporte a partir de ella de los elementos del patrón MVVM.

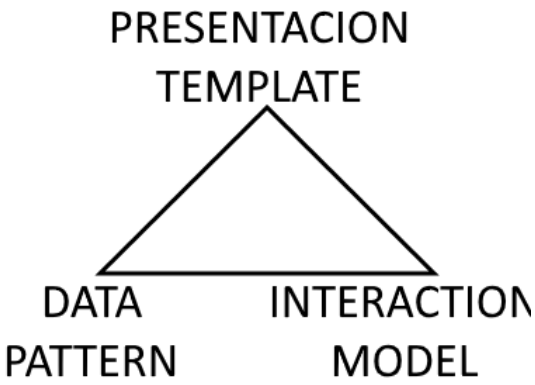

Figura 31 Elementos que conforman la variabilidad de un patrón

El término variabilidad en esta tesis representa la composición de la triada, es decir, cuando hacemos referencia al concepto variabilidad del patrón, se abstrae los diferentes elementos que se mencionan en esta sección.

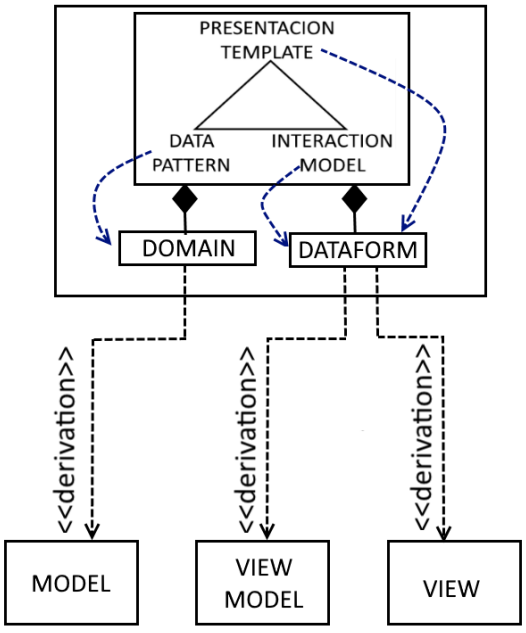

Figura 32 Proceso de Generación de la BUI a partir de la Triada

Una vez definidos los diferentes DataForm resultantes de la variabilidades y combinaciones de patrones, se requiere un mecanismo que permita agrupar y navegar las distintas presentaciones de los patrones, ya que como se explicó en la sección 5.5.2 la utilización de los patrones se realiza a partir de estas. Para este

fin se implementa un árbol de navegación de patrones, el cual se basa en las combinaciones identificadas en el proceso de definición de los patrones, en la sección siguiente se describe este recurso.

5.4 CONSTRUCCIÓN DE ÁRBOL DE NAVEGACIÓN

Como se describe en las tareas anteriores, un patrón de negocio tiene un conjunto de variabilidades en datos, cada una de ellas se conforman de un modelo de datos y un subconjunto de presentaciones representadas cada una con modelos DataForm. Para lograr representar el catálogo de patrones, en el cual se pueda visualizar y navegar por las distintas variabilidades de los patrones y sus posibles combinaciones se define un árbol de navegación, en el cual cada patrón represente una rama del árbol y a partir de ellas se visualice cada una de sus variabilidades, tanto a nivel de datos como a nivel de presentación.

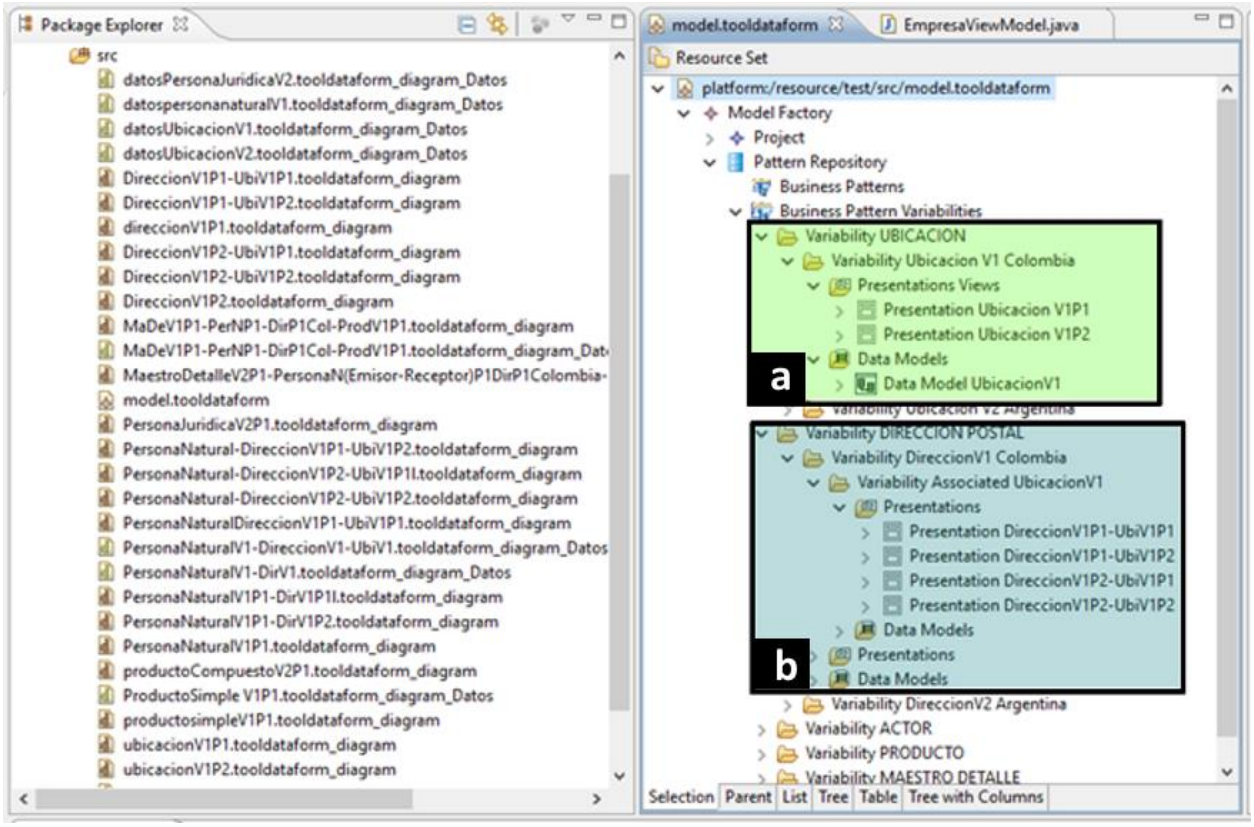

Figura 33 Catalogo de variabilidades implementado

Los patrones de negocios se agrupan en el concepto Business Patterns y a través del concepto Business Patterns Variability las Distintas variabilidades. Cada Variabilidad está compuesta por un modelo de datos

y los DataForm derivados de cada presentación, asimismo los patrones con los cuales se puede asociar. En la Figura 85 se puede observar la implementación del catálogo.

Los patrones podrán ser seleccionados a través del árbol de navegación, en la Figura 33.a se puede visualizar que el patrón UBICACIÓN GEOGRÁFICA tiene una variabilidad Colombia la cual incluye un modelo de datos y dos presentaciones. Asimismo, en la Figura 33.b se observa que el patrón DIRECCIÓN se relaciona a través de la variabilidad 1 con el patrón UBICACIÓN GEOGRÁFICA. De esta forma podemos a partir de un patrón, visualizar sus diferentes variabilidades y navegar a través de los patrones que tiene relacionados.

5.5 PROTOTIPO DE INTERFAZ A PARTIR DE LAS VARIABILIDADES DE LOS PATRONES

Un prototipo de interfaz de usuario de negocio es una plantilla de presentación para los datos necesarios para la realización de una tarea, en este trabajo se pretende que su construcción sea posible a partir de la reutilización de patrones, por lo tanto, es necesario definir un mecanismo que facilite la búsqueda, selección y utilización de los patrones, permitiendo mezclar las plantillas de presentación existentes para generar una nueva.

Tomando como base metodológica TD-MBUID, se identifica que la realización del prototipo de interfaz de usuario de negocio (ver Figura 7) se soporta en las disciplinas de modelado del negocio y del modelado de la interfaz de usuario de negocio, así mismo, como se puede observar en la Figura 8 su desarrollo se basa en los componentes del *DataForm, data domain model y paper form*. Teniendo en cuenta estos fundamentos se propone realizar las siguientes actividades:

5.5.1 Definición del contexto

Este paso se fundamenta en la disciplina del Diseño del dominio de negocio, en el cual se estudia el dominio de la aplicación, pero se centra en la identificación del contexto del problema, se busca entender cuáles son los conceptos de negocios involucrados, para posteriormente buscar y seleccionar, si existe, los patrones que los puedan representar.

Para ilustrar este paso, se toma de ejemplo el problema de definir un directorio de clientes que permita a una empresa ubicarlos. De este contexto se puede identificar los conceptos de negocio de **cliente** y **dirección**.

5.5.2 Selección del patrón y variabilidad

Los patrones definidos representan soluciones a problemas particulares, por lo tanto, para su selección se debe primero analizar el contexto del dominio, buscar del conjunto de patrones descrito en la sección anterior, las variabilidades que representen los conceptos identificados en el contexto del dominio.

De acuerdo a los conceptos de negocio identificados en la sección anterior se identifican los siguientes patrones:

Actor: representa algo o alguien representativo en un contexto, por lo tanto, El cliente es un actor de la empresa, quien interactúa con la empresa va a realizar la compra, por lo tanto, se necesitan los datos para identificarlo, asimismo su dirección para su ubicación.

Dirección: Como se menciona en la descripción del contexto, se requiere la ubicación del cliente para su ubicación, por lo tanto, este patrón permite a través de la variabilidad *dirección Postal* definir el lugar exacto donde viven.

Ubicación Geográfica: Este patrón permite definir una ubicación geo-política, es decir, indicar el país, ciudad donde reside.

Este conjunto de patrones satisface los datos necesarios para definir un directorio de clientes, por lo tanto, seleccionamos de estos las presentaciones que queremos definan el prototipo de interfaz. En la Figura 34.a) se expone el orden de los patrones y presentaciones seleccionadas, se puede observar que primero se selecciona la presentación de la variabilidad *PERSONA NATURAL*, luego *DIRECCIÓN POSTAL* y por último *UBICACIÓN GEOGRÁFICA*. La combinación de estos patrones genera la propuesta de interfaz de usuario de negocio, en la Figura 34.c) se visualiza la interfaz generada en donde se puede reconocer que esta se conforma de la mezcla de las presentaciones seleccionadas.

Al selección un patrón se debe adicionar al dominio de la aplicación las clases que componen su estructura en datos, por lo tanto, en la Figura 35.a) se presenta el dominio generado después de seleccionar los tres patrones indicados; se puede identificar que la clase *Persona* que corresponde al patrón *PERSONA NATURAL*, la clase *DireccionPostal* adicionada al dominio a causa de la selección de *DIRECCIÓN POSTAL* y las clases *Municipio, Departamento y País* que representan el patrón ubicación geográfica.

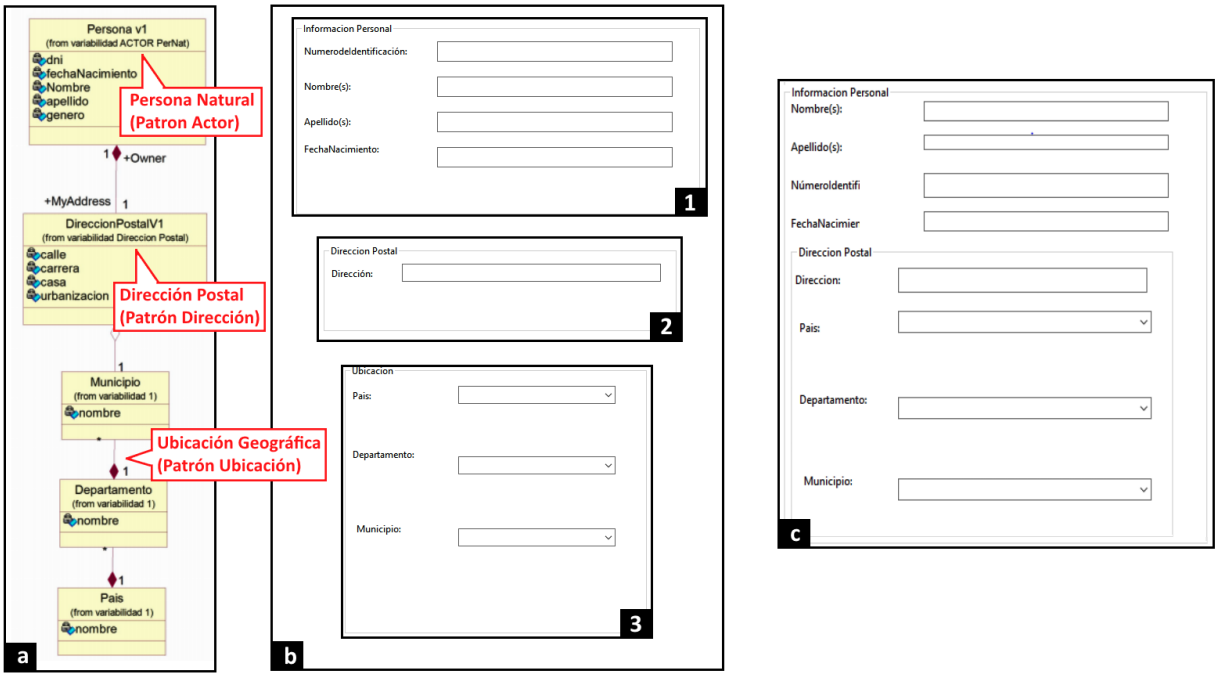

Figura 34 Construcción del prototipo de interfaz a partir de los patrones de negocio

Con este proceso se define el *Business Domain Data,* el cual se forma con el conjunto de clases que generan los patrones seleccionados (Figura 34.a) el cual da soporte a los datos de la aplicación, la combinación de las presentaciones escogidas conforman el *Paper Form* (Figura 34.c). *El DataForm* no se visualiza ya que está construido en el background de las presentaciones de cada patrón, aunque en este proceso manual no se demuestra su uso, es un componente fundamental en la implementación de la generación automática, tal como se expuso en la sección 5.3, a partir del DataForm y el patrón de datos se derivan la view-viewmodel y model que permite la implementación del prototipo de interfaz de usuario.

CAPÍTULO SEXTO

6. PATRONES EN ESTA TESIS

En esta propuesta se define el método para identificar la asociación de los patrones de negocios, de presentación e interacción, asimismo, se presenta la propuesta metodológica para incorporarlos en el modelado de la interfaz de usuario de negocio definido en TD-MBUID.

6.1 PROCESO DE IDENTIFICACIÓN DE PATRONES

Como se ha mencionado un patrón es un modelo que representa una solución recurrente a un dominio de problema específico, es decir, describe y especifica cómo se puede solucionar un problema en particular ya sea de modelado, implementación u otro nivel de abstracción. A diferencia de TD-MBUID, el método propuesto para el desarrollo de la interfaz de negocio se centra en los datos, por esta razón los patrones que se buscan en este trabajo son los relacionados a soluciones de modelado de datos

6.1.1 Identificación y asociación de patrones de negocios y patrones de presentación

En la Figura 16 se presenta el método para la identificación y asociación de las presentaciones con los patrones de negocios, como se puede observar, se compone de tres pasos: identificar el patrón de negocio; definir las plantillas de presentación del patrón y por último definir su interacción.

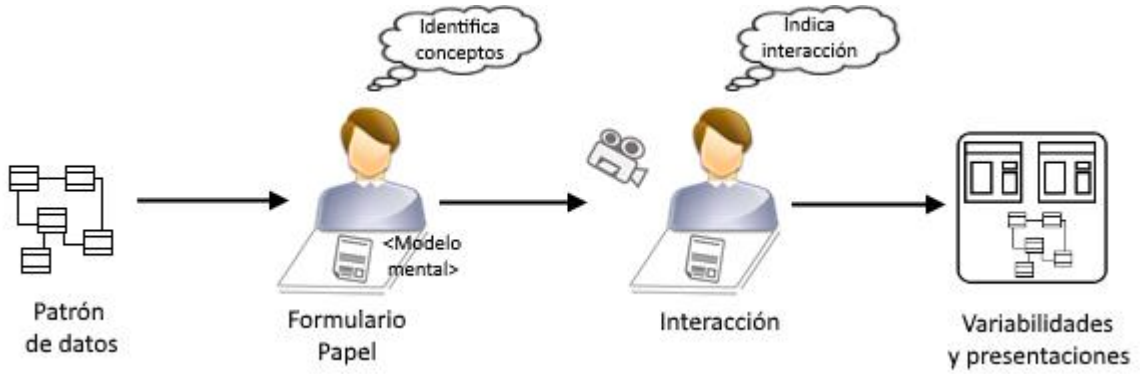

Figura 35 Proceso Identificacion y asociacion de patrones

6.1.2 Identificación de patrón de negocio

El proceso empieza con la definición de los patrones de negocios, los cuales como se mencionó describen conceptos (datos) a nivel de negocio, especifican cómo modelar y estructurar recursos y conceptos.

La identificación de estos patrones se puede realizar de dos maneras:

Revisión en la literatura: a través de una búsqueda en el estado del arte se puede encontrar lenguajes o catálogos de patrones propuestos por expertos en el área, su divulgación y discusión garantiza la calidad y veracidad de dichos patrones, en este sentido tenemos propuestas como (Eriksson & Penker, 1998; E. Gamma et al., 1994; Hay, 1996, 2010; Silverston & Agnew, 2009).

Observación y experiencia: Otra forma de identificar patrones es por medio de la observación, para encontrar semejanzas y de abstracción, descartando detalles irrelevantes y describir con precisión la esencia del problema. La experiencia obtenida en el desarrollo de proyectos permite encontrar conceptos comunes a múltiples contextos de problemas, formando a través de estos nuevos patrones de negocios.

En esta tesis se utilizó la primera opción, la revisión en el estado del arte, para determinar los patones que conforman el catálogo propuesto.

Una vez definido el patrón, se debe determinar su variabilidad, para esto se debe analizar distintos dominios, determinando que se debe ajustar del patrón de negocio para satisfacer las especificidades de cada contexto diferente.

6.1.3 Plantilla de Presentación

Una vez definidos los conceptos y sus relaciones (patrón de negocio) que soportan los datos del domain, se procede a identificar el modelo mental del usuario para definir el patrón de presentación que lo satisfaga, para esto se propone con el apoyo de los usuarios y de acuerdo al contexto de una tarea, realizar los siguientes pasos:

Identificar conceptos: de acuerdo al contexto de la tarea se le solicita al usuario que de forma escrita o hablada indique cuales son los conceptos de negocio involucrados, de esta manera podemos obtener cuales son los conceptos que el usuario necesita manejar para realizar la tarea.

Identificar relaciones: Teniendo en cuenta los conceptos identificados, se le solicita que revele la relaciones que existen entre ellos, para ello, se le puede indicar que realice oraciones con ellos, esto permite identificar como el usuario piensa que están asociados, es decir, cual es el modelo mental que tiene sobre los datos.

Construir el bosquejo: de acuerdo a los pasos anteriores se genera el bosquejo en papel, que involucre en su diseño tanto los conceptos como la asociación que el usuario definió sobre los datos.

6.1.4 Patrón de interacción

Con el formulario en papel definido y a través del apoyo tecnológico, se le solicita al usuario que, dentro del contexto de una tarea, diligencia el formulario en papel. La interacción del usuario es grabada en video para ser revisado y analizado, de acuerdo a la forma como el usuario diligencio el formulario se determina el patrón de interacción.

En esta propuesta, no se documenta, ni se visualiza en el catálogo este tipo de patrón, ya que en la plantilla de presentación y en el *DataForm* se incluye la forma de la interacción que tendrá la Interfaz de usuario de negocio una vez se les realice la renderización de ellos.

6.2 CATALOGO DE PATRONES DE NEGOCIO PROPUESTO

Como se mencionó un catálogo es un conjunto de patrones relacionados entre sí y clasificados por algún criterio, los cuales pueden ser utilizados conjuntamente o de manera independiente, que permite y facilita a un desarrollador, es este caso de la interfaz de usuario, buscar y hacer uso de los patrones que lo conforman. En este capítulo se describe el catálogo propuesto de la siguiente manera: primero se detallan los patrones (sección 6.2) y luego se explican las variabilidades de los patrones (sección 5.3)

El catálogo propuesto consta de 10 patrones, los cuales se han clasificado de acuerdo al framework propuesto por John Zachman en (Zachman, 1987), en el cual se provee una taxonomía sistemática de conceptos para relacionar cosas que describen el mundo real con los conceptos que describen su respectivo sistema de información y su implementación (Sowa & Zachman, 1992).

El framework se compone por seis columnas y cinco filas. Cada fila o **perspectiva**, es una representación arquitectónica fundamental asociada a cada uno de los conjuntos de roles (planificador, propietario, diseñador, desarrollador e integrador) que está asociada a un nivel de abstracción. Las perspectivas son:

- Alcance (Scope): Es el primer bosquejo arquitectural del sistema
- Modelo de negocio o de empresa (Enterprise or business model): Modelos de la empresa, los cuales constituyen el diseño del negocio.
- Modelo de sistema (System model): Modelo del sistema, diseño desde la perspectiva del diseñador.
- Modelo de la Tecnología (Technology model): perspectiva del desarrollador, los cuales deben considerar las restricciones de la tecnología.

- Componentes (Components): Especificaciones detalladas que son dadas a los programadores, quienes codifican módulos individuales sin preocuparse por el contexto o estructura completa del sistema.

El framework plantea la existencia de diferentes tipos de descripciones relativas a diferentes aspectos de los sistemas que se describen. Cada una de las diferentes descripciones se prepara con un motivo distinto, cada una es independiente y diferente de las otras, aunque todas puedan pertenecer al mismo objeto, razón por la cual están intrínsecamente relacionadas unas con otras. Esta última idea queda plasmada gracias a la especificación de distintas vistas o abstracciones.

Las columnas o **Vistas** representan un nivel de abstracción diferente, un tipo de descripción relativa de la perspectiva que se describe. Las vistas que propone el framework:

- Los Datos (qué), The Data (What).
- Los procesos (cómo), The Process (How).
- La red (dónde), The Network (Where).
- Las personas (quién), The People (Who).
- El tiempo (cuándo), The Time (When).
- La motivación (Por qué), The Motivation (Why).

Para la clasificación o agrupación de los patrones que conforman el catalogo propuesto, se usa los clasificadores propuestos por las vistas del framework de Zachman, las perspectivas no son tenidas en cuenta ya que este trabajo se enfoca en el modelado del negocio y por lo tanto solo aplicaría la perspectiva de Enterprise or business model. En la Tabla 2 se puede observar los patrones que conforman el catalogo propuesto con su respectiva clasificación.

Clasificación	Patrón
Que	Producto
	Orden – Detalle
	Documento
	Empleo
Cómo	Pago
Dónde	Ubicación Geográfica
	Dirección Postal
Quién	Actor
	Unidad Organizacional
Cuando	
Por qué	

Tabla 2 Patrones propuesto por clasificación

Esta clasificación permite la incorporación ordenada de nuevos patrones, asimismo describe la cobertura de los contextos de dominio que pueden ser soportados por esta colección, por ejemplo, se puede concluir que los términos de negocio relacionados al cuando se salen del alcance de los problemas que pueden ser modelados a través del catálogo.

Un factor diferenciador de este catálogo es la conformación de cada patrón. Se presenta por cada patrón un sub-catálogo conformado por parejas dispares de variabilidad – presentaciones; variabilidades o adaptaciones del patrón y las respectivas formas de presentación (interfaz de usuario de negocio) que soporten los datos descritos en la variabilidad.

6.3 LA COMPOSICIÓN DE LOS PATRONES PROPUESTOS:

Los patrones propuestos en este catálogo a diferencia de propuestas en otros trabajos como (Blaha, 2010; Eriksson & Penker, 1998; Hay, 1996, 2010; Silverston & Agnew, 2009) se componen cada uno de la asociación de un patrón de datos con su variabilidades (adaptaciones del patrón según determinado contexto), plantillas de presentación que soporta cada variabilidad con su respectiva interacción.

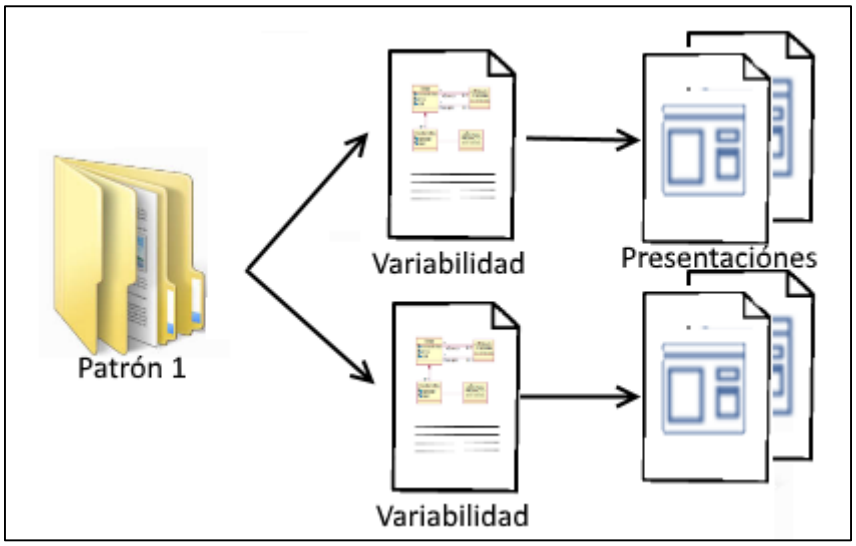

Figura 36 Composicion de cada patrones de negocio del catalogo propuesto

Cada patrón que lo conforma se divide en dos partes: patrones de negocio, el cual está conformado por los conceptos genéricos los cuales son las bases para definir las soluciones de modelado de datos; segundo las variabilidades del patrón, conformado por el conjunto de variabilidades de cada patrón de negocio, estas variabilidades son adaptaciones o producciones del patrón, aplicadas a contextos específicos, cada

variabilidad se compone del modelo de datos (generado de acuerdo al patrón de negocio) y de plantillas de presentación para soportar los conceptos que se exponen en dicho modelo. En la Figura 36 se describe la composición de cada patrón, en ella se puede observar que un patrón tiene un subconjunto de variabilidades y estas a su vez un subconjunto de presentaciones.

6.4 FORMATO DE ESPECIFICACIÓN

Teniendo en cuenta que uno de los propósitos principales de los patrones es comunicar soluciones reutilizables a otros desarrolladores, un factor crítico es cómo transmitir al destinatario la información clara para que sea capaz de comprender la motivación, la esencia y la utilidad del patrón. Para lograr es necesario emplear un formato o plantilla lo más homogénea posible para describir las características de los patrones(Moreno, 2002).

En la literatura encontramos diversas propuestas para la presentación de patrones, entre ellas se destacan las plantillas “Alejandrina” propuesta por Christopher Alexander (de ahí su nombre) en (Alexander et al., 1977)y la plantilla “GoF format” definida por la banda de los cuatros en (Erich Gamma et al., 1995)

El formato propuesto por Alexander describe la solución del patrón de forma más abstracta, mostrando un dibujo que representa dicha solución y permitiendo que sea implementado de diferentes maneras, mientras que el formato “GoF” es más detallado, llegando a incluir campos tales como el pseudocódigo o los diagramas de diseño, que hacen casi directa la implementación software del patrón.

En general, una plantilla debe contener las secciones necesarias para entender y poder reutilizar la solución planteada en el patrón, para este trabajo se utilizará una plantilla basada en el formato GOF, pero ajustarla a los patrones propuestos, por lo tanto, se le adiciona la sección de variabilidad.

La plantilla usada se compone de las siguientes secciones:

Nombre: Cada patrón propuesto tiene un nombre corto y distintivo que permite diferenciarlo y recordarlo fácilmente, debe representar claramente el concepto de negocio que representa. El nombre es de gran importancia porque permite definir un vocabulario común para su discusión, uso y combinación con otros patrones entre los desarrolladores o modeladores que utilizan el catálogo.

Clasificación: Indica la clasificación del patrón de negocio.

Propósito: Descripción resumida del propósito general del patrón, indica sus metas y objetivos, es decir, que resuelve el patrón y que problemas soluciona.

Aplicabilidad: Esta sección describe las situaciones problemáticas en las cuales se puede aplicar el patrón y cuales problemas pueden ser resueltos por él. Indica en qué contexto puede ser aplicado el patrón.

Estructura: Constituye la representación gráfica genérica del patrón en UML, el modelo usado para su visualización es el diagrama de clases.

Participantes: Define y describe los conceptos (elementos de modelado) que se visualizan en la sección *Estructura* y que conforman el patrón.

Ejemplo: Proporciona un ejemplo de implementación donde se usa el patrón para resolver un problema

Patrones relacionados: Indica patrones complementarios que se pueden relacionar con el patrón descrito.

Fuente: En esta sección se indica o se da crédito a la fuente o inventor del patrón original.

Variabilidades: En este ítem se explican cada variabilidad en datos del patrón y se presenta las plantillas de presentación.

6.5 PATRONES DE NEGOCIO

En esta sección se describen cada uno de los patrones de negocios propuestos hacienda uso de la plantilla de presentación indicada en el capítulo 6.4.

6.5.1 Patrón Ubicación Geográfica

Nombre: Ubicación Geográfica

Clasificación: Donde

Propósito: La intención del patrón Ubicación Geográfica es permitir modelar cualquier ubicación geopolítica en la tierra independiente de su constitución, ya que según la región o país la forma de división política cambia, por ejemplo, Colombia está dividida en 3 niveles de jerarquía mientras que España en 4; asimismo, el patrón permite definir el nivel de granularidad que se necesite para definir la ubicación, como lo puede ser, llegar a especificar barrios, zonas.

Aplicabilidad: El patrón Ubicación Geográfica puede ser usado, en cualquier caso, que se necesite modelar un sitio geográfico en la tierra, por ejemplo, un país, una región.

Este patrón puede ser aplicado en dominio de problemas de oficinas de correos, definición de viajes, ventas por internet.

Estructura:

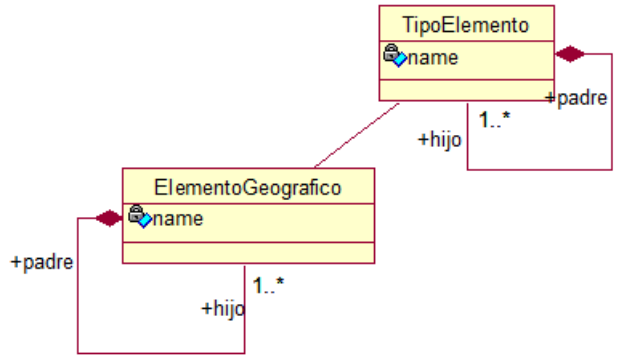

Figura 37 Patrón Ubicación Geográfica

Participantes:

ElementoGeografico representa una posición geográfica en particular, puede ser un país, una ciudad, una población. Una ubicación geográfica se construye como una jerarquía, ya que es común que geopolíticamente se subdividan, por ejemplo, el elemento geográfico Quindío forma parte de Colombia. Toda ubicación tiene en común el atributo nombre.

TipoElementoGeográfico Especifica el tipo específico de un Elemento Geográfico. Un tipo de elemento geográfico puede ser País. Toda instancia de Elemento Geográfico debe estar relacionado con este elemento para indicar que tipo de elemento tiene. Se construye como una jerarquía para indicar los subtipos que un Tipo de elemento, por ejemplo, País – Departamento – Municipio.

Ejemplo: Un ejemplo para el uso de este patrón es definir el destino de un viaje donde se desea especificar la ciudad de Cartagena de Indias ubicada en el departamento de Bolívar del país Colombia como destino. En esta ubicación se tendría como elementos geográficos Cartagena de Indias que pertenece a Bolívar y a su vez esta última forma parte de Colombia, Cada uno de ellos relacionados respectivamente a los tipos de elementos municipio, departamento y país.

Otro ejemplo de aplicación de este patrón es definir la ubicación de un puesto especifico en un evento, donde se puede especificar zona, fila y silla, siendo cada uno de ellos un tipo de elemento geográfico y las respectivas instancias elementos geográficos.

Patrones relacionados:

El patrón puede combinarse con cualquier patrón que requiera una *Ubicación Geográfica* especifica como dirección, Unidad Organizacional.

Fuente: Este patrón es basado en el patrón localización geográfica presentado por David C. Hay en el libro Data Model Patterns: Conventions of Thought.

Variabilidades: Para este patrón se presentan las variabilidades de: Ubicación Colombiana, Ubicación Argentina, Ubicación Española**.**

Variabilidad Ubicación Colombiana*:* Esta variabilidad permite definir una ubicación con la estructura geopolítica de Colombia.

Estructura:

Figura 38 Estructura Variabilidad Ubicación Colombiana

Presentaciones:

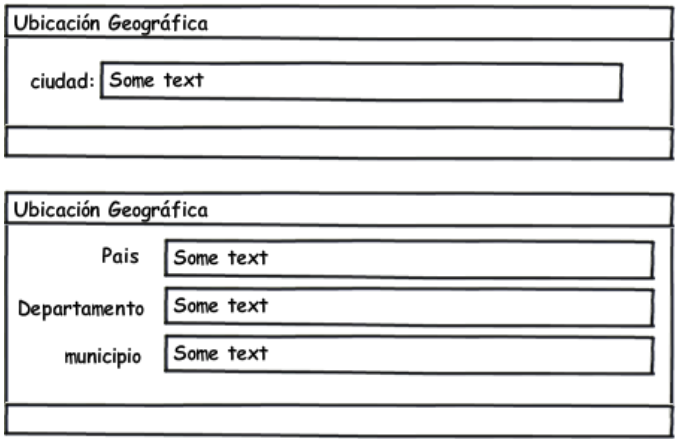

Figura 39 Presentaciones de la variabilidad Ubicación Colombia

Variabilidad Ubicación Española: Permite definir una ubicación geográfica haciendo uso de la estructura geopolítica española

Estructura:

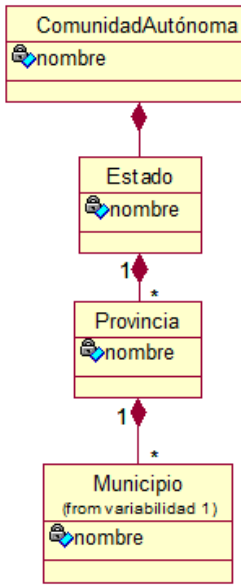

Figura 40 Estructura Variabilidad Ubicación Española

Presentaciones:

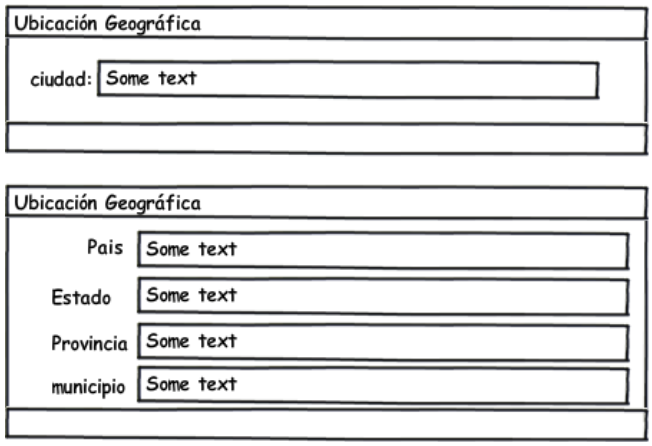

Figura 41 Presentaciones Variabilidad Ubicación Española

6.5.2 Patrón Dirección

Nombre: Patrón Dirección

Clasificación: Donde

Propósito: La motivación de este patrón es poder modelar una dirección, la cual representa un método a través del cual se puede ubicar o comunicar con un actor, sin importar el mecanismo preciso, por ejemplo, una dirección postal, una dirección email, teléfono o una URL.

Una dirección es uno de los conceptos vitales en un negocio, no obstante, suele ser mal modelado o interpretado, históricamente la dirección se ha entendido como un sitio físico especifico, pero en la actualidad este concepto se ha ampliado a definiciones no físicas como el teléfono, el email. Hoy en día es habitual que un actor, por ejemplo, un cliente tiene varias direcciones postales, asimismo email, teléfonos, este patrón permite modelar los principales tipos de direcciones.

Aplicabilidad: El patrón Dirección puede ser usado, en cualquier caso, que se necesite especificar cualquier tipo de dirección de algún actor, por ejemplo, una persona, una empresa, un sitio web.

Estructura:

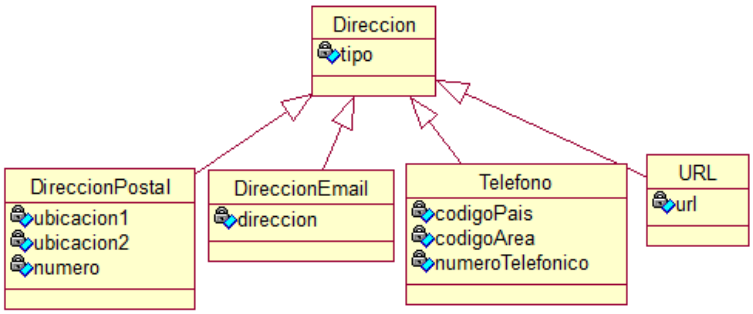

Figura 42 Patron Dirección

Participantes:

Dirección: Clase abstracta que representa cualquier variedad de dirección, en ella se especifica un tipo que permite aplicar algún clasificar a una dirección concreta, por ejemplo, personal o empresarial.

DireccionPostal: Clase concreta que permite definir una dirección postal, entendiendo esta como una dirección que especifica la ubicación de un lugar físico.

Teléfono: Clase concreta que permite definir el número telefónico, este se considera una dirección ya que es un mecanismo de comunicación con un actor.

URL: Permite definir la dirección Localizador de recursos uniforme (URL) que permite definir la dirección de un sitio web.

Ejemplo: Un ejemplo de dominios de aplicación para el uso de este patrón es el envío de una correspondencia, en la cual se debe especificar la dirección postal de dos personas, el remitente y el destinatario, asimismo se podría indicar el teléfono de cada uno de ellos.

Otro Contexto de aplicación se da en negocios donde se tienen proveedores, clientes, en donde es necesario indicar los mecanismos de comunicación con cada uno de ellos, por ejemplo, teléfonos, email y dirección postal.

Patrones relacionados: Este patrón se puede combinar con cualquier patrón que requiera una dirección, por ejemplo, el *patrón Actor*, *Empleado*, *Unidad Organizacional*.

Este patrón se puede complementar con el *Patrón Ubicación Geográfica,* para indicar el lugar geográfico al cual pertenece la dirección.

Fuente: Este patrón es basado en el arquetipo Address descrito por Michael Blaha en el libro Patterns of Data Modeling y del patrón localización geográfica presentado por David C. Hay en el libro Data Model Patterns: Conventions of Thought.

Variabilidades: Las variabilidades de este patrón se presenta en la dirección postal, la cual varía de acuerdo al país o región. Para la dirección Postal se presentan las variabilidades de: Dirección Colombiana, Dirección Europea***.***

Variabilidad Dirección Postal Colombiana: Esta variabilidad permite definir la dirección de alguna ubicación con el formato utilizado en Colombia.

Estructura:

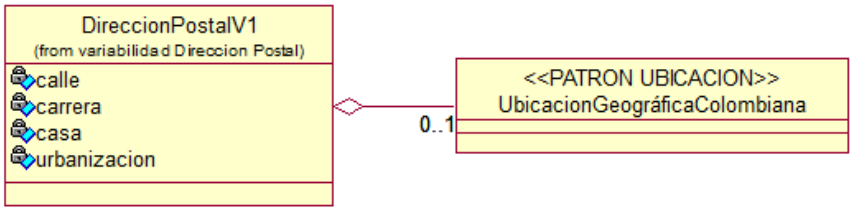

Figura 43 Estructura variabilidad Dirección Postal Colombiana

Presentaciones:

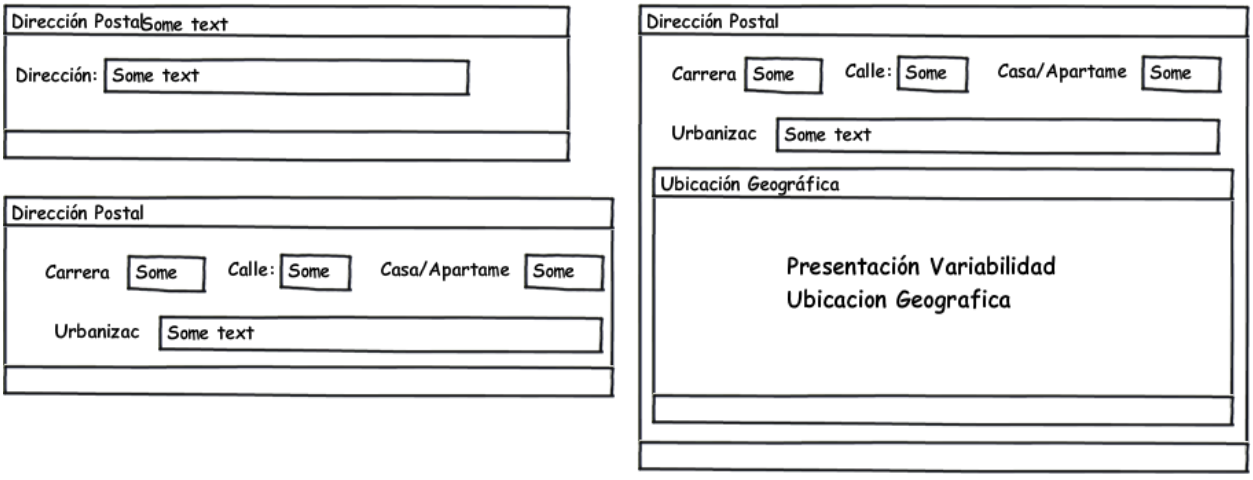

Figura 44 Presentaciones Variabilidad Dirección Postal Colombiana

Variabilidad Dirección Postal Europea: Permite describir una dirección postal usando el formato europeo.

Estructura:

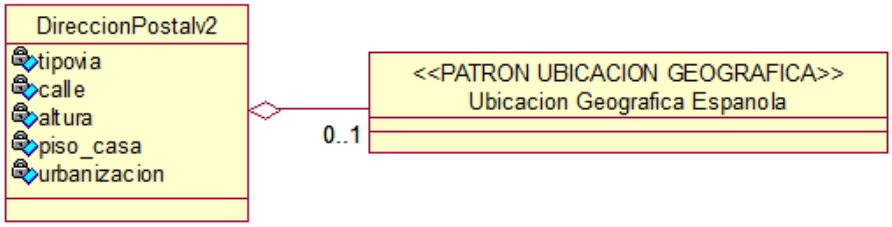

Figura 45 Estructura variabilidad Dirección Postal Europea

Presentaciones:

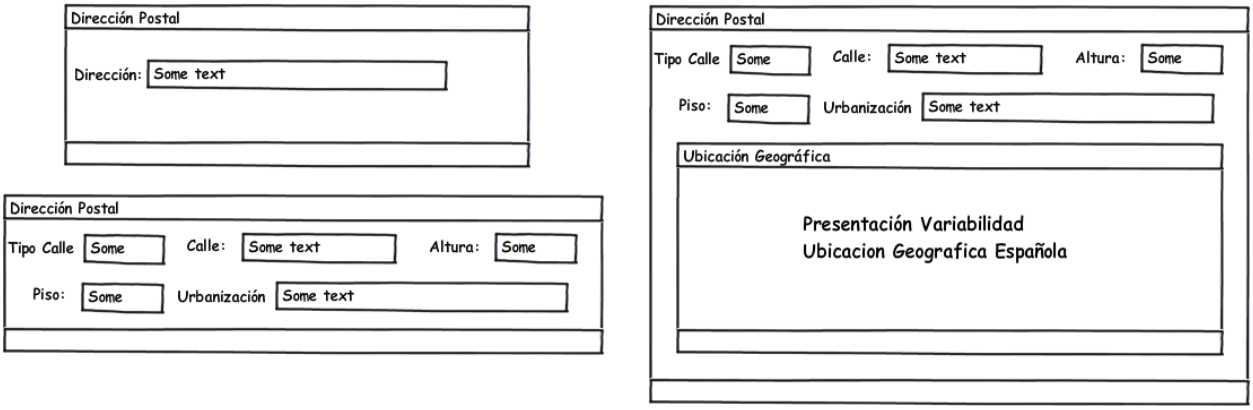

Figura 46 Presentaciones variabilidad Dirección Postal Europea

Variabilidad Dirección Telefónica: Permite definir un número telefónico, se considera dirección ya que es un medio para localizar a algún Actor.

Estructura:

Figura 47 Estructura Variabilidad Dirección Telefónica

Presentaciones:

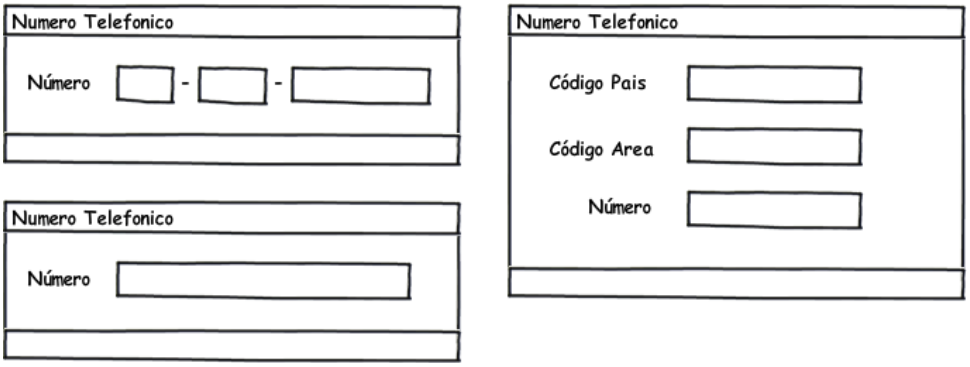

Figura 48 Presentaciones Variabilidad Dirección Telefónica

6.5.3 Patrón Unidad Organizacional

Nombre: Unidad Organizacional

Clasificación: Quien

Propósito: El patrón unidad organizacional permite definir de manera flexible la composición de una organización. Las empresas están estructuradas en unidades para facilitar su gestión y comprensión, por ejemplo, una empresa puede estar dividida en tres departamentos, estos a su vez en dependencias.

Una organización rara vez es estática, lo que genera un problema de modelado, ya que su especificación puede variar mucho y generar que el modelo realizado quede obsoleto, este patrón da soporte a la modificación de la estructura organizacional de una compañía.

Aplicabilidad: Este patrón puede ser usado en cualquier situación donde se desee modelar una estructura organizacional, es decir, para definir cualquier el organigrama institucional.

Estructura:

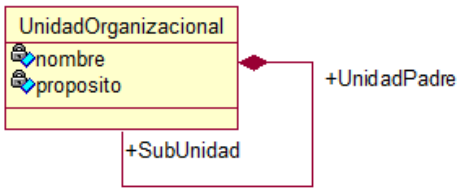

Figura 49 Patrón Unidad Organizacional

Participantes:

UnidadOrganizacional: Representa una unidad o parte especifica de la empresa y su propósito. Tiene una relación jerárquica permite definir la estructura de la empresa, es decir, unidad y sus subunidades.

Ejemplo: Un ejemplo de aplicación de este patrón es cualquier organización, tomemos como caso una empresa genérica constituida de acuerdo al organigrama mostrado en la Figura 1

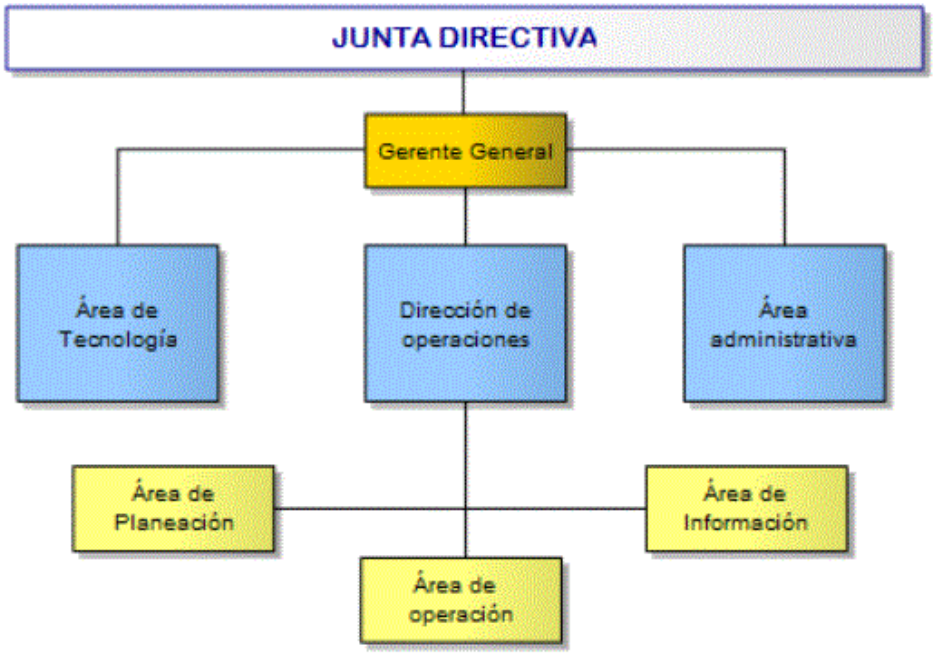

Figura 50 Organigrama de una empresa genérica

Esta estructura se puede modelar fácilmente con el patrón, donde cada área sería una instancia de la clase *UnidadOrganizacional*, las cuales estarían relacionadas a través de relación de composición con la dependencia de nivel jerárquico superior, por ejemplo, el Área de Tecnología tendría una relación de subunidad a la Gerencia General (su unidad padre).

Patrones relacionados: Este patrón puede relacionarse con cualquier patrón al cual se requiera indicar en con cual dependencia está asociado, por ejemplo, el patrón empleado, para indicar en que unidad está asociado. Se puede complementar con el patrón *dirección* o *ubicación geográfica* para especificar su ubicación.

Fuente: Este patrón es basado en el propuesto por Hans-Erik Eriksson and Magnus Penker en el libro Business Modeling with UML: Business Patterns at Work.

Variabilidades: este patrón no tiene variabilidades

Presentaciones:

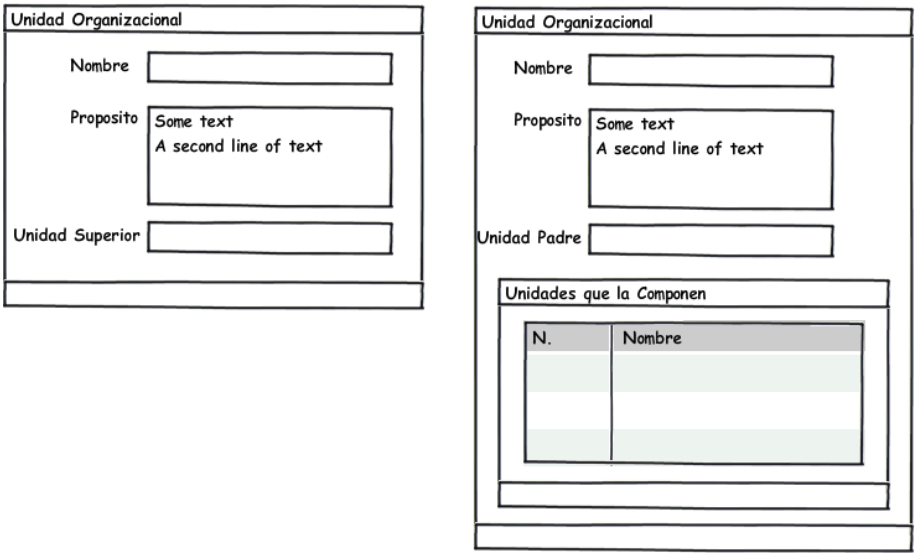

Figura 51 Presentaciones Patrón Unidad Organizacional

6.5.4 Patrón Actor

Nombre: Patrón Dirección

Clasificación: Quien

Propósito: Un actor es alguien o algo que funciona por sí solo, Tal como una máquina, una la persona o una empresa. El actor es un concepto muy importante ya que permite definir alguien o algo que participa en la realización de alguna actividad. Por ejemplo, una tarea puede tener alguien que la revise, ese alguien puede ser una persona o la descripción de un cargo o rol.

Un rol es una función o papel que un actor desempeña en un lugar o situación, por ejemplo, auditor; un actor puede tener o desempeñar más de un rol, y un mismo rol ser ejercido por varios actores.

Otro ejemplo que motiva este patrón, es el caso en que un sistema puede ser actor de otro sistema, por ejemplo, un cajero automático posee un sistema propio que se convierte en actor del sistema bancario.

La motivación principal de este patrón es separar el actor de los roles que desempeña, de no ser así, la dificultad de expresar situaciones donde un mismo actor puede cumplir varias funciones sería alta.

Aplicabilidad: El patrón Actor se puede utilizar en todas las situaciones problemáticas en las que hay una necesidad de separar los actores de los roles o funciones que desempeña. Por ejemplo, una regla de negocio para un banco podría ser que todos los grandes retiros deben ser aprobados por el administrador de la oficina del banco.

Estructura:

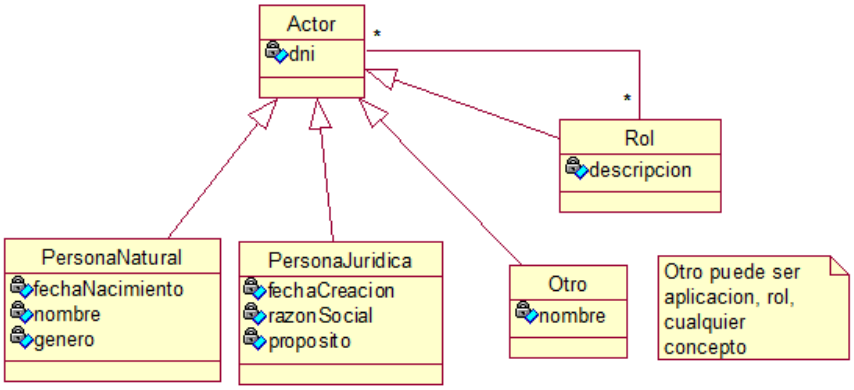

Figura 52 Patrón Actor

Participantes:

Actor: Clase abstracta que representa cualquier tipo de actor, los cuales tienen en común que deben tener un número de identificación.

PersonaNatural: Clase concreta que permite definir que el actor es una persona natural.

PersonaJuridical: Clase concreta que permite definir que el actor es una empresa u organización

Rol: La clase rol permite representar un cargo o función como un actor de algo, asimismo, permite separar las funciones del actor que las realiza.

Otro: Clase concreta que permite definir que el actor es cualquier elemento como un software, un dispositivo electrónico.

Ejemplo: Un ejemplo del uso de este patrón es un partido de futbol, tenemos diferentes actores como los jugadores, los árbitros un jugador puede cumplir varias funciones o posiciones, por ejemplo, defender y atacar por los bordes, los árbitros en cambio tienen un solo rol, ser árbitro principal o auxiliar de línea.

En este caso podemos ver que el patrón permite definir los actores generales del partido, jugador, arbitro y arbitro auxiliar, asimismo, permite definir las personas naturales (actores) que participan en el juego, por ejemplo, jorge es jugador con la posición (rol) de defensa central, pedro es jugador que tiene las funciones(roles) de defensa lateral y carrilero, Mario tiene el rol de árbitro central.

Patrones relacionados: Este patrón se puede complementar con el patrón ubicación geográfica o dirección para especificar el lugar donde se encuentra. También se puede combinar con unidad organizacional para determinar los distintos cargos o perfiles que la compone. Asimismo, se puede combinar con patrones que denoten actividades, procesos o tareas para determinar quienes participan en su ejecución

Fuente: Este patrón es basado en el arquetipo Actor descrito por Michael Blaha en el libro Patterns of Data Modeling y del patrón Actor-Role presentado por Hans-Erik Eriksson and Magnus Penker en el libro Business Modeling with UML: Business Patterns at Work

Variabilidades: Las variabilidades de este patrón representa la creación de las clases concretas, también en definir actores sin especificar sus funciones.

Variabilidad Persona Natural: Permite describir un actor que representa una persona.

Estructura:

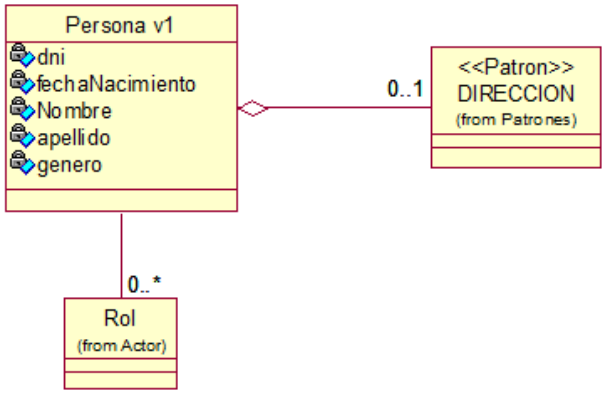

Figura 53 Estructura Variabilidad persona Natural

Presentaciones:

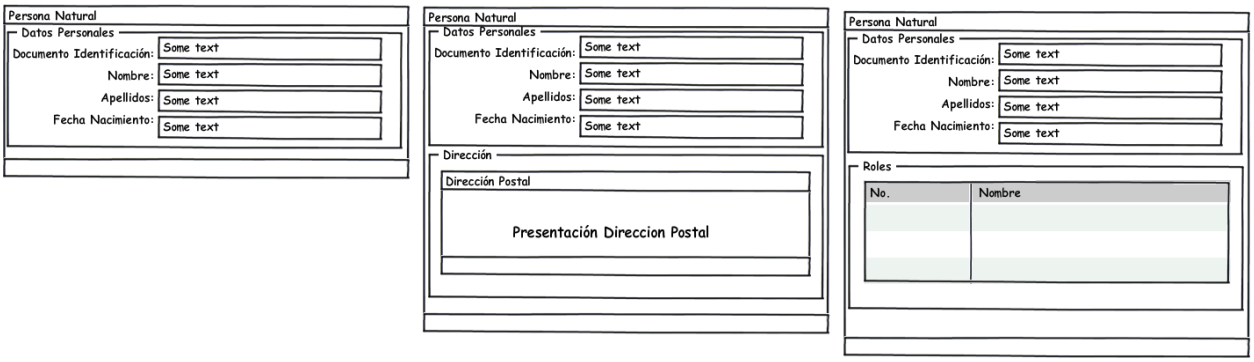

Figura 54 Presentaciones Variabilidad Persona Natural

Variabilidad Persona Jurídica: Permite describir un actor que representa empresa o entidad.

Estructura:

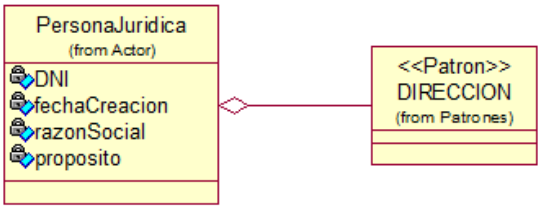

Figura 55 Estructura Variabilidad Persona Jurídica

Presentaciones:

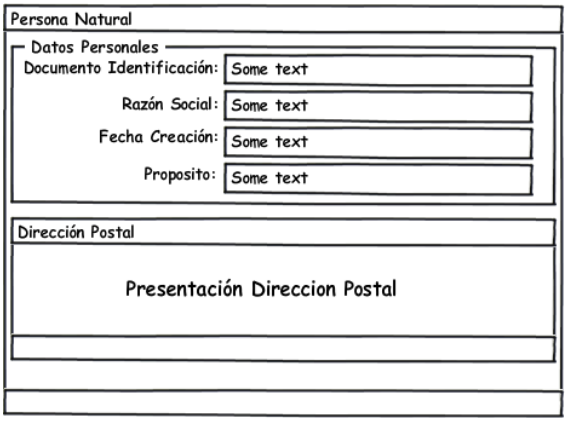

Figura 56 Estructura Variabilidad Persona Jurídica

6.5.5 Patrón Empleo

Nombre: Empleo

Clasificación: Quien

Propósito: Un empleo surge de la relación contractual entre una persona y una organización, en donde se indica factores como el contrato, tiempo de duración de la vinculación, las funciones o responsabilidades adquiridas. En este patrón se separan y se organizan estos conceptos para manejar de manera flexible la información de un empleo. Se busca solucionar los problemas de modelado que intentan indicar las condiciones generales entre un empleado y su empleador, asimismo, soportar situaciones en donde una persona puede tener varios puestos de trabajos o varias vinculaciones diferentes.

Aplicabilidad: Este patrón se puede aplicar en cualquier contexto donde exista una relación de empleado y empleador entre una persona y una organización. Por ejemplo, en contextos de negocios donde existe la figura de contratos de trabajo.

Estructura:

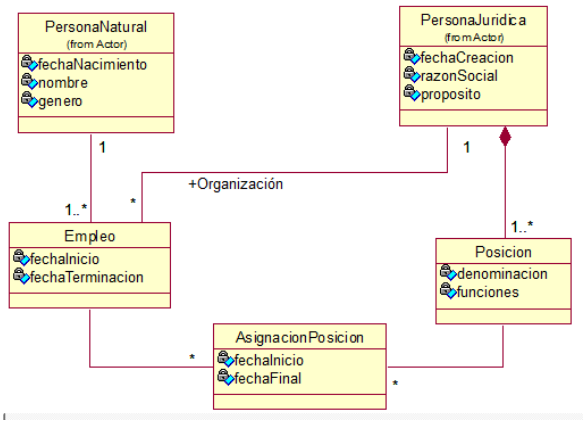

Figura 57 Patrón Empleado.

Participantes:

PersonaNatural: Es una subclase del patrón Actor, representa un humano. La persona natural es quien interactúa con una organización (*PersonaJuridica)* que son sistemas artificiales.

PersonaJuridica: Es una especialización del patrón Actor, representa una organización o empresa, el propósito de una organización es estructurar los recursos dentro de un negocio. Su composición interna,

es decir, sus unidades y sub-organizaciones no se presenta en este patrón, ya que esos conceptos son descritos por el patrón Unidad Organizacional.

Empleo: Representa la relación de empleo entre una persona natural y una jurídica, esta relación se genera por un periodo de tiempo, por eso los atributos de fecha de inicio y terminación.

Posición: Indica la responsabilidad o cargo asignado al empleo por la organización. Las posiciones son definidas por la organización, la cual define su denominación y funciones. Una posición puede ser asignadas varias personas a través del tiempo y del mismo modo una persona natural puede llegar a tener muchas posiciones.

AsignacionPosicion: Expone la relación entre la persona natural y la posición, incluyendo la fecha de inicio y de finalización se su asignación.

Ejemplo: Un programa académico de Ingeniería de sistemas y computación de una institución Universitaria está conformado por un director (posición), docentes(posición) y una secretaria(posición). El director del programa también es un docente, y tanto el como los demás cargos pueden cambiar de persona con el tiempo. Con el uso de este patrón se puede modelar cada una de las personas que son empleadas para esos cargos, por ejemplo, Juan es el director del programa durante el 1 de enero de 2014 y el 31 de diciembre del 2015, en el mismo periodo fue docente de algunos espacios académicos, en este caso la instancia de la persona natural Juan tendría un empleo con esas fechas de inicio y de finalización, y este empleo estarían asignadas los cargos de director y docente.

Patrones relacionados: Este patrón está relacionado con el patrón actor para representar la persona y organización involucrada en el empleo. Asimismo, Se puede complementar con el Patrón unidad Organizacional para representar la estructura de la organización e indicar por cada posición a que dependencia o unidad de la organización pertenece.

Fuente: Este patrón es basado en el patrón Employment presentado por Hans-Erik Eriksson and Magnus Penker en el libro Business Modeling with UML: Business Patterns at Work.

Variabilidades: Las variabilidades que se presentan de este patrón son extensiones que se le realizan: Empleo y Empleo supervisado, indicar que actor (persona o rol) supervisa las labores del empleado.

Variabilidad Empleo Simple: Permite Representar un empleado sin supervisión, este realmente no es una variabilidad, corresponde al patrón mismo.

Estructura:

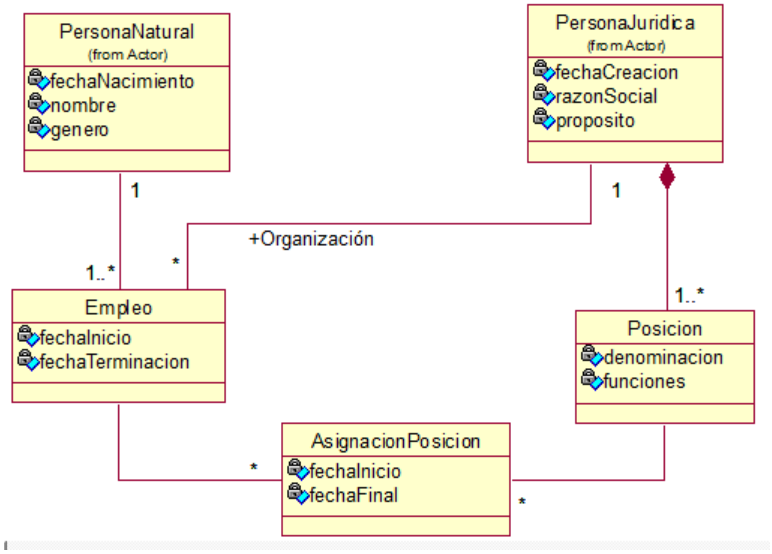

Figura 58 Estructura Variabilidad Empleo Simple

Presentaciones:

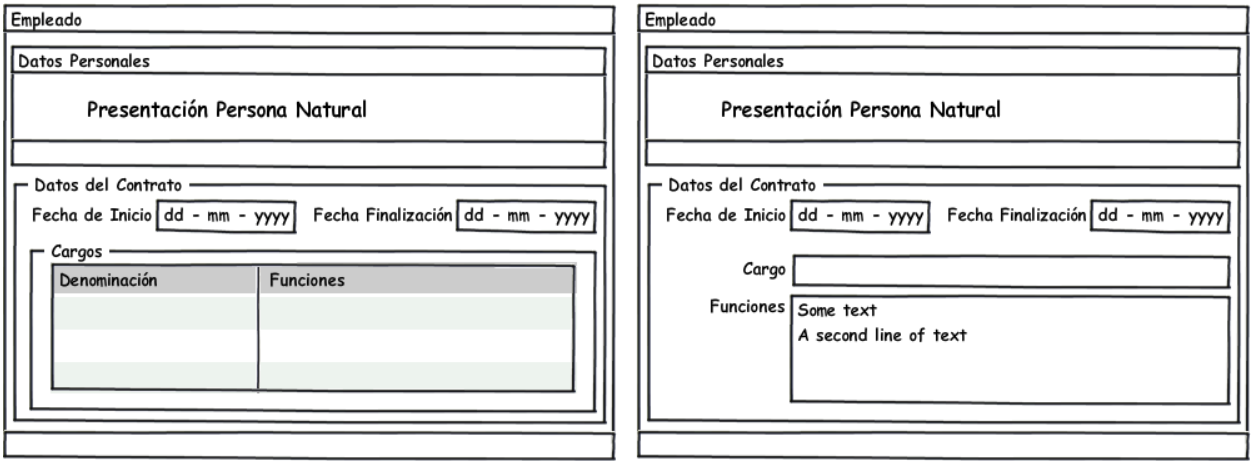

Figura 59 Presentación Variabilidad Empleo Simple

Variabilidad Empleo Supervisado: permite definir un empleo en el cual sea necesario especificar el jefe o supervisor el contrato.

Estructura:

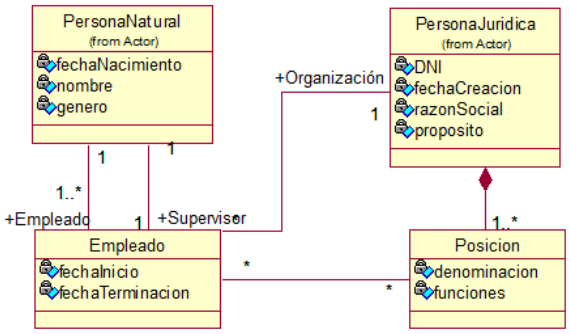

Figura 60 Estructura Variabilidad Empleo Supervisado

Presentaciones:

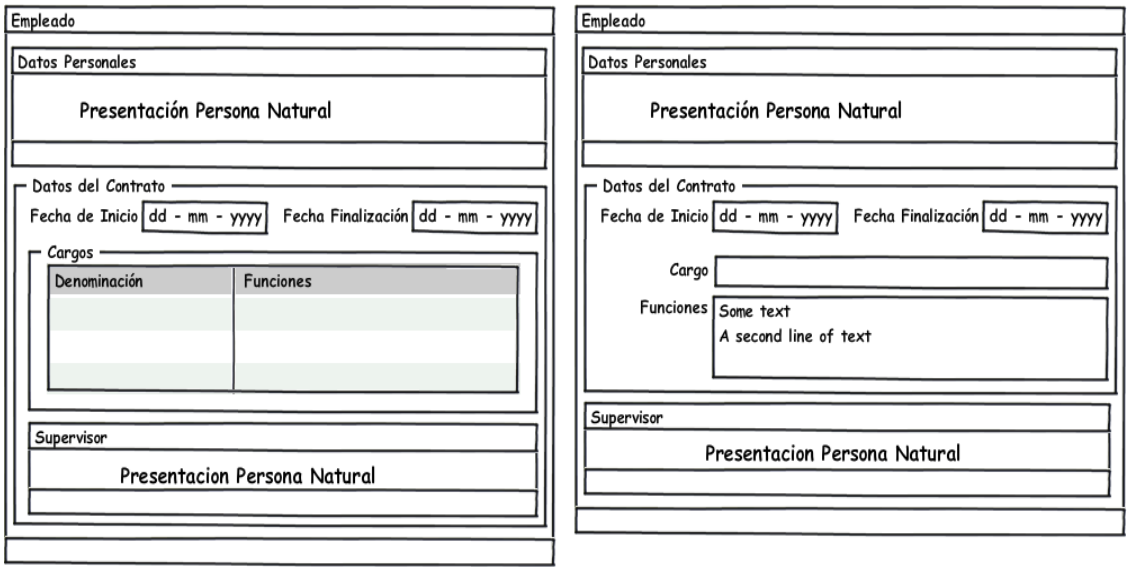

Figura 61 Presentaciones Variabilidad Empleo Supervisado

6.5.6 Patrón Producto

Nombre: Producto

Clasificación: Que

Propósito: Un producto es un concepto que representa algo que se negocia o se vende, puede ser un elemento físico como un computador o un elemento no físico como una cuenta bancaria o un servicio. Un

producto puede estar compuestos de otros productos, por ejemplo, un computador puede tener monitor, teclado, CPU. Este concepto es vital en las organizaciones, ya que este concepto envuelve todo lo que ellas ofrezcan (negocien) a sus clientes.

Aplicabilidad: Este patrón puede ser aplicado en cualquier situación donde se deba describir un producto o elemento que se debe producir, ofrecer, vender o comprar. Ejemplo de dominios de uso de este patrón es cualquier tipo de tienda o empresa que venda productos, una organización educativa que ofrece cursos de capacitación, estos constituyen los productos que ella ofrece.

Estructura:

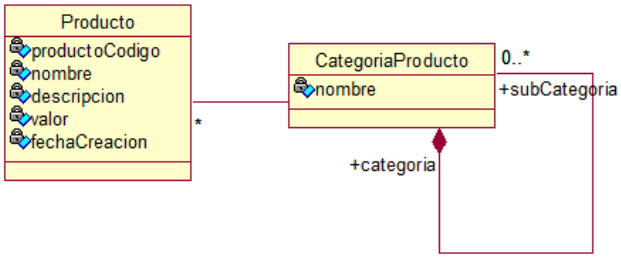

Figura 62 Patrón Producto

Participantes:

Producto: Clase que representa un elemento físico o servicio. Un producto tiene un código y nombre que sirven para identifican, una descripción que lo detalla, su valor o costo y la fecha en la cual fue creado.

CategoríaProducto: Los productos suelen agruparse en categorías, que permite una mayor organización y clasificación, una categoría puede tener una estructura jerárquica.

Ejemplo: Una Institución Universitaria cuenta con programas académicos universitarios, tecnológicos y posgrados, a nivel universitario ofrece ingeniería de sistemas e ingeniería civil, a nivel tecnológico desarrollo de software y en posgrado maestría en ingeniería. En este caso, haciendo uso del patrón se puede representar los niveles de formación con clase categoríaProducto y los distintos programas con la clase producto.

Patrones relacionados: Este patrón puede relacionarse con el patrón Maestro-Detalle, para representar el detalle por ejemplo de una factura, cotización. Puede ser complementado por el patrón Actor para indicar quien es la persona natural o jurídica que lo provee o produce.

Fuente: Este patrón es basado en el arquetipo Product descrito por Michael Blaha en el libro Patterns of Data Modeling.

Variabilidades: Las variabilidades presentadas para este patrón son producto simple y producto compuesto.

Variabilidad Producto Simple: Permite definir un producto simple, es decir, que representa un todo, no se subdivide en partes.

Estructura:

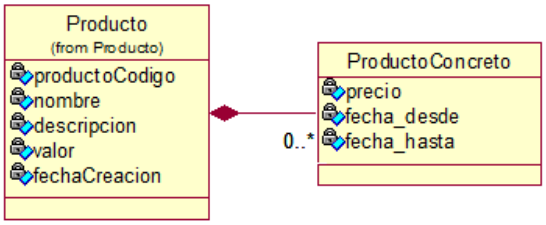

Figura 63 Estructura Variabilidad Producto Simple

Presentaciones:

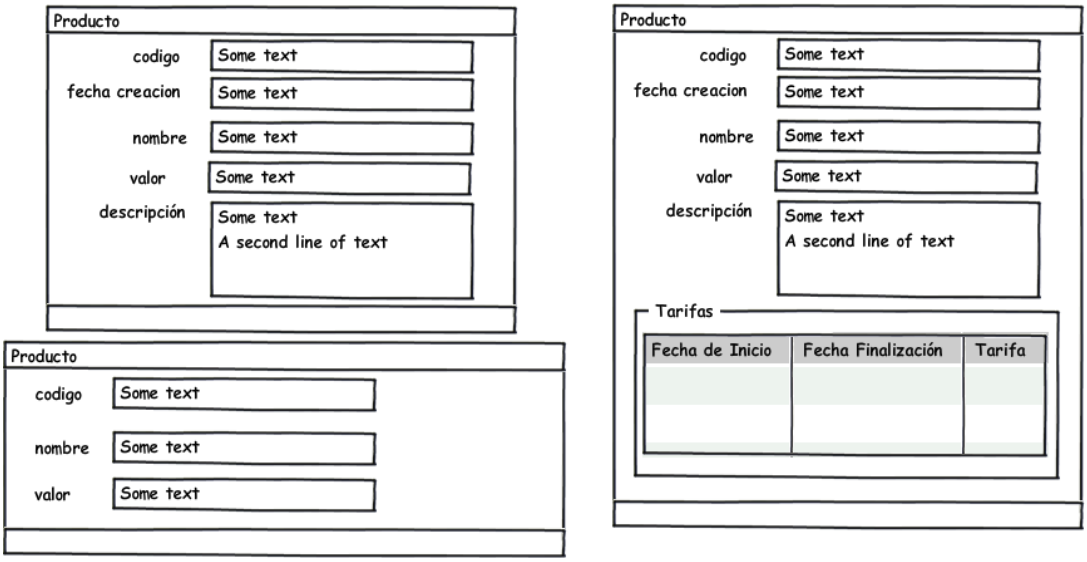

Figura 64 Presentaciones Variabilidad Producto Simple

Variabilidad Producto Compuesto: Permite definir un producto que se compone de otros productos

Estructura:

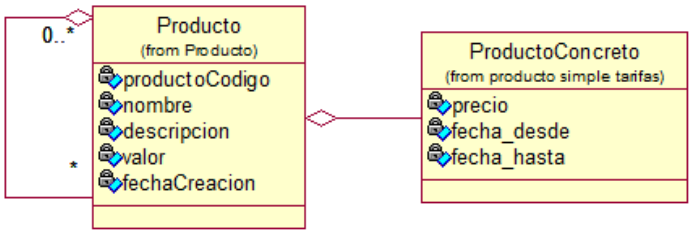

Figura 65 Estructura Variabilidad Producto Compuesto

Presentaciones:

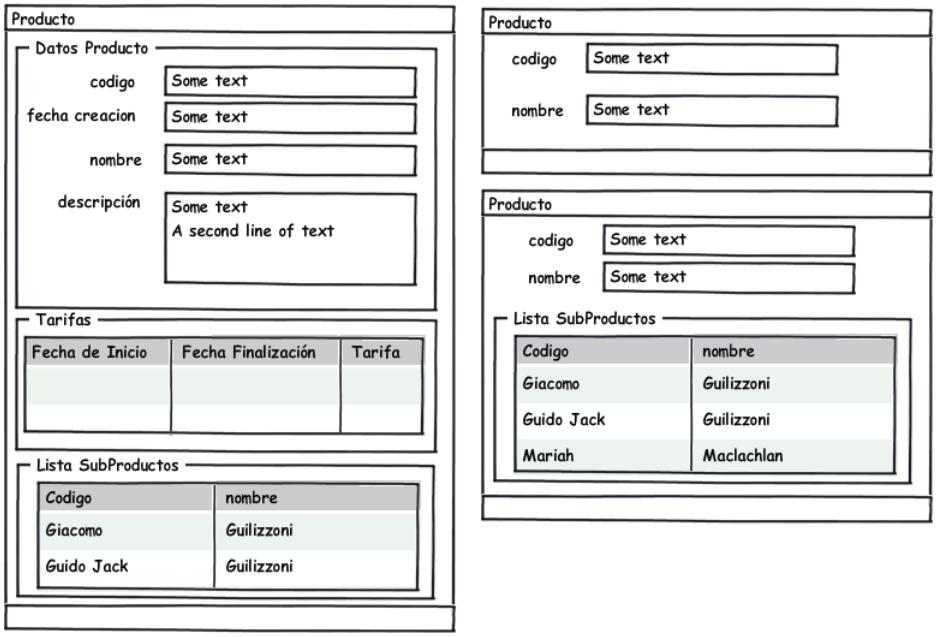

Figura 66 Presentaciones Variabilidad Producto Compuesto

6.5.7 Patrón Pago

Nombre: Pago

Clasificación: Como

Propósito: Un pago es la asignación de dinero a cambio de algo de valor. Hay diferentes tipos de pago, cada uno con sus propios datos asociados por ejemplo un cheque está escrito contra una cuenta bancaria.

El propósito de este patrón es separar el concepto del monto del valor a pagar del método usado, facilitando la definición de este concepto de manera independiente al objeto que lo genera.

Aplicabilidad: Este patrón puede ser aplicado en cualquier situación o transacción que genere la realización de un pago, ejemplo un servicio, una factura.

Estructura:

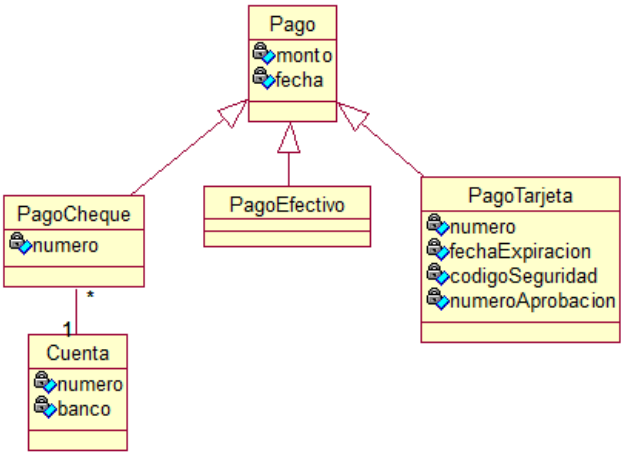

Figura 67 Patrón Pago

Participantes:

Pago: Clase abstracta que contiene los datos genéricos de cualquier pago, la fecha y el monto a pagar.

PagoCheque: Clase que especializa el pago en forma de cheque, está asociado a la clase Cuenta que indica la cuenta bancaria asociada al cheque.

Cuenta: Representa la cuenta bancaria que da respaldo a los fondos involucrados en el pago a través de cheque.

PagoTarjeta: Especializa el pago con tarjeta de crédito.

Ejemplo: Un ejemplo de la aplicabilidad de este patrón es en una empresa comercializador que debe generar una factura, registrando en ella la forma de pago que el cliente utiliza.

Patrones relacionados: Este patrón puede relacionarse con el patrón orden – Detalle.

Fuente: Este patrón es basado en el arquetipo Product descrito por Michael Blaha en el libro Patterns of Data Modeling.

Variabilidades: Este Patrón no tiene variabilidad, ya que un pago puede realizarse a través varios métodos distintos

6.5.8 Patrón Maestro Detalle

Nombre: Maestro Detalle

Clasificación: Como

Propósito: Este patrón tiene como propósito modelar cualquier concepto que se componga de una cabecera y un listado, donde la cabecera representa el termino maestro, al cual se le asocia un listado de detalle o ítems. Soporta el modelado de cualquier documento o transacción que cumpla esta característica.

Aplicabilidad: Este patrón, puede ser aplicado a contextos donde a partir de un concepto principal se obtiene o se relacionan muchos otros conceptos, por ejemplo, una factura donde hay una cabecera, datos generales de la factura y a partir de esta se requiere un listado de productos.

Estructura:

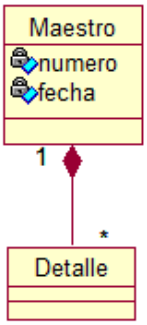

Figura 68 Patrón Pago

Participantes:

Maestro: Clase que representa el concepto principal que agrupa un listado, por ejemplo, la cabecera de una transacción que permite identificarla.

Detalle: Clase constituye el detalle o ítems que están componen los elementos involucrados en una transacción.

Ejemplo: Este patrón puede ser aplicado a cualquier concepto que represente una transacción en donde exista una relación de uno a muchos, por ejemplo, una factura la cual se compone de muchos productos involucrados en la venta o compra.

Patrones relacionados: Este patrón puede relacionarse con el patrón producto para especificar los ítems asociados al maestro; con el patrón Actor, para indicar los participantes involucrados en una transacción.

Fuente: Propia.

Variabilidades: Este patrón puede tener múltiples adaptaciones, en este catálogo solo se muestra la variabilidad Orden Detalle.

Variabilidad Orden Detalle: Esta variabilidad permite describir cualquier soporte de una transacción que tenga una cabecera identificando los participantes (Emisor y receptor) y un detalle en el cual se lista los elementos que la conforman (Productos), por ejemplo, Facturas, Cotizaciones.

Estructura:

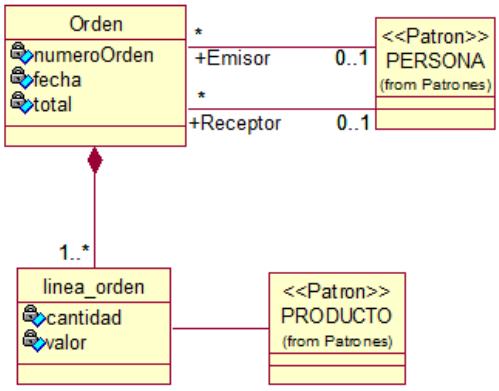

Figura 69 Estructura Variabilidad Factura

Presentaciones:

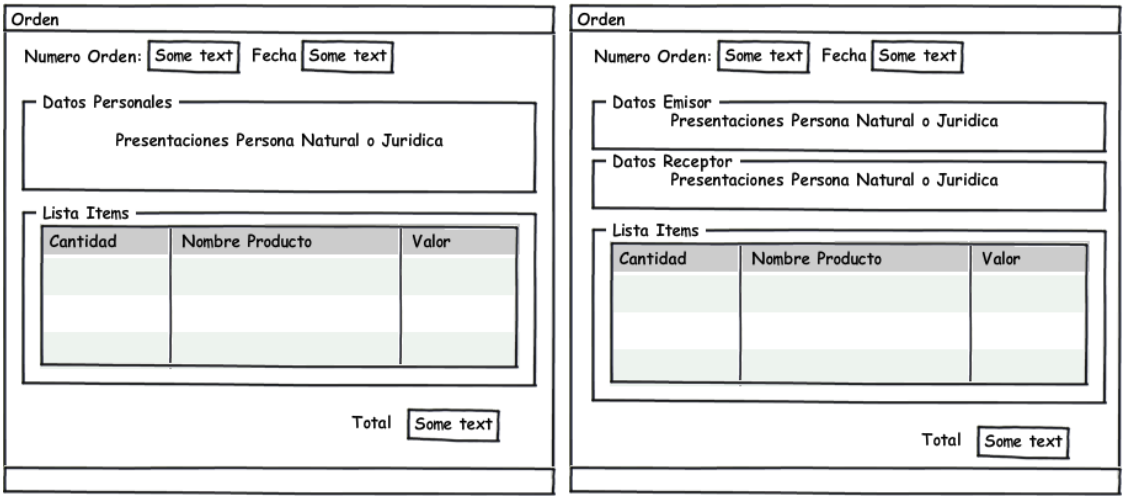

Figura 70 Presentaciones Variabilidad Orden Detalle

CAPÍTULO SÉPTIMO

7. MÉTODO PROPUESTO PARA EL DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO

En esta sección se presenta la propuesta para desarrollo de la interfaz de usuario de negocio, que se fundamenta metodológicamente en el nivel de modelado del negocio de la metodología TD-MBUID, en la cual se realiza el Diseño del dominio de negocio (Business Domain Design) y el Modelado de la interfaz de usuario de negocio (Business User Interface Modeling).

Como se explicó en el capítulo 3.3, Para diseñar la interfaz de usuario de negocio y promover la reutilización se debe identificar, con ayuda de los usuarios, las tareas más importantes y frecuentes del sistema y los datos que necesitarían para realizar dichas tareas, esto se logra interpretando el modelo mental que el usuario tiene de sus tareas y de los datos que necesita para realizarlas. En consecuencia, la eficiencia de los resultados de este proceso depende de la habilidad y experiencia del equipo de desarrollo, constituyendo esto un riesgo importante en la calidad de las propuestas de interfaz de negocio generadas.

Este método propone un proceso que busca disminuir el grado de incertidumbre o subjetividad en la adquisición del modelo mental del usuario, aplicando patrones de negocio; los cuales representan experiencia probada y documentada; que describe las entidades y datos que comúnmente se involucran en tareas a nivel de negocio, es decir que son comunes a contextos similares, aunque se trate de dominios distintos, por ejemplo, cliente y ubicación geográfica, son conceptos universales en la mayoría de negocios, por ejemplo, en una tienda, un hotel, una universidad.

Teniendo en cuenta que en TD-MBUID, el proceso de *modelado de la Interfaz de Usuario de Negocio* se busca agrupar los datos comunes dentro de unas pocas ventanas y que en caso de que una tarea requiere una ventana de contexto ya identificada, simplemente se reutiliza dicha ventana. En este proceso de modelado se tiene como principio buscar y promover la reutilización de las ventanas de contexto (se plantea en este trabajo la asociación de plantillas de presentación a cada patrón de negocio) las cuales soporten los datos del dominio del patrón en diferentes contextos.

La reutilización de las plantillas de presentación permite que para distintas tareas que están en cierto modo interrelacionadas y, que en consecuencia podrían compartir la misma o parte de la información de contexto exista el conjunto de ventanas que las materialicen. En consecuencia, cada vez que se instancie o se combine un patrón para formar el dominio de una tarea, la interfaz asociada a cada uno de ellos se combinará también para formar la interfaz de usuario de negocio.

En la Figura 71 Ejemplo reutilización de la ventana de contexto, se muestra un ejemplo de reutilización mencionada en el párrafo anterior, se puede observar que la ventana de contexto de Dirección Postal es reutilizada en la interfaz de negocio de la Persona, ya que en el contexto de esta última se requieren la dirección donde vive, y por lo tanto existiendo ya la BUI para esto, simplemente se reutiliza.

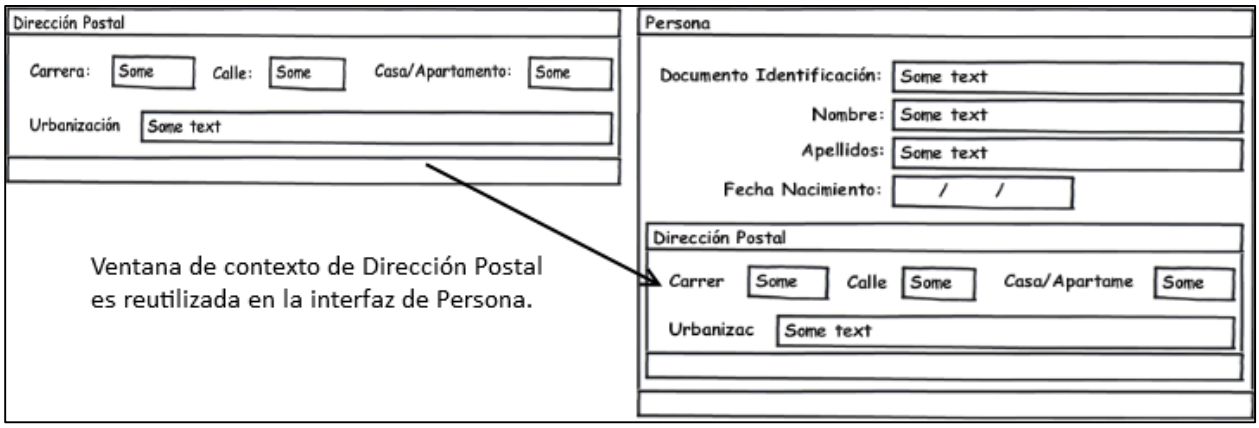

Figura 71 Ejemplo reutilización de la ventana de contexto

Este trabajo propone un método que permita la aplicación de los patrones (datos + presentación + interacción) en el diseño y en la generación de la interfaz de usuario de negocio, en la Figura 72 se muestra los componentes que conforman esta propuesta; el proceso metodológico, un lenguaje que especificación la notación utilizada y una herramienta que soporta la realización de las tareas definidas en la metodología propuesta.

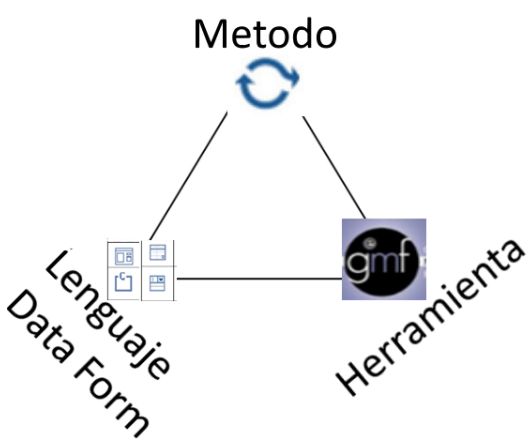

Figura 72 Elementos que Componen la propuesta

El componente metodológico abarca desde la forma de identificar e incorporar nuevos patrones hasta la forma de utilizarlos y generar la BUI, el lenguaje que soporta la metodología es el lenguaje de DataForm, la herramienta constituye el apoyo tecnológico que permite generar la interfaz gráfica de acuerdo a la metodología definida y haciendo uso del lenguaje. La combinación de estos tres componentes busca ofrecer a un desarrollador los elementos necesarios para la utilización de esta propuesta.

Esta propuesta apoya el diseño de la interfaz de usuario de negocio mediante el uso de herramientas de modelado en las cuales se pueden incorporar datos de prueba y un conjunto de transformaciones que permiten obtener una interfaz final. Es posible especificar un diseño orientado a una plataforma concreta en este nivel, si se tienen en cuenta ciertas restricciones iniciales o si se conoce cuál es la plataforma que mejor puede soportar el diseño. En este caso es indispensable diseñar las transformaciones necesarias para la generación en dicha plataforma, si no están disponibles. Una vez definidos los prototipos se pueden mostrar las ventanas de contexto a los usuarios. Estas evaluaciones pueden ser fructíferas puesto que los prototipos de las ventanas pueden ofrecer datos reales y detalles gráficos.

El objetivo del proceso metodológico y de la herramienta de soporte que permita la generación prototipos funcionales es facilitar la validación inmediata con el cliente, es decir, que en tiempo real se pueda presentar al usuario una propuesta de interfaz gráfica verificando que en ella se capture el modelo mental que el posee de los datos que necesita.

7.1 MÉTODO DE DESARROLLO DE LA INTERFAZ DE USUARIO DE NEGOCIO

En esta sección se describen las características principales de la propuesta de desarrollo de la interfaz de usuario, se especifica cada una de las fases que conforman el método de generación automática de la interfaz de usuario de negocio haciendo uso del catálogo de patrones de negocio propuesto, se pretende

proponer para la generación de la interfaz de usuario de negocio un enfoque de ingeniería basado en la triada (patrón de datos – plantilla de presentación – modelo de interacción).

En la Figura 73 se presenta el flujo de desarrollo del método propuesto para la generación de la Interfaz de Usuario de Negocios a partir de patrones de negocios; como se puede observar el flujo se divide en 2 niveles, el primero (color verde) representa las actividades relacionadas con la análisis del dominio y selección de patrones, el cual inicia analizando el contexto para entender el dominio y contexto del problema, a partir de este se seleccionan del catálogo de patrones de negocios las variabilidades y las presentaciones que permitan soportar los datos que el usuario requiere para la realización de las tareas de acuerdo al contexto del problema y por último adaptarlos de acuerdo a las particularidades especificas del modelo mental el usuario, es decir, realizar los ajustes a los datos y presentaciones propuestas en los patrones de negocios seleccionados. El segundo nivel (color anaranjado) corresponde a las tareas necesarias para preparar los prototipos de interfaz de usuario de negocios para ser validada de una manera realista con el usuario, la finalidad de estas tareas es realizar de manera temprana, casi inmediata, la evaluación de los prototipos, buscando disminuir el impacto y costos de posibles correcciones y, asimismo, mejorar la calidad de interfaces de usuario final.

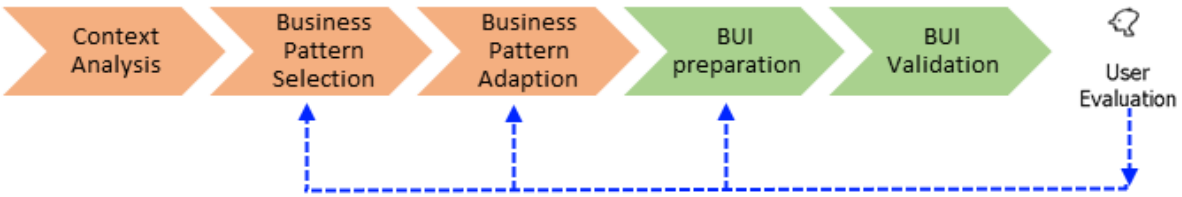

Figura 73 flujo de desarrollo del método propuesto para la generación de la Interfaz de Usuario de Negocios a partir de patrones de negocios

A continuación, se describen cada uno de las tareas que componen el método:

7.1.1 **Análisis del contexto (Context Analysis)**

En este paso se estudia el dominio de la aplicación, para identificar el contexto del problema, es decir, identificar al usuario, las tareas que debe realizar y los conceptos de negocio y datos que son manipulados para realizarlas. El resultado de este paso es una descripción general de los conceptos de negocio (Entidades) y los datos del contexto del problema. En esta actividad no se intenta realizar el modelo de dominio, se busca identificar los conceptos principales de negocio asociados al dominio del problema y el contexto particular donde se presentan.

7.1.2 Selección del patrón de negocio (Business Pattern Selection)

En este paso se lleva a cabo la selección de los patrones de negocio, que dan soporte a los conceptos obtenidos en la *Análisis del contexto*. En esta fase se debe seleccionar los patrones de acuerdo a los conceptos de negocio identificado, definir para cada patrón la variabilidad que se ajuste al contexto del problema y de acuerdo a los datos que el usuario requiere para realizar la tarea seleccionar la plantilla de presentación asociada.

Para esta tarea se debe tener por cada patrón de negocio un árbol de variabilidad a través del cual se visualiza las distintas opciones (variabilidades del patrón) que ofrece un patrón para seleccionar la variabilidad necesaria y a partir de este se determina el Business domain data y el DataForm.

Esta tarea genera la versión inicial de: *Business Domain Data*, el cual representa las entidades de negocio y los datos que soportan las tareas del negocio; y El DataForm*, el cual* constituye la estructura de los datos que actúa como persistencia y lógica de la *plantilla de presentación.* Asimismo, esta tarea genera o adapta el modelo de dominio, adicionando o modificando las entidades, atributos y relaciones que conforman el patrón seleccionado.

7.1.3 Adaptación del patrón de negocio (Business Pattern Adaption)

Los patrones por su naturaleza pueden ser adaptados a situaciones concretas de un problema en específico; aunque en el catálogo de patrones de negocio se definen posibles variaciones para soportar distintos contextos, pueden surgir casos donde se requieran datos específicos que no fueron contemplados, por lo tanto, en esta tarea se debe validar que el patrón soporte completamente las tareas y datos del usuario, y de ser necesario agregar o quitar datos que no estén representados, modificando el D*ata Form* y el *Business Domain Data* que se obtiene de la tarea anterior.

7.1.4 Preparación de la interfaz de usuario de negocio (BUI preparation)

En este paso se lleva a cabo la implementación de las producciones y de la interacción del prototipo de la BUI para proceder a su generación. La interacción se refiere a realizar los aspectos de lógica de programación que permitan exponer funcionalidades básicas en los prototipos a generar, que permitan definir aspectos básicos de interacción, por ejemplo, filtrar un el listado de departamentos de acuerdo al país seleccionado, las producciones se refiere a generar un conjunto de datos reales de prueba, instancias de las clases del *Business Domain Data* que soporta la interfaz, es decir, generar datos reales de los conceptos del negocio involucrados; esto se realiza con el fin de poder realizar una validación temprana con el cliente. Las pantallas generadas se lleva a cabo a partir de los prototipos de presentación asociado

a las variabilidades redefinidos en la *sección 7.1.3*, los cuales dan soporte al modelo de datos y las descripciones de las tareas.

Al finalizar esta tarea, se genera la renderización de la plantilla de presentación asociada a la variabilidad, este proceso da como resultado la generación a partir de la triada que representa la variabilidad, el model, el view model y la view, este último representa el prototipo de interfaz de usuario de negocio, el cual posteriormente se debe validar con el usuario.

7.1.5 Validación de la Interfaz de usuario de negocio (BUI validation)

En esta etapa se evalúan las BUI generadas. Dicha prueba se podrá hacer de muchas maneras; por ejemplo, aplicando pruebas de comprensibilidad (una prueba de usabilidad simplificada), se verifican los accesos e interacción a los elementos de las pantallas, los datos que se requieren para llevar a cabo cada tarea. Este paso a menudo da lugar a cambios en las descripciones de las tareas y en el modelo de datos.

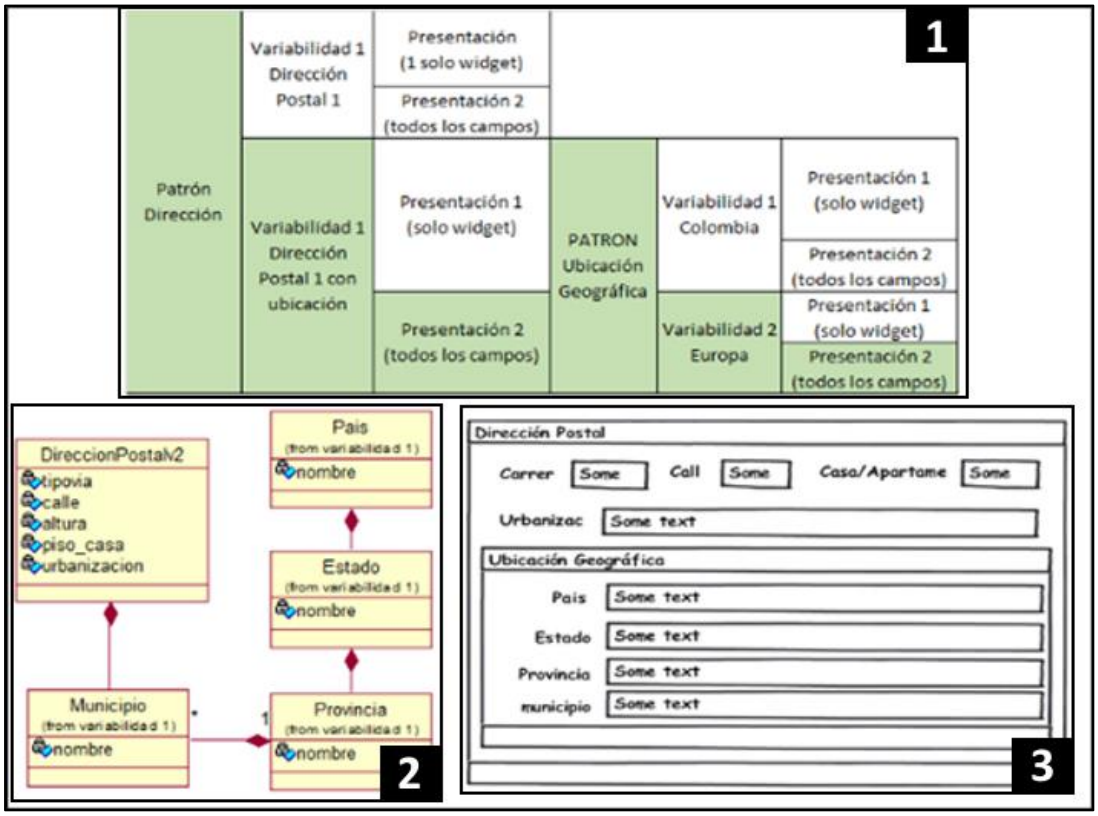

Figura 74 Ejemplo aplicación de Patrones de Negocios

La Propuesta de Interfaz de usuario de negocio se genera de manera automática basándose en el Bussines Domain Data, el Data Form y el Paper Form previamente definidos. En la Figura 74 se puede observar un ejemplo simplificado de la aplicación de los patrones de negocios para la generación de la BUI, en la parte superior (1) de la figura se resalta en color verde los Patrones de negocios, sus variabilidades y presentaciones asociadas seleccionadas. Con base a la variabilidad del patrón seleccionado, se genera el Business *domain data* (2) que da soporte a los datos de los patrones usados, asimismo en la parte inferior derecha (3) se observa la plantilla de presentación que junto con el modelo de interacción (implementado en el DataForm) genera el prototipo de interfaz de usuario que brinda la visualización de los datos. Se puede distinguir que la propuesta de BUI está conformada por una de las presentaciones de la *variabilidad dirección Postal* del *patrón dirección*, combinada con una ventana de contexto de la *variabilidad Europa* del *patrón Ubicación Geográfica*.

Lo que se busca con este proceso es el diseño de la interfaz de usuario basado en los patrones de negocio, de tal manera que permita la generación automática de la interfaz a nivel de negocio casi en tiempo real, a medida que se realiza su modelado.

7.2 ARTEFACTOS DEL SISTEMA

En esta sección se presenta el conjunto de artefactos necesarios para el desarrollo de la interfaz de usuario (Figura 75). Estos artefactos promueven un diseño dentro del paradigma del MBUID orientado en la búsqueda de la generación automática de la interfaz de usuario, asimismo, buscan cumplir con los fundamentos metodológicos expuestos en TD-MBUID.

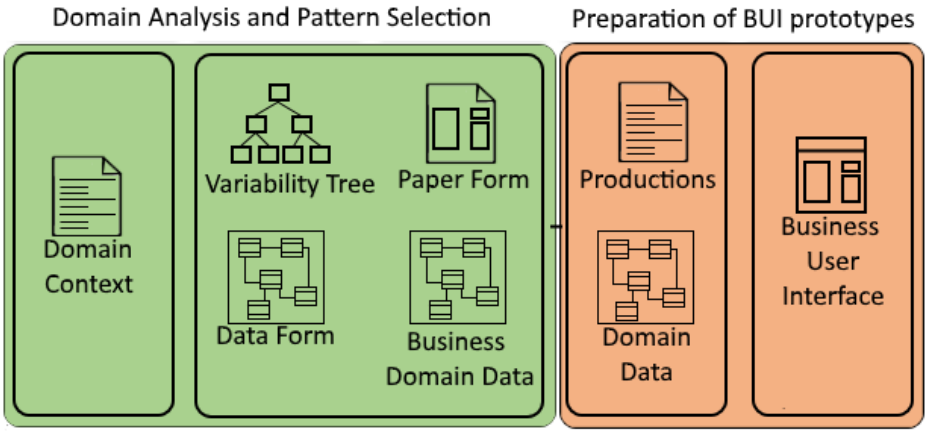

Figura 75 Artefactos para el desarrollo de la interfaz de usuario de negocio

A continuación, se presenta una descripción de los artefactos involucrados en el desarrollo de la interfaz de usuario de negocio:

- *Contexto de Dominio (domain context)*: este artefacto contiene abstracciones relativas a la información contextual que describe el dominio del problema y de las tareas asociadas a este contexto, se busca describir los principales conceptos del negocio involucrados en el problema. En esta propuesta, el modelado de la información de contexto se recopila fuera del lenguaje de modelado de manera textual, es decir, no se han incorporado nuevos elementos de modelado para tal fin.

- Árbol de Variabilidades (variability tree): Este componente representa el catálogo de patrones. Un catálogo es un conjunto de patrones relacionados entre sí y clasificados por algún criterio, los cuales pueden ser utilizados conjuntamente o de manera independiente. La finalidad de este componente es facilitar la búsqueda y selección de los patrones de negocio, asimismo, definir y limitar las relaciones o combinaciones soportadas entre ellos (variabilidad en datos y presentaciones).

- Plantilla de presentación (Paper Form): Este componente representa describe la estructura de la presentación, ligada a la semántica de los datos, muestra un diseño visual de manera realista de los Business Domain data asociado a un patrón, la renderización de este componente constituye la interfaz de usuario de negocio. Este componente se determina desde la perspectiva del usuario, ya que se busca representar solo las entidades y la porción de los datos que el usuario necesita de ellas para realizar la tarea, por lo tanto, de un mismo Business domain data pueden surgir diversos Paper Form.

- *Forma de los datos (Data Form)*: Este componente representa los datos y la forma de la interfaz de usuario, separa los datos del dominio de los datos de la interfaz, puesto que podemos tener fragmentos de los datos del Business domain data que son de interés para el usuario y por lo tanto serán mostrados en la interfaz, representado la información que será visualizada y manipulada en la interfaz de usuario independiente de la manera en cómo es modelada en el dominio, por lo tanto, cada Paper Form debe tener asociada un Data Form.

- *Datos del dominio del negocio (Business domain data):* Este componente está conformado por las entidades de negocio, que representan los datos que soportan las variabilidades de los patrones de negocios definidos. El Business domain data representa la estructura de cada una de las variabilidades de los patrones de negocio. Para su representación se usan los diagramas de clases de UML, ya que es el estándar *de facto* en el modelado de los diagramas de clase.
- *Modelo del dominio (domain model):* el modelo de dominio captura la semántica del contexto del sistema y define los requisitos de información para el desarrollo de la interfaz de usuario. Especifica los datos que las personas utilizan, relacionados con las entidades del mundo real. El análisis del dominio es parte de la mayoría de los enfoques de desarrollo, y no es algo específico a considerar en el diseño de la interfaz de usuario. Los objetos del dominio son considerados como instancias de clases que representan los conceptos que son manipulados y que son totalmente independientes de la forma en que se muestran en la pantalla y cómo se almacenan en el ordenador. Se usan los diagramas de clases UML como base para la representación del modelo de dominio, ya que es el estándar *de facto* en el modelado de los diagramas de clase.

- *Prototipo(Prototype):* Un prototipo es una implementación parcial de un sistema o una parte del mismo. Un prototipo explora algunos elementos aislados de un sistema, como por ejemplo un nuevo algoritmo, un modelo de interfaz de usuario, o un esquema de base de datos (Orozco, 2010). Un prototipo de interfaz de usuario es un ejemplo que se construye con el fin de explorar y/o validar el diseño, generalmente son mucho más fácil de entender para un usuario, de tal forma que su uso puede evitar una gran cantidad de soluciones de diseño deficientes en etapas tempranas del desarrollo. El principal objetivo de la creación de un prototipo es que sea capaz de probar el diseño, incluyendo sus características de usabilidad, antes de que el desarrollo real comience (IBM_Rational, 2003).

- *Producciones (Productions):* Este componente representa el conjunto de datos que serán usados y visualizados en los prototipos, su finalidad es poder visualizar a través de la propuesta de interfaz de manera realista datos reales, permitiendo una validación rápida con el usuario. El termino producción hace referencia a instancias de las clases que conforman el Domain Model.

- *Interfaz de usuario de negocio (Business User Interface)*: Corresponde a una parte del modelo de la interfaz de usuario; es la información que el usuario debe imaginar que está detrás de la pantalla física - en la parte posterior del sistema (Lauesen, 2005). Normalmente se dibuja esta interfaz, en términos de elementos de interfaz de usuario concretos (pantallas en papel) debido a que se asemejan a las pantallas finales en el ordenador, con detalles gráficos y con contenidos realistas de los datos, este componente representa la renderización del Paper Form.

7.3 LENGUAJE DATA FORM (DFL)

DFL(Lenguaje DataForm) es el lenguaje definido para la creación de modelos DataForm y propuesto por (Muñoz & Orozco, 2014), es un lenguaje gráfico para visualizar y construir modelos DataForm basados en los elementos de modelado asociados a la Forma (Interfaz de Usuario) y a los Datos (Datos de la Interfaz), captura aspectos relacionados con el layout, tamaño de los componentes gráficos de la interfaz, tipos de elementos gráficos como por ejemplo tablas, textos, combos, botones, etiquetas, contenedores; también describe la semántica de los datos: captura aspectos como atributos, métodos, relaciones de contención, de agregación, de asociación, de dataBinding.

En la Tabla 3 : Notación DataForm, se observa la descripción de los elementos que hacen parte de la notación que conforma el lenguaje DFL, dicha notación es creada a partir de la identificación de los elementos que intervienen en la creación de la interfaz de usuario y su modelo de datos correspondiente.

Icono	Nombre del Elemento	Descripción
	View	La view representa la interfaz de usuario la cual contiene todos los elementos (contenedores, TextView, etc.).
	Contenedor	Este elemento representa un container en la interfaz de usuario y es representado como una clase en el modelo de datos que contiene otros elementos.
	TableView	Un tableview es el elemento que representa una tabla en la interfaz de usuario y muestra una lista de elementos creados en el modelo de datos.
	ComboView	Un comboView es elemento que representa un combo en la interfaz y muestra una lista de elementos.
	TextView	Un textView representa un campo de texto en la interfaz y es un atributo en el modelo de datos.
	DateView	Un dateView representa un campo de tipo date en la interfaz y un atributo en el modelo.
	FilterView	Un filterView representa un campo de texto en la interfaz que permite realizar un filtro de búsqueda a una tabla.
	Containment	Es una relación de composición por valor que representa una relación de todo-parte.
	Sharing	Es una relación de agregación por referencia.
	Binding	El binding es la relación de tipo “enlace” que existe entre un elemento de la interfaz y un elemento del modelo.
	Asociation	Es una relación entre elementos del DataForm.
	Trace	Es una relación que indica que un conjunto de datos es un subconjunto de otro.
	Atributo	Un atributo es elemento que corresponde a un campo de un objeto de los datos que es mostrado en las columnas de una tabla o en una lista de un comboView.

Tabla 3 : Notación DataForm (Muñoz & Orozco, 2014)

El DFL es el lenguaje definido para TD-MBUID que soportar la definición de la Interfaz de Usuario de Negocio, por lo tanto, define la notación para soporta el método propuesto en esta tesis, aunque el método asocia propuestas de BUI a los patrones y no directamente modelos DataForm, estos últimos son fundamentales pues definen como se mencionó, los datos y la forma de la interfaz y por lo tanto permiten y facilita el proceso y transformaciones para lograr su generación automática. En la

7.4 HERRAMIENTA DE SOPORTE

En esta sección se presenta la herramienta desarrollada, un aplicativo de software basada en modelos para apoyar a los diseñadores e ingenieros en la creación de patrones de negocios y modelos basados en la notación DFL. Esta herramienta permite la edición manual y generación automática de la Interfaz de usuario de negocios, facilitando la validación temprana de los prototipos generados al proveerle datos reales del dominio (conjunto de producciones de cada variabilidad de los patrones) que permiten exponer su funcionalidad de una manera realista. La Herramienta ha sido desarrollada dentro del marco de modelado de Eclipse que proporciona las herramientas para guiar el modelado de software mediante el uso de conceptos de metamodelado (Moore, Dean, Gerber, Wagenknecht, & Vanderheyden, 2004).

Una herramienta de soporte debe ofrecer un equilibrio entre flexibilidad y orientación, ya que mientras mayor sea la flexibilidad, el equipo de desarrollo tiene que decidir más por sí mismo cómo utilizar la herramienta, lo cual puede generar ambigüedades o confusiones. Es por esto que la herramienta tiene que ir acompañada de una metodología, o mejor aún, una herramienta debe implementar (ser compatible con) cierta metodología, la cual oriente o indique como realizar los procesos a través de ella. Son las personas que desarrollan software, quienes perciben una mejora de su productividad con el uso de herramientas.

La herramienta está implementada como un *plugin* de Eclipse mediante el uso de EMF (*Eclipse Modeling Framework*) y GMF (*Graphical Editing Framework*). Eclipse se ha consolidado, hoy en día, en el área de desarrollo de aplicaciones debido, principalmente, a su capacidad de ampliar su funcionalidad mediante el uso de *plugins*. Eclipse no es un programa monolítico, sino más bien consiste en un pequeño núcleo denominado "*PlugIn Loader*" rodeado de cientos de *plugins*, cada uno de los cuales provee servicios a los demás (Steinberg, Budinsky, Paternostro, & Merks, 2009).

Eclipse proporciona soporte para MDE, por medio de EMF, en el desarrollo de modelos a partir de lenguajes. Un servicio de gran importancia para la realización de esta tesis que ofrece EMF son mecanismos para la persistencia de los modelos en forma de un archivo XML, los cuales permiten almacenar producciones (instancias de las clases) realizadas a partir de los elementos de los patrones de negocios, permitiendo probar los modelos realizados y en nuestro caso contar con un conjunto de datos de pruebas que permite mostrarlos de manera realista en los prototipos generados.

El Marco de Edición Gráfica (GEF) facilita el desarrollo de las representaciones gráficas de los modelos a partir de representaciones realizadas a través de *Draw2D*, como una adaptación de SWT que es el

estándar de dibujo de Eclipse. Mediante la combinación de GEF y EMF, es posible implementar un *plugin* dentro del marco de trabajo de Java para Eclipse. El objetivo de GMF es la definición de herramientas de modelado gráfico en múltiples dominios (R. Gronback & Roy, 2006).

7.4.1 Desarrollo de la Herramienta

El desarrollo de una herramienta a través de un proyecto *Graphical Modeling Framework* (GMF) se lleva mediante los siguientes pasos:

7.4.1.1 Definición de Metamodelo

En un proyecto GMF, el metamodelo define la sintaxis abstracta del lenguaje que se desarrolla. Esta sintaxis abstracta actúa como una guía para la comunicación de todos los artefactos y herramientas que intervienen en el desarrollo de una herramienta de soporte a MDE.

Una técnica común para la especificación de lenguajes es definir primero la sintaxis del lenguaje y luego describir su semántica estática y dinámica (OMG, 2008). La sintaxis define qué elementos de modelado existen en el lenguaje y cómo estos elementos de modelado se construyen en términos de otros. Cuando el lenguaje tiene una sintaxis gráfica, es importante definir la sintaxis en una notación de forma independiente (es decir, para definir la sintaxis abstracta del lenguaje). La sintaxis concreta se define entonces mediante un *mapping* entre la notación y la sintaxis abstracta.

El desarrollo del metamodelo (ver Figura 76) dentro de Eclipse se hace mediante un modelo *Ecore* en EMF, el cual permite refinar la estructura y la semántica usando el lenguaje de restricciones a objetos (OCL). Generalmente, los modelos vienen provistos de un lenguaje de definición basado en enfoques tradicionales como el BNF, sin embargo, un modelo que se describe en términos de *Ecore* es más expresivo y puede tener asociado un número de sintaxis concretas definidas para la generación de editores gráficos o de texto (R. C. Gronback, 2009).

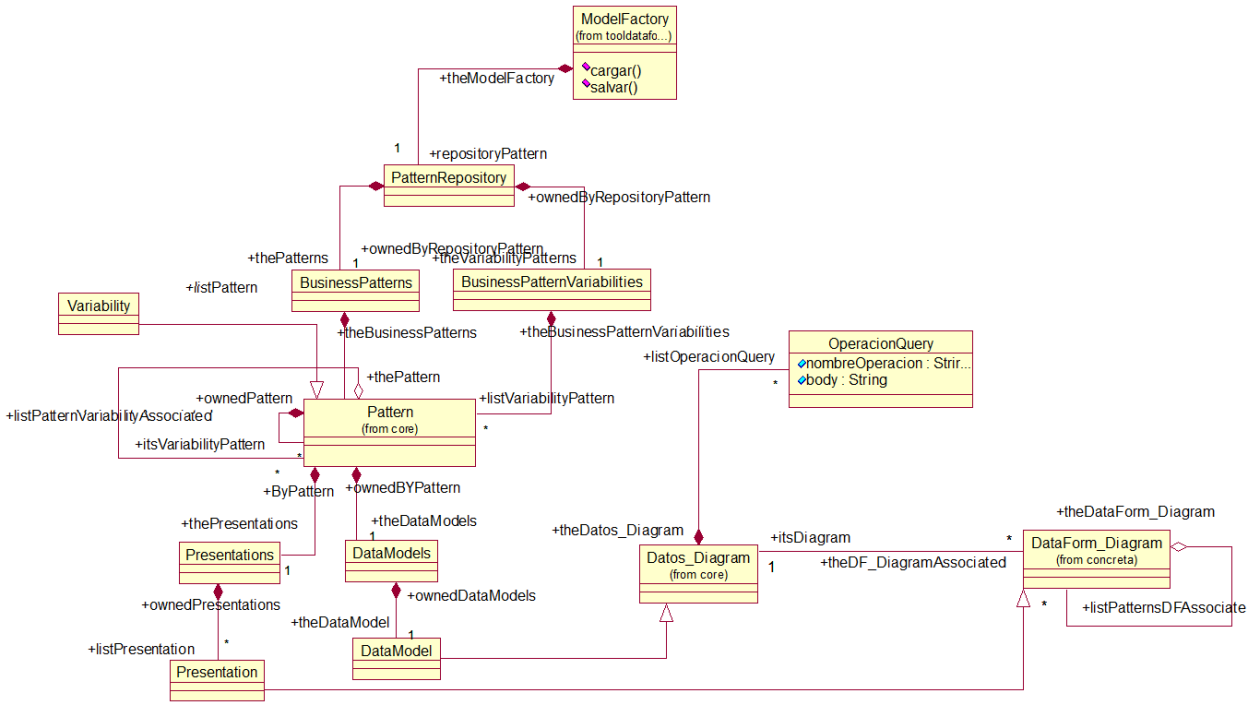

Figura 76 Metamodelo de la Herramienta

En la Figura 76, se observa los elementos de modelado definidos para la sintaxis abstracta del lenguaje DFL, este metamodelo está construido sobre el lenguaje EMOF. Mediante EMOF, es posible definir lenguajes y herramientas que permitan una manipulación externa de los modelos. Esto facilita la implementación de diversas propuestas para llevar a cabo una generación automática por medio de transformaciones. Así, pueden agregarse en un futuro nuevos algoritmos de procesamiento de modelos, sin que esto implique un cambio en los actuales editores.

7.4.1.2 **Creación del fichero generador de modelos**

El generador de modelos EMF es un modelo, que se obtiene a partir del metamodelo *Ecore*, que es necesario para la generación de las clases del dominio en lenguaje Java. El generador de modelos EMF, tiene una extensión de archivo *"*. genmodel"* y, es esencialmente un modelo decorado de su respectivo metamodelo *Ecore*. Este generador de modelos utiliza una serie de plantillas predefinidas que son especificadas en el lenguaje de emisión de plantillas Java (JET). La tecnología JET es utilizada para la transformación de modelos a texto a partir de modelos ECORE.

7.4.1.3 **Especificación de la sintaxis concreta**

La especificación de la sintaxis concreta es uno de los pasos preliminares en la especificación de cualquier lenguaje, debido a que en ella se define sus particularidades y el *mapping* con respecto de la sintaxis abstracta. Está especificación tiene como punto de inicio el análisis visual de cómo se presentarán al usuario desarrollador los elementos de modelado del lenguaje. Este modelo define la estructura de componentes gráficos usados para representar los conceptos. (Orozco, 2010).

Como se menciona anteriormente, el lenguaje usado para soportar el método propuesto es el Lenguaje DataForm, por lo tanto, en esta herramienta la sintaxis concreta es basada en la definida en toolDataForm. Para aclarar el concepto de sintaxis concreta se presenta como ejemplo se presenta en la en la Figura 77.a se ilustra el diseño gráfico del elemento de modelado interfaz de toolDataForm, el cual posteriormente, debe ser especificado a partir de los elementos de modelado visual que dispone GEF para la implementación en GMF (Figura 77.b). luego para cada componente de la notación gráfica del lenguaje se debe considerar aspectos de *layout*, etiquetado, estructura y contención que tendrán la presentación final del elemento de modelado(Figura 77.c).

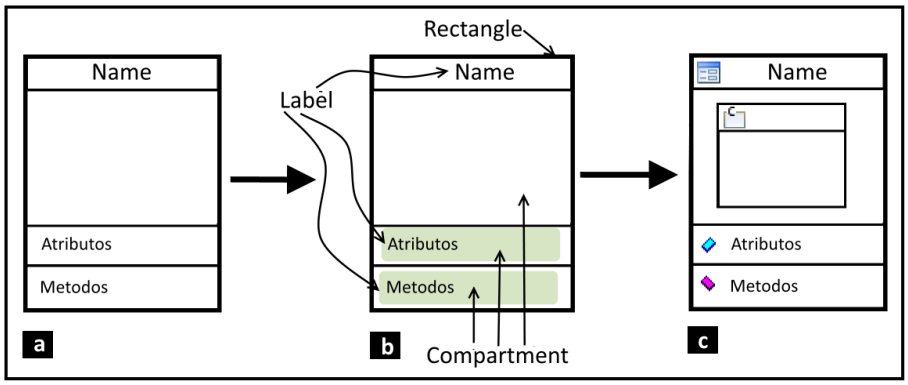

Figura 77 Representación visual del elemento Ventana

7.4.1.4 **Generación de la paleta de herramientas.**

Las herramientas de modelado, usualmente, incluyen paletas de herramientas y otras opciones para manipular los elementos de modelado. El propósito del modelo de paleta de herramientas es especificar estos elementos. La paleta de herramientas está compuesta por la barra de herramientas y diferentes menús que pueden ser definidos para un diagrama.

Mediante el uso de GMF es posible especificar todos y cada uno de los componentes de la barra de herramientas. Esta operación es bastante simple y consiste en crear instancias de cada uno de los botones de la barra de herramientas para que sea reconocida por el modelo de *mapping*.

7.4.1.5 **Definición del modelo de mapping**

El modelo de *mapping* es quizás el más importante de los modelos de GMF, esto debido a que, a través de él, los elementos de la definición del diagrama (nodos y enlaces) se asignan al modelo de dominio y se les asigna elementos de la paleta. El modelo de *mapping* representa la definición del *plugin* de diagrama y es utilizado para la creación del modelo de generación de diagrama de extensión *".gmfgen"*.(Orozco, 2010)

En la Figura 78 se presenta el modelo de *mapping "metamodelo.gmfmap"* de la herramienta. Mediante este modelo es posible especificar todas y cada una de las relaciones de contención de los componentes gráficos para obtener instancias de los objetos del dominio con el objetivo de hacer un *mapping* con su información. Adicionalmente, se relacionan los elementos gráficos entre sí y con la paleta.

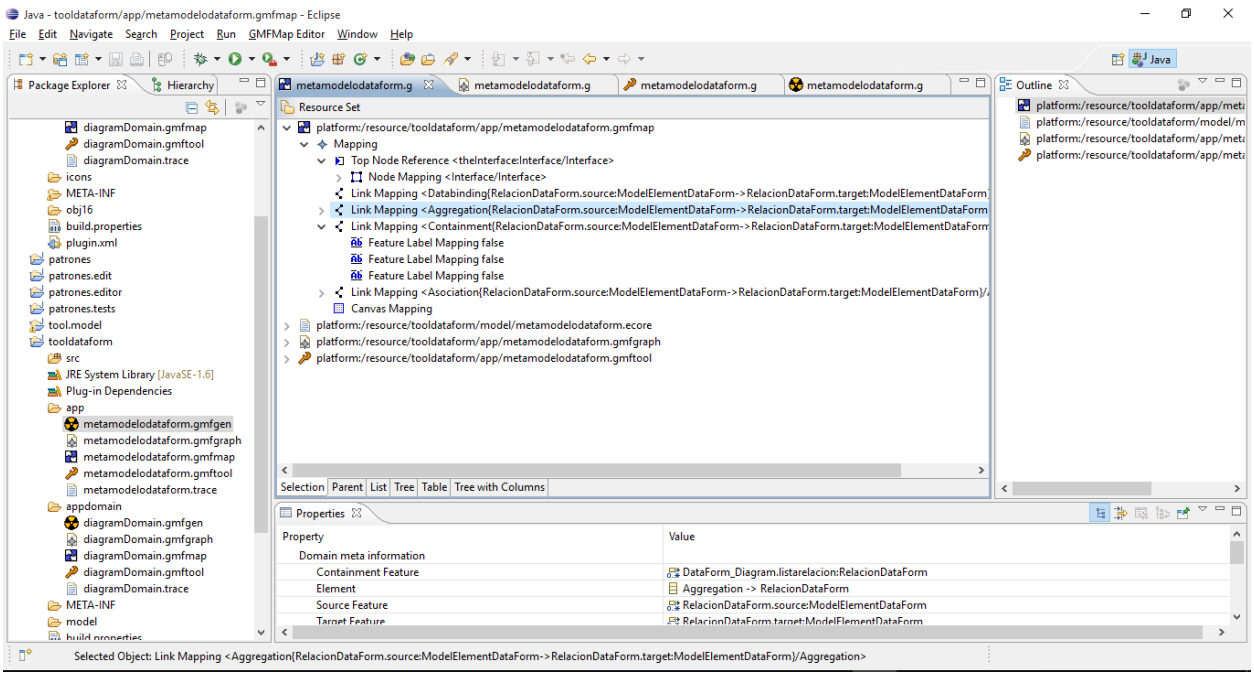

Figura 78 Definición del mapping para la herramienta

Antes de llevar a cabo un modelo de *mapping* es necesario conocer muy detalladamente la estructura de la sintaxis abstracta y concreta. Desafortunadamente, no siempre es posible obtener representaciones

gráficas que se adecuen a una estructura deseada de modelo de dominio. Esto se debe a que en GMF, cuando se arrastra un componente gráfico sobre el *canvas*, se está obligado a crear una instancia de dicho elemento de modelado dentro de la clase de dominio que está representando el *canvas* en ese momento determinado.

7.4.1.6 **Implementación de las Transformaciones**

Una transformación como lo define la real academia española es transmuta algo en otra cosa, en EMF desarrollo basado por modelos, un modelo inicial puede derivar en otro modelo final, una transformación es la encargada de implementar dicha acción.

Aunque existen muchos lenguajes para desarrollar transformaciones de modelos tal como ATLAS Transformation Language (ATL), Query Views Tranformation(QVT), para llevar a cabo todas las transformaciones que tienen lugar en el proceso del método de desarrollo de la interfaz de usuario de negocio se decidió usar java porque aporta más flexibilidad para el caso de nuestra propuesta.

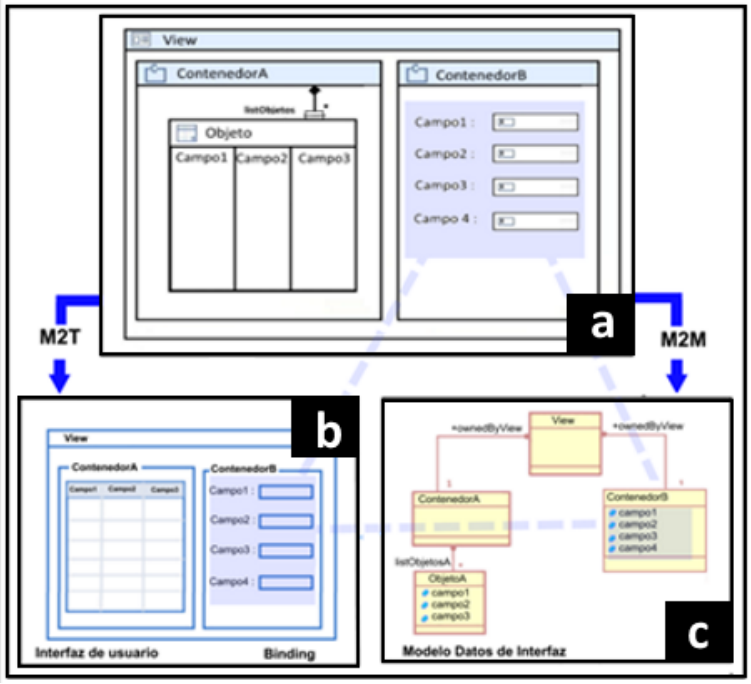

Figura 79 ejemplo aplicación de Transformaciones

Un ejemplo de la aplicación de las transformaciones se puede observar en Figura 79 ejemplo aplicación de Transformaciones, en donde a partir de un modelo DataForm se deriva la interfaz de usuario (Figura 79.b) utilizando la transformación M2T (Model to Text) , asimismo se genera el modelo de datos (Figura 79.c) mediante una transformación M2M (Model To Model).

7.4.2 Presentación de la Herramienta

En esta sección se presenta la descripción general de la herramienta desarrollada y se explican los elementos que la conforman. En la Figura 80 se muestra una vista general de la interfaz gráfica de la herramienta en donde se pueden observar los elementos que la componen

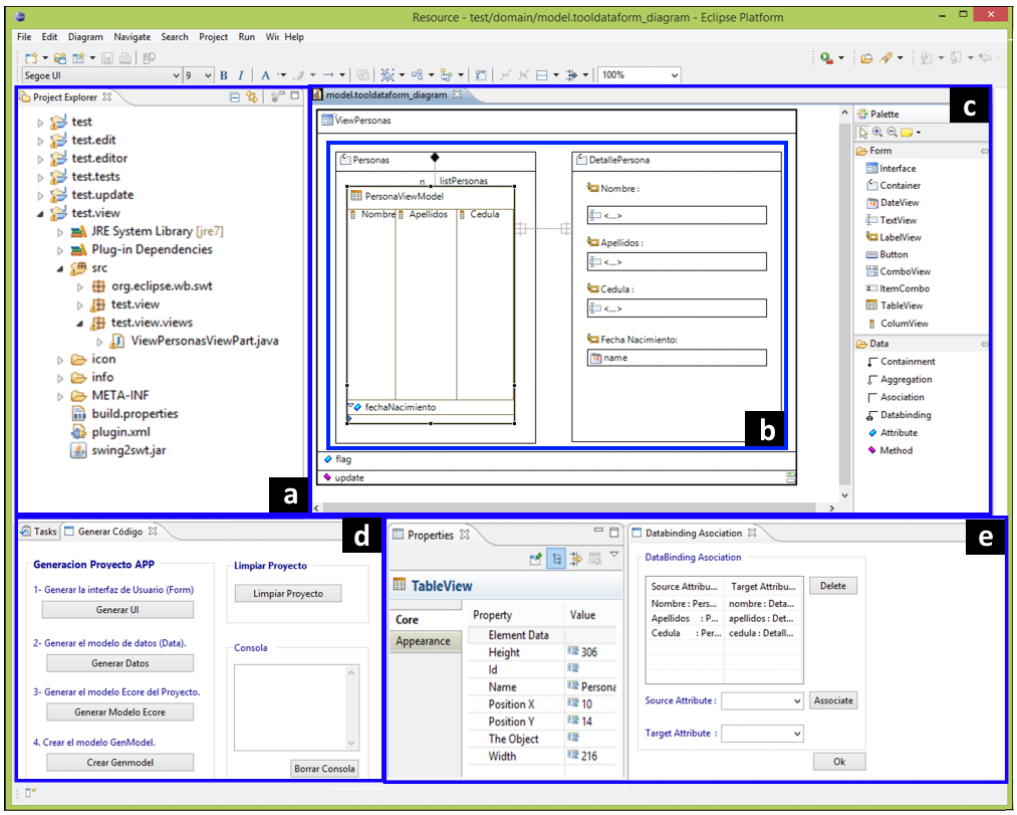

Figura 80 Vista General de la herramienta

A continuación, se explica brevemente cada componte:

7.4.2.1 Explorador de Proyecto

Como se puede observar en la Figura 80.a, este componente ilustra en forma de arbol los proyectos que son generados atomaticamente por la herramienta y se listan los archivos generados luego de realizar los pasos de generacion.

7.4.2.2 Editor Gráfico de Modelos

Este componente (Figura 80.b) permite la edición y construcción manual de los modelos DataForm haciendo uso de la notación definida en el lenguaje DFL, es un canvas donde el desarrollador haciendo

uso de la paleta de elementos (Figura 80.c) construye un modelo, arrastrando el elemento seleccionado de la paleta al canvas. Cada elemento que conforma el DataForm tiene un conjunto de propiedades que puede ser editado en la vista de propiedades (Figura 80.d).

7.4.2.3 **Paleta de Elementos**

La paleta de Elementos (Figura 80.c) contiene los distintos elementos que se pueden utilizar para la creación o edición de un modelo DataForm de acuerdo a la notación definida en el Lenguaje DataForm, como se puede observar está dividido en dos partes, la primera (parte superior de la paleta de elementos) agrupa los elementos que determinan aspectos de la forma y el segundo (parte inferior) los elementos relacionado a los datos.

7.4.2.4 **Vista de Selección de Patrones**

Este componente Figura 81, representa el árbol de navegación de patrones, permite seleccionar los patrones de negocios y determinar cuál variabilidad de datos y presentación se desea utilizar, para generar a partir de estos el modelo DataForm que represente tanto los datos del patrón como la forma de la presentación.

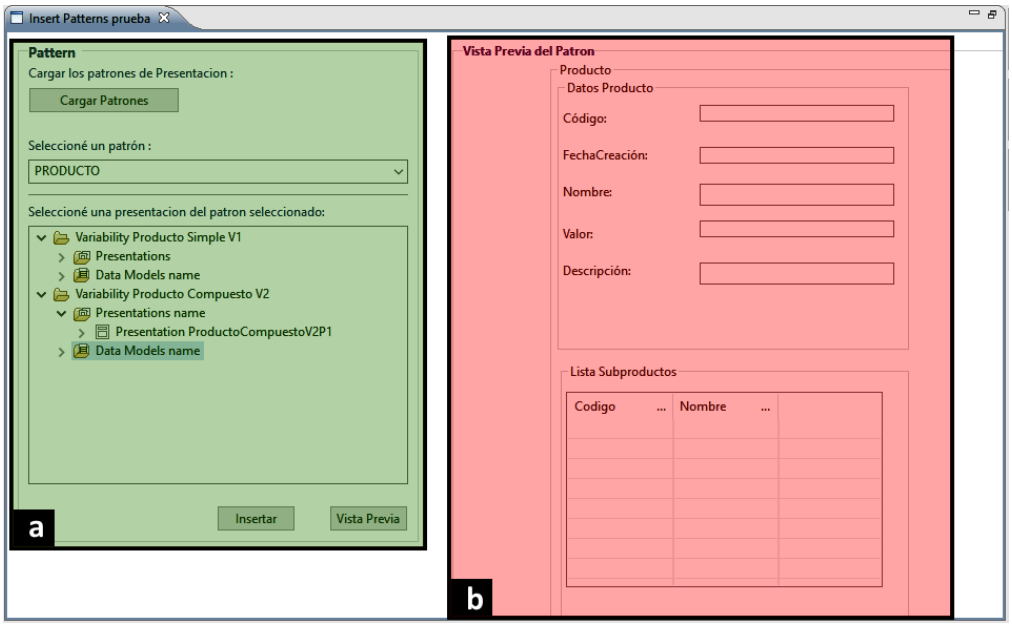

Figura 81 Vista de selección de patrones

Esta vista se pueden observar dos partes, la primera expuesta en la Figura 81.a) permite la selección del patrón y variabilidad a usar, se puede observar que por cada patrón se muestra un árbol de navegación, en donde se listan las distintas variabilidades del patrón y sus presentaciones asociadas permitiendo

generar una vista previa, como se puede ver en Figura 81.b) de cada uno de los patrones. Por último, este componente permite insertar el patrón seleccionado en el editor gráfico.

7.4.2.5 **Vista de Generación**

Este componente de la interfaz de la herramienta (Figura 80.d) cuenta con una serie de pasos que permiten generar el código automáticamente del prototipo de la interfaz de usuario de negocio que se esté creando, la Generación del código inicia desde la Generación de la interfaz de usuario hasta la generación del código java generado con EMF. En la Figura 82 se muestran los pasos necesarios para crear el código que permitirá ejecutar la aplicación o prototipo.

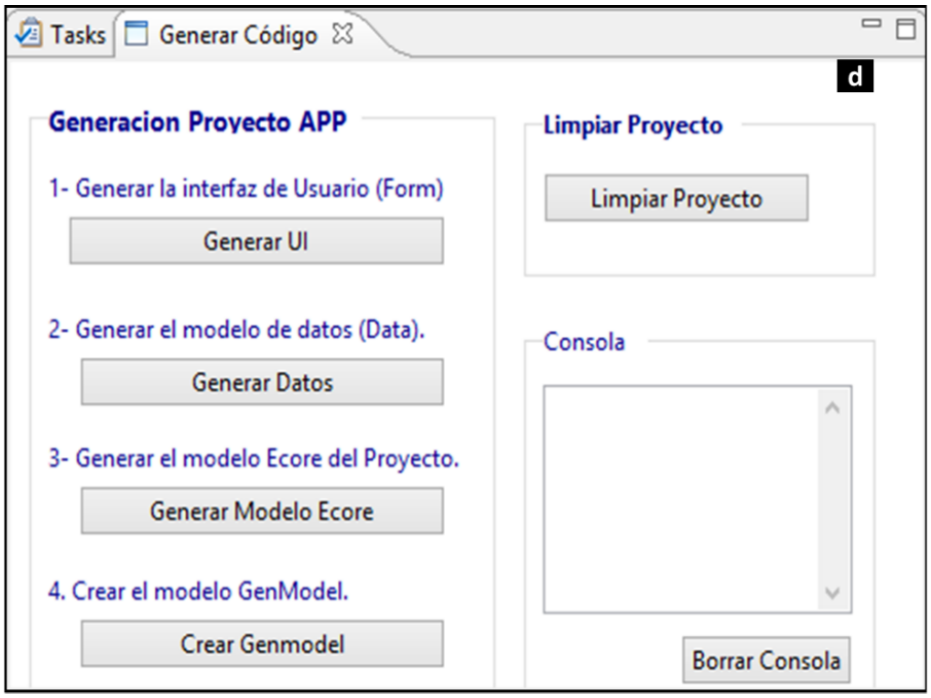

Figura 82 Vista de Generación de la Interfaz

Al ejecutar los pasos indicados la herramienta genera los archivos y proyectos necesarios para ejecutar la aplicación y poder visualizar e interactuar con la interfaz de negocio, los cuales se pueden visualizar en el explorador del Proyecto, asimismo genera las instancias de los patrones seleccionados (producciones de los datos preestablecidas creados por cada variabilidad de los patrones) para disponer de datos reales que permitan desplegar en el prototipo de interfaz de usuario de negocio información realista.

Una vez realizados todos los pasos indicados se debe a través del explorador (Figura 80.a) ejecutar la aplicación generada la cual produce una nueva instancia de Eclipse (Runtime), con la interfaz y el archivo

con las instancias del modelo de datos. Así, obtenemos el prototipo de interfaz generado automáticamente, en donde se pueden manipular los datos que serán mostrados en la interfaz de usuario, en la Figura 83.b se ilustra un ejemplo de una aplicación y prototipo de interfaz generada.

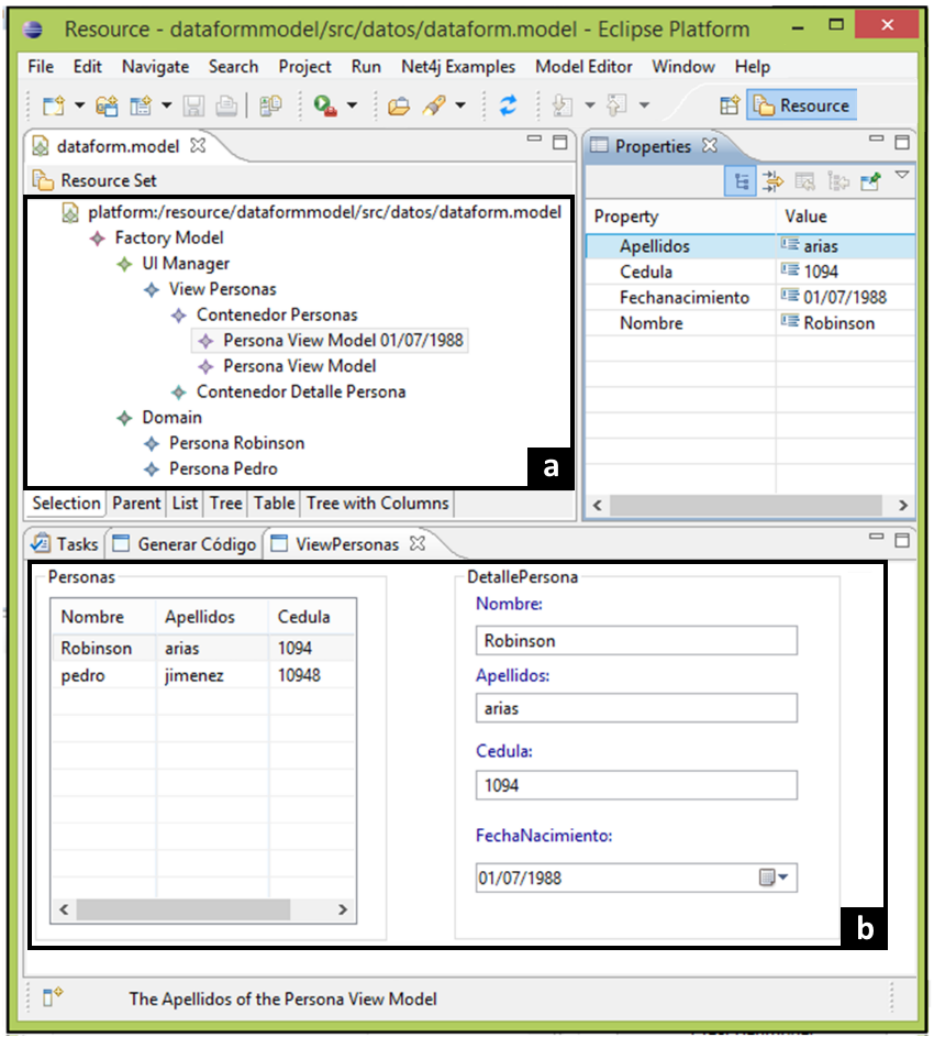

Figura 83 Aplicación con el prototipo de interfaz generada.

7.4.3 Catálogo de Patrones

Un catálogo es un conjunto de patrones relacionados entre sí y clasificados por algún criterio, su finalidad es clasificar los patrones y organizarlos permitiendo su búsqueda y utilización. Para la construcción de un prototipo de interfaz usando los patrones identificados, es necesario un catálogo que agrupe los DataForm y dominios definidos por cada variabilidad de acuerdo al patrón al que pertenecen, el objetivo es lograr un mecanismo que permita visualizar el conjunto de patrones, navegar a través de sus distintas

variabilidades y detallar la combinación de patrones soportada, facilitando aun desarrollador la utilización de ellos.

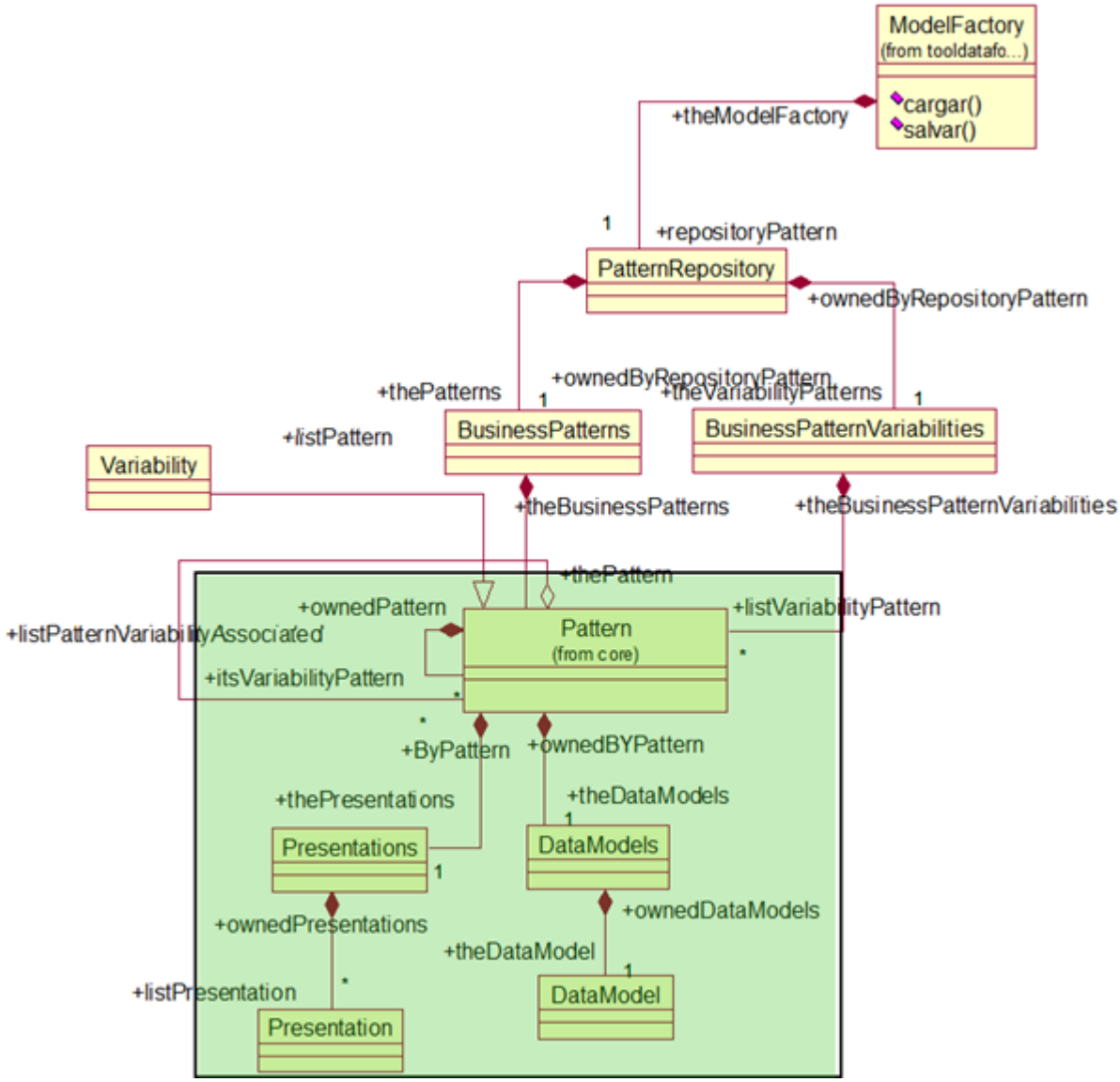

Figura 84 Metamodelo del Catálogo de Patrones

Como se describe en las tareas anteriores, un patrón de negocio tiene un conjunto de variabilidades en datos, cada una de ellas se conforman de un modelo de datos y un subconjunto de presentaciones representadas cada una con modelos DataForm. Para lograr la implementación del catálogo se define un metamodelo que permita su representación, el cual se puede visualizar en la Figura 84 sombreado en verde claro se puede observar que un patrón se compone de un Data Model y varias Presentation.

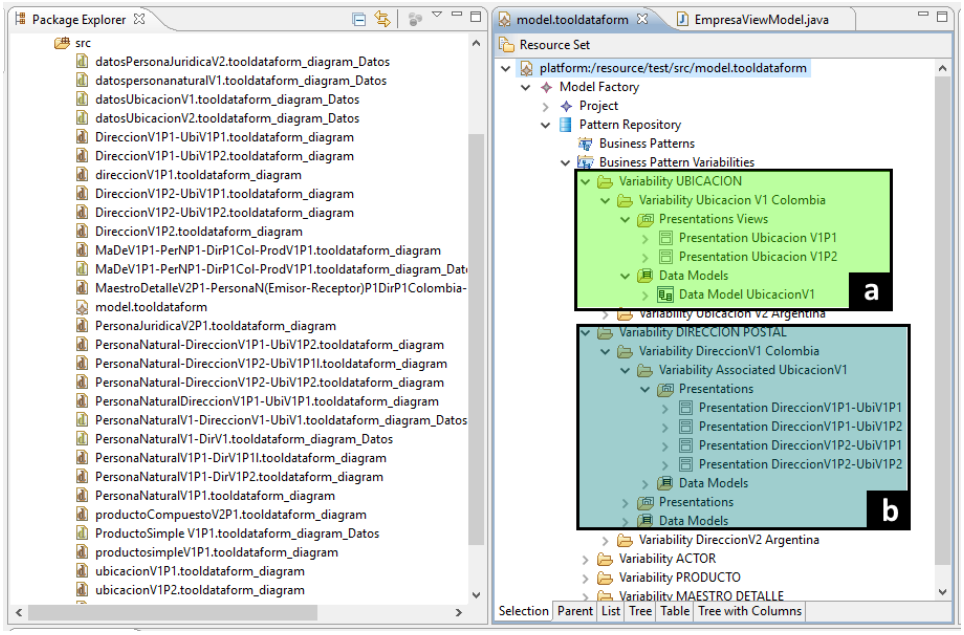

Figura 85 Catalogo de variabilidades implementado

Los patrones de negocios se agrupan en el concepto Business Patterns y a través del concepto Business Patterns Variability las Distintas variabilidades compuestas por un modelo de datos y los DataForm de cada presentación, asimismo los patrones con los cuales se puede asociar. En la Figura 85 se puede observar la implementación del catálogo, la cual corresponde a una producción del meta modelo expuesto en la Figura 84.

El catálogo de patrones permite que estos sean visualizados a través de la *vista de selección de patrones* o a través de la producción del meta modelo (árbol de variabilidad), en la Figura 85.a) se puede visualizar que el patrón UBICACIÓN GEOGRÁFICA tiene una variabilidad Colombia la cual incluye un modelo de datos y dos presentaciones. Asimismo, en la Figura 85.b) se observa que el patrón DIRECCIÓN se relaciona a través de la variabilidad 1 con la primera variabilidad del patrón UBICACIÓN GEOGRÁFICA. De esta forma podemos a partir de un patrón, visualizar sus diferentes variabilidades y navegar a través de los patrones que tiene relacionados.

7.5 RESUMEN DEL MÉTODO PROPUESTO

A continuación, para una mejor comprensión del método propuesto se presenta una tabla resumen donde se relacionan las actividades de cada disciplina, las herramientas de soporte para realizarlas y los productos generados.

	Context Analysis	Business Pattern Selection	Business Pattern Adaptation	BUI Preparation	BUI Validation
Activities	A1. Identificar el Contexto del problema	A2. Selección del patrón de negocio A3. Definir la Variabilidad del patrón	A4. Ajustar Business Domain Data A5. Ajustar DataForm	A6. Implementar Interacción A7. Generar Producciones	A7. Pruebas de Comprensibilidad
Tools	T1. Programa de Ofimática	T2. CIAT (Vista de seleccion de patrones)	T3. CIAT (Editor Gráfico de modelos)	T4. CIAT (Vista de Resource Set) T5. CIAT (Editor Código Fuente)	T5. CIAT (View Interfaz Generada)
Products	P1. Contexto del Dominio	P2. Data Form P3. Paper Form P4. Business Domain Data	P5. Data Form P6. Paper Form P7. Business Domain Data	P6. Productions P7. Domain Data	P8. Business User Interface

Tabla 4 Resumen Método Propuesto

A: Actividad

T: Herramienta

P: Producto

CAPÍTULO OCTAVO

8. VALIDACIÓN DE LA METODOLOGÍA PROPUESTA

En este capítulo se presenta el proceso de validación de la metodología propuesta. En primer lugar, se expone un caso de validación hipotético donde se aplica el método de desarrollo de la interfaz de usuario de negocio propuesto. En segundo lugar, se presenta el análisis de una prueba realizada con estudiantes de ingeniería de sistemas y computación de la Universidad del Quindío sobre la aplicación de los patrones definidos.

8.1 CASO DE VALIDACIÓN

Para validar la aplicación de esta tesis se expondrá un caso hipotético a través del cual se demostrará la factibilidad de uso del método propuesto, el Contexto del caso a usar se limitará al proceso de venta de una Empresa de comercializadora de productos en Colombia, limitándolo a la información de clientes, productos y facturas de venta.

El objetivo principal de caso de validación es soportar las tareas de registro de ventas, productos y de clientes de la empresa comercializadora. A continuación, se describe la utilización del proceso metodológico propuesto y patrones usados para generar los prototipos para una aplicación que permita realizar el proceso de venta.

8.1.1 ANÁLISIS DEL CONTEXTO

En el análisis del contexto se estudia el dominio de la aplicación, para identificar el contexto del problema, es decir, identificar al usuario, las tareas que debe realizar y los conceptos de negocio y datos que son manipulados para realizarlas. El resultado de este paso es el artefacto *contexto de dominio,* el cual consiste en la descripción general de los conceptos de negocio (Entidades) y los datos del contexto del problema.

Como ya se indicó, el contexto de este caso es una empresa colombiana de venta de productos, la cual tiene como tareas principales: *registrar clientes, registrar productos* y *registrar ventas.* Se requiere generar las propuestas de interfaces para soportar dichas tareas.

De acuerdo al contexto del problema se identifican los conceptos de negocio principales, comunes a este tipo de escenario:

I. **Clientes:** El cliente es el actor que va a realizar la compra, por lo tanto, se necesitan los datos para identificarlo y ubicarlo.

II. **Producto:** Según el contexto, la empresa se dedica a ventas de productos; el producto constituye un algo que ofrece a sus clientes, sea servicios o elementos físicos, para este caso de validación supondremos que son productos físicos y pueden estar compuesto de subproductos.

III. **Factura de venta:** Toda venta que se realice en una empresa requiere un soporte que demuestre la realización de la transacción entre la empresa y el cliente, detallando los productos involucrados en dicha transacción.

8.1.2 SELECCIÓN DEL PATRÓN DE NEGOCIO

En este paso, basados en la tarea de *Análisis del contexto,* se seleccionan el o los patrones que sirven para dar soporte a los conceptos de negocios identificados. Para cada uno de ellos se define la variabilidad que se ajuste al contexto del problema y de acuerdo a los datos que el usuario requiere para realizar la tarea se debe seleccionar la plantilla de presentación (Paper Form).

Una vez identificados los conceptos de negocios, se debe buscar los patrones de negocio aplicables al contexto. Esta tarea esta soportada por la herramienta a través de *la vista de selección de patrones*, la cual permite visualizar el catálogo de patrones, en la Figura 86, se puede observar sombreado en azul claro los patrones del catálogo.

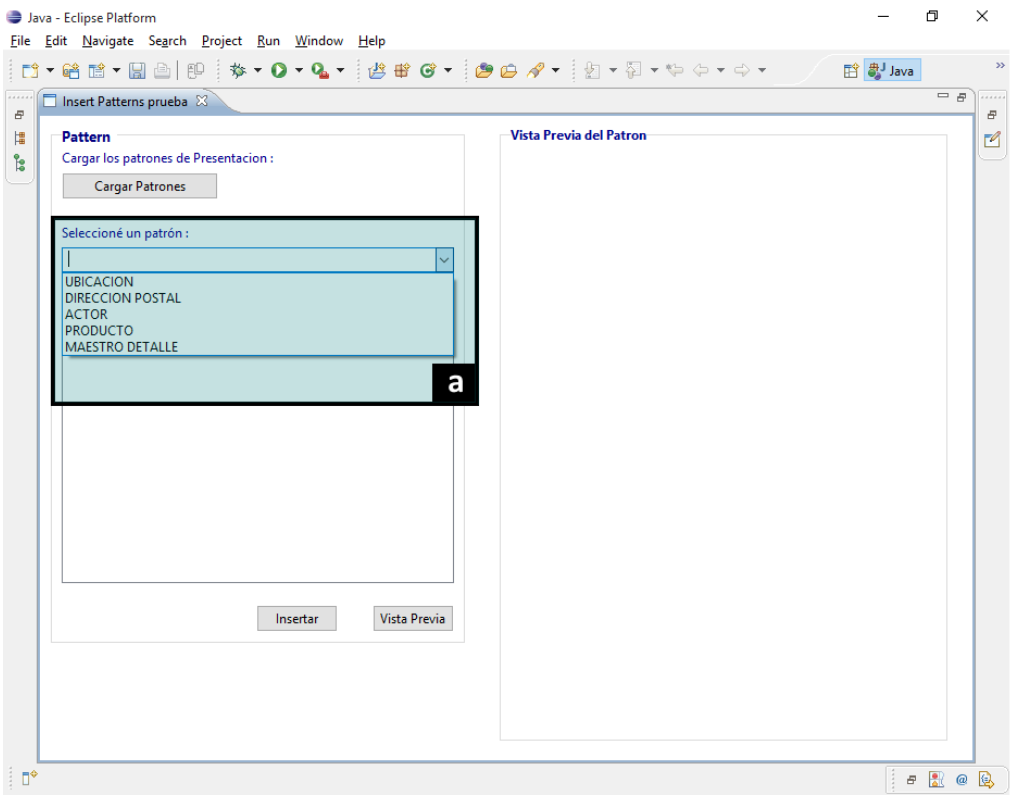

Figura 86 Visualización del catálogo de patrones en la herramienta

Como se mencionó, por cada concepto de negocio se debe seleccionar el patrón de negocio que permita su representación y luego la variabilidad que represente el contexto del problema, a continuación, se ilustrara esta tarea tomando de ejemplo el concepto *CLIENTE*, al final de esa sección se mostrara de manera resumida los artefactos generados para los demás conceptos de negocios.

8.1.2.1 **Seleccionar patrón**

De acuerdo al contexto, el cliente es el actor que realiza la compra, por lo tanto, se necesita identificarlo, en el dominio de una empresa de venta, lo común es que se incluya en los datos del cliente su dirección postal. De acuerdo a lo anterior se selecciona del catálogo el patrón ACTOR, el cual, como se muestra en la Figura 87 tiene en el árbol de variabilidades dos diferentes.

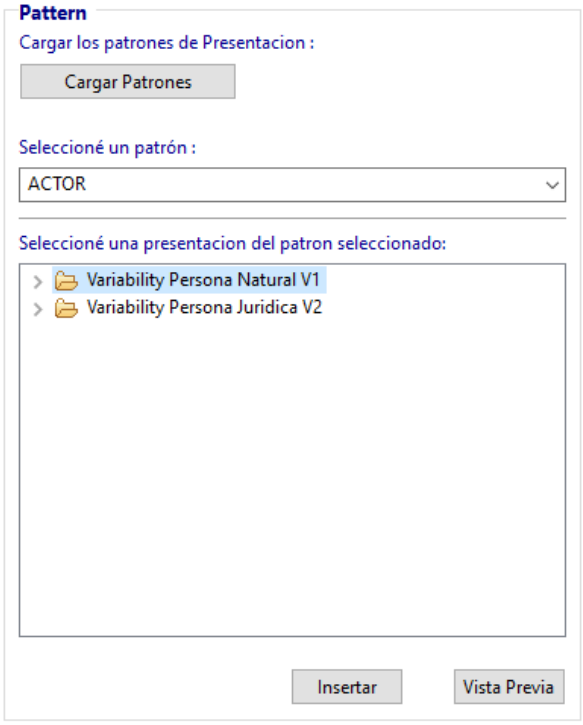

Figura 87 Árbol de variabilidad del Patrón Actor

8.1.2.2 **Seleccionar Variabilidad**

Una vez definido el patrón se debe seleccionar de acuerdo al contexto la variabilidad, cada una de ellas permite identificar los patrones con los cuales se puede relacionar, en la Figura 87 se puede observar que el patrón ACTOR posee dos variabilidades, la primera para personas naturales y la segunda para personas jurídicas (empresas).

El árbol de variabilidad permite identificar y seleccionar los patrones relacionados, por ejemplo, en la Figura 88.a) se puede observar que la variabilidad persona natural tiene relación con la variabilidad dirección postal, la cual permite definir la dirección postal de la persona y su ubicación dentro del territorio colombiano, y esta a su vez con ubicación geográfica, a través de la cual se maneja la estructura de país – departamento – municipio, asimismo se puede observar que existen 4 plantillas de presentación diferentes, para este caso de validación se selecciona la primera (Figura 88.a) recuadro negro), la vista previa de esta presentación se observa en la Figura 88.b)

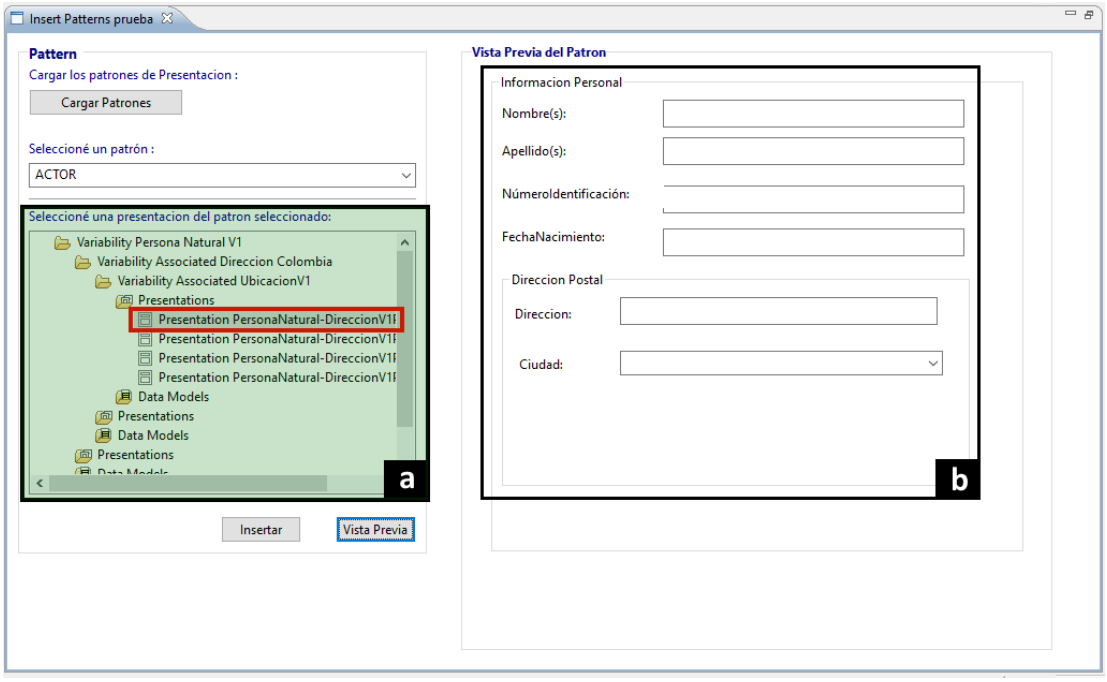

Figura 88 Árbol variabilidad desplegado del patrón Actor y Paper Form Asociado

En esta tarea, se puede validar con el usuario cada una de las presentaciones disponibles, para determinar cuál es la que él considera se ajusta adecuadamente a su modelo mental y continuar con esta el proceso de generación de la BUI. Una vez seleccionada el prototipo de interfaz, se define a partir de esta el Business Domain Data, el Paper Form y el DataForm, la herramienta propuesta genera a partir del Paper Form asociado a la variabilidad seleccionada estos componentes de manera automática.

8.1.2.3 **Definición del Bussines Domain Data**

Este componente está conformado por las entidades de negocio, que representan los datos que soportan las tareas del negocio, a partir de la variabilidad patrón o patrones seleccionados se constituye este elemento conformado adicionando al modelo de dominio de la aplicación las clases, atributos y relaciones especificadas en la variabilidad definida.

Como se indicó en la sección anterior, para representar el concepto de negocio cliente se selecciona la variabilidad de persona natural con dirección postal y ubicación Geográfica del patrón Actor, por lo tanto, se adiciona al domain las clases de las variabilidades involucradas (ver Figura 89). Este modelo está conformado por la clase Persona natural (Figura 89.a) que corresponde al patrón Actor, por la clase

Dirección Postal generada del patrón Dirección (Figura 89.b) y las clases País, Departamento y Municipio (Figura 89.c) obtenidas del patrón Ubicación Geográfica.

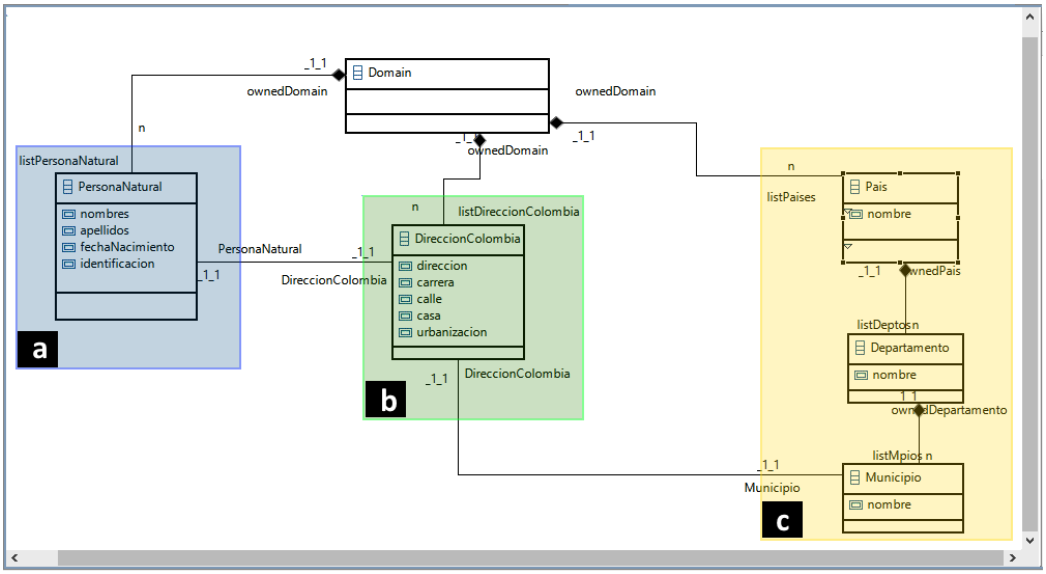

Figura 89 Business domain data de la variabilidad Persona Natural con Dirección postal

8.1.2.4 **DataForm**

El DataForm es un modelo de información de contexto que representa los datos persistentes en la interfaz que serán mostrados al usuario, representa los fragmentos del Business domain Data que serán de interés para el usuario y serán mostrados en la interfaz, es muy útil para la especificación de la información que será visualizada y manipulada en la interfaz de usuario independiente de la manera en cómo es utilizada en el dominio.

El DataForm está asociado a la variabilidad del patrón seleccionada, cada una de ellas tiene preestablecido este componente, su generación a través de la herramienta es automática, en la Figura 90.a se puede observar la vista previa de la plantilla de presentación del patrón previamente seleccionado y en la Figura 90.b se presenta el DataForm.

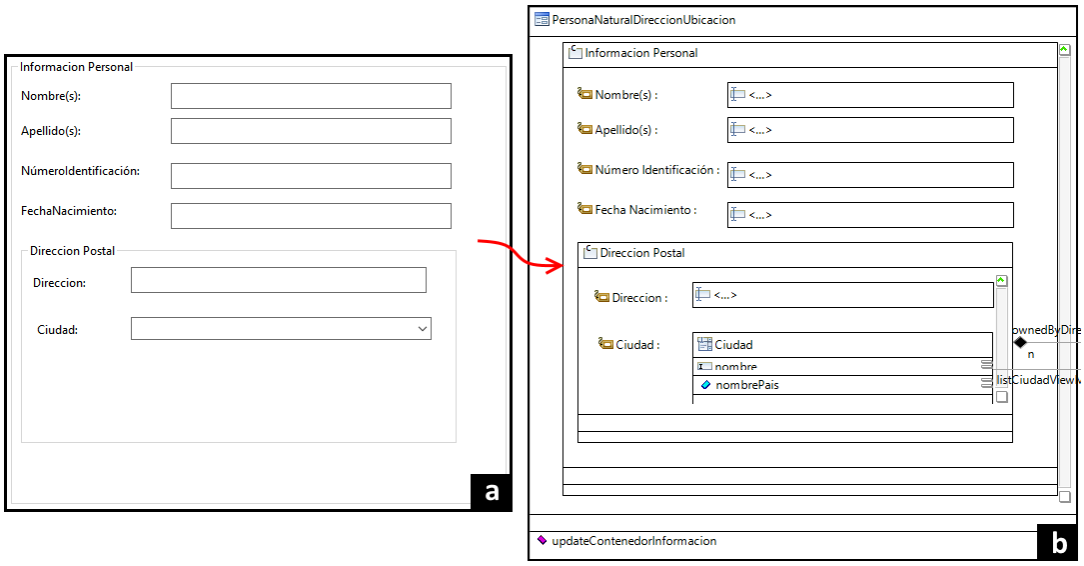

Figura 90 DataForm de la variabilidad Persona natural con dirección postal del patrón Actor

8.1.2.5 **Paper Form**

El Paper Form describe la estructura de la presentación, ligada a la semántica de los datos, muestra un diseño visual de manera realista de los Business Domain data que soportan la tarea, la renderización de este componente constituye la interfaz de usuario de negocio. Como se explicó este componente corresponde a cada una de las plantilla de presentación asociada a las variabilidades de los patrones, por lo tanto, tal como se evidencia en la Figura 88.b, a través del árbol de variabilidad se puede ver una vista previa del Paper Form de la opción seleccionada (Figura 90.a).

Como se indicó al inicio de esta sección, por cada concepto de negocio se deben realizar el conjunto de pasos descritos, en este caso de validación solo se muestra en detalle el proceso a partir del concepto Cliente, los artefactos usados y resultantes de esta etapa para el concepto Producto, para el cual utilizamos el patrón Producto Compuesto y Factura venta el cual se soporta con el patrón orden –detalle. Los cuales se muestran en la Figura 91 y Figura 92 respectivamente, en cada una de ellas se observan el árbol de variabilidad (identificado con a)), el Paper Form (identificado con b)) y el DataForm (identificado con c)).

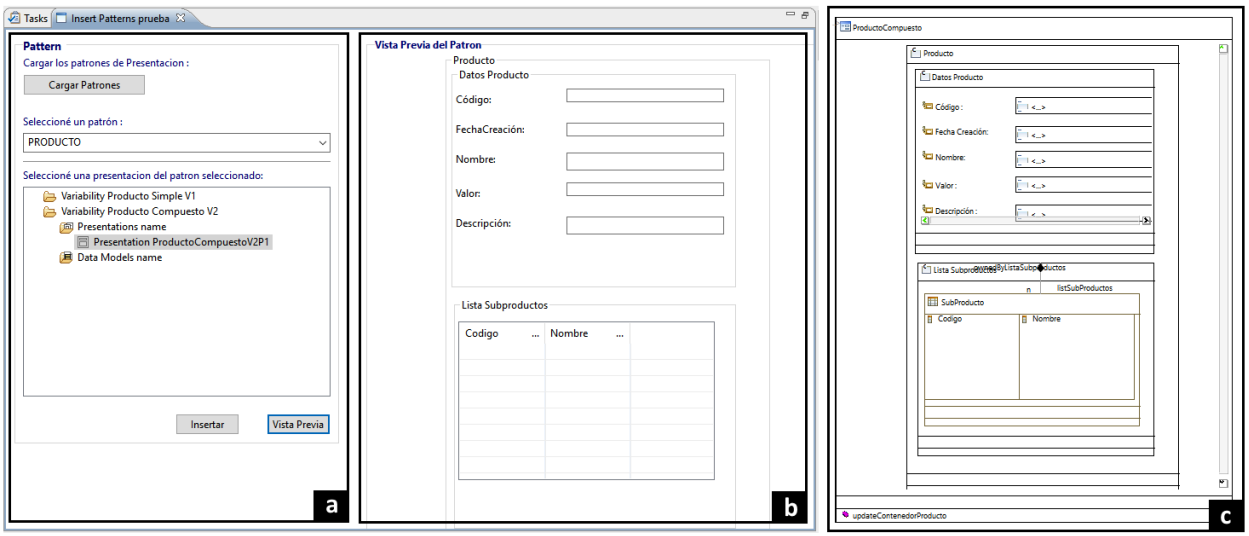

Figura 91 Artefactos Generados para el concepto Producto

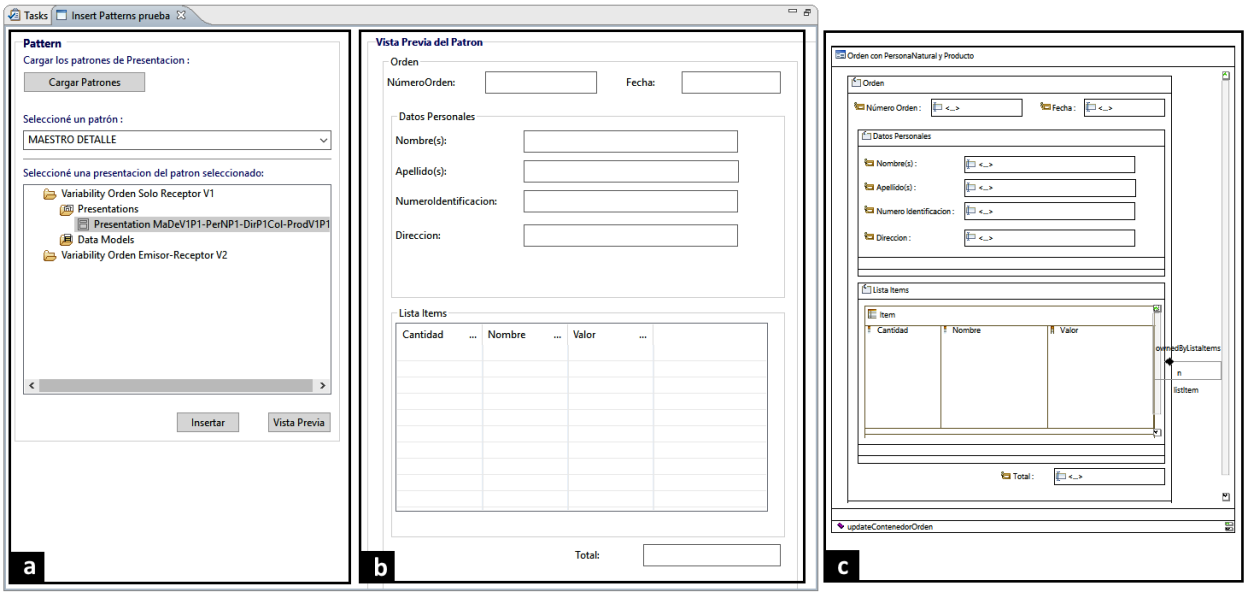

Figura 92 Artefactos Generados para el concepto Factura de Venta

Al finalizar esta etapa se tiene el domain de la aplicación creado, el cual se compone de las clases y relaciones que conforman los patrones seleccionados, es decir, los Business Domain Data asociados a cada variabilidad escogida se fusionan para soportar todos los conceptos identificados, en la Figura 93 se observa el domain data de la aplicación, en la cual Figura 93.a corresponde al patrón producto compuesto, la Figura 93.b a la variabilidad orden detalle, la Figura 93.c representa la persona natural con dirección y la Figura 93.d las clases del patrón ubicación geográfica.

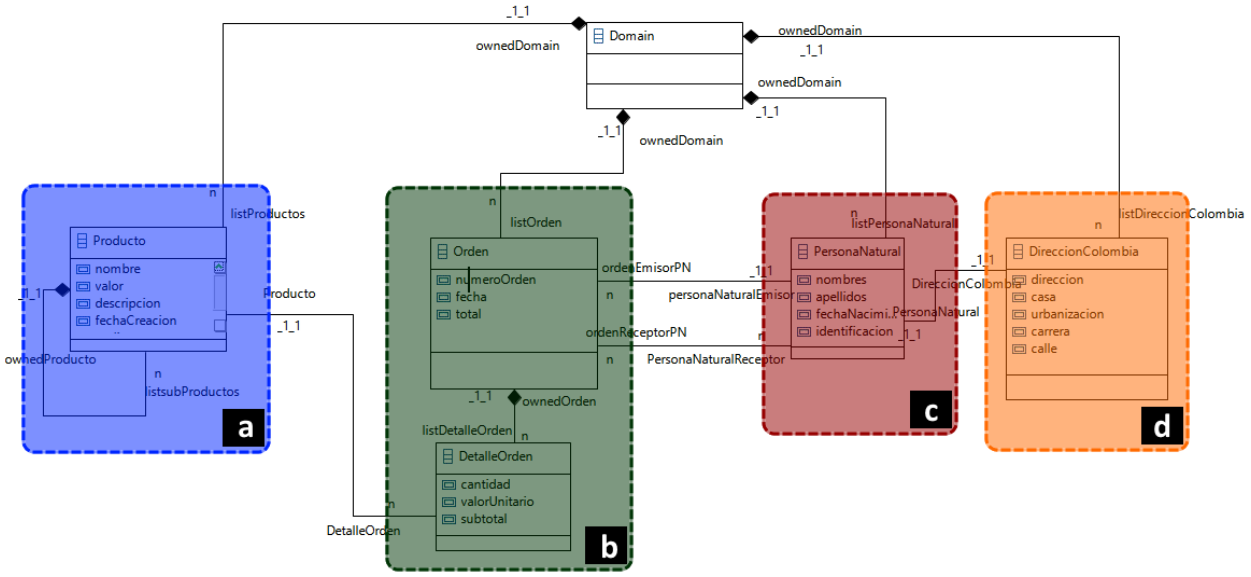

Figura 93 Domain model de la aplicación

8.1.3 ADAPTACIÓN DEL PATRÓN DE NEGOCIO

En esta tarea se debe validar que el patrón soporte completamente las tareas y datos necesarios para realizarla, y de ser necesario agregar o quitar datos que no estén representados, modificando el *DataForm* y el *Business Domain Data*. Esta modificación debe ser armónica, ya que se debe garantizar que los datos de la forma estén mapeados en el dominio.

La adaptación del patrón se realiza desde la perspectiva del usuario, por esto se realiza a partir de la vista previa de la plantilla de presentación asociada al patrón de negocio seleccionado, un ejemplo de esta puede verse en la Figura 92.b. En este caso de validación no se muestra la realización de este paso, se toma la consideración de que los patrones de negocio seleccionados (presentaciones y datos) representan completamente el modelo mental del usuario.

8.1.4 PREPARACIÓN DE LA BUI

Los prototipos de interfaz de negocios se generan realizando un renderizado de las presentaciones de los patrones de negocios las cuales tienen predefinidas un modelo de interacción básico, por lo tanto, esta de ser necesario pueden ser adaptada según el modelo mental del usuario y se deben preparar para ser sometidos a una validación realista con el cliente.

En este paso se refinan los prototipos de interfaz generados, implementando la interacción del prototipo de la BUI, esta implementación se refiere a realizar los aspectos de lógica de programación que permitan exponer funcionalidades básicas en los prototipos, asimismo, se debe generar o cargar (en caso de existir)

una *producción* de reales de los conceptos del negocio involucrados; esto se realiza con el fin de poder realizar una validación temprana con el cliente.

8.1.4.1 **Generación del banco de datos (Producciones)**

Este paso tiene por finalidad la creación o carga del conjunto de datos (producciones) al domain de la aplicación, permitiendo mostrar de manera realista registros en los formularios y exponer asimismo su funcionalidad (ver Figura 94.a). Al referirnos a producciones hacemos alusión a la creación de instancias (objetos) de las clases que conforman el Data Domain, de tal manera, que en el formulario se puedan precargar y visualizar dichos registros, facilitando la realización de la siguiente actividad, la validación de la Interfaz de usuario de negocio.

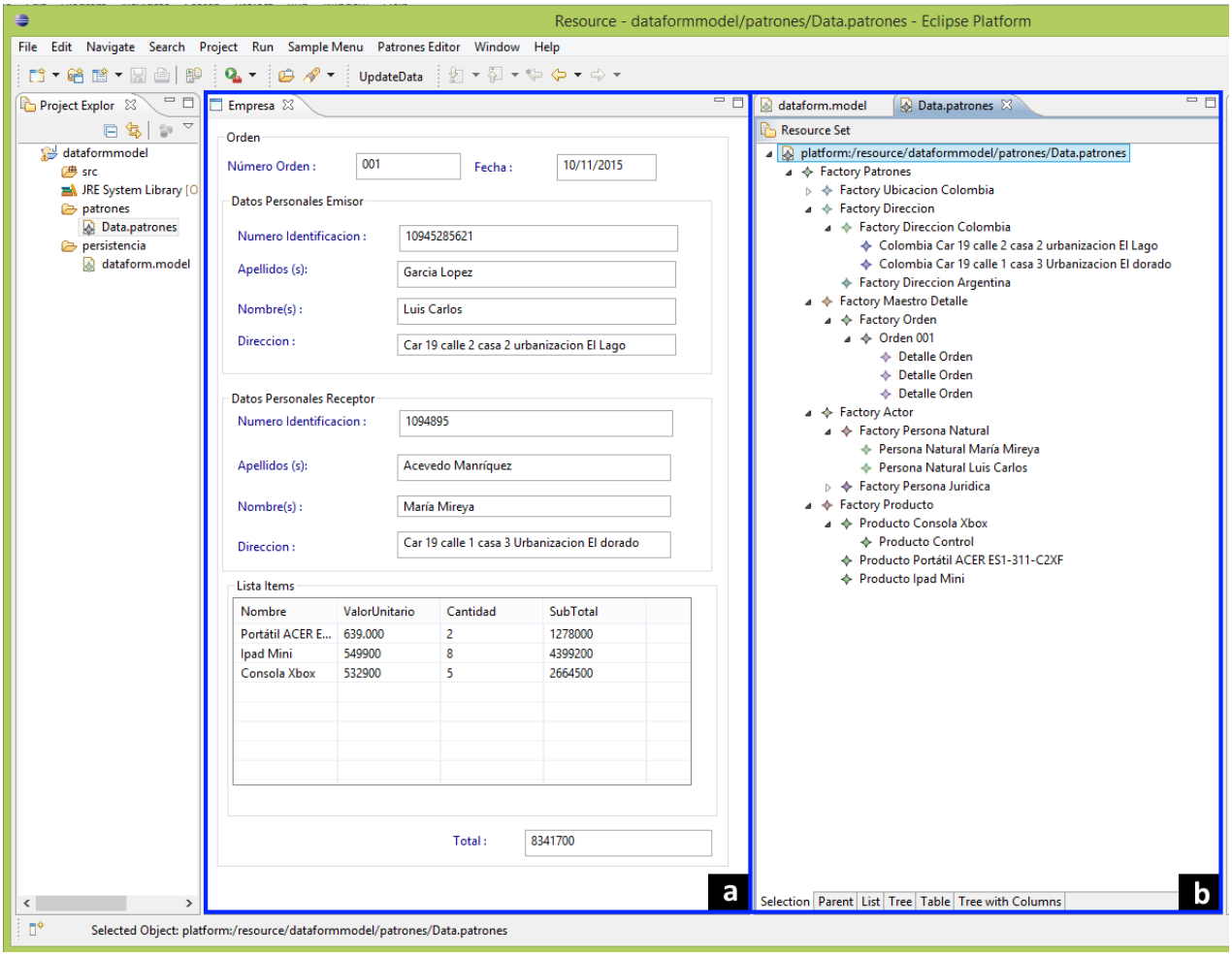

Figura 94 Producciones del Domain

Cada patrón tiene asociado un banco de datos, por lo tanto, las producciones se cargan al domain de manera automática al usar la herramienta propuesta, no obstante, pueden ser creada de manera manual a través haciendo uso de la facilidad del eclipse (ver Figura 94.b) que nos permite visualizar el modelo

8.1.4.2 **Implementación de la Interacción**

La implementación de la interacción consiste en adicionar a los prototipos de la interfaz las funcionalidades necesarias para plasmar la forma como el usuario quiere interactuar con el sistema, esto se realiza a través de lógica de programación. En este caso de validación vamos a suponer que el usuario desea la siguiente funcionalidad: "En el formulario de registrar la factura, al digitar la cedula del cliente se completen sus datos de manera automática".

Figura 95 Implementación de la Interacción

Para la implementación de la interacción requerida por el usuario, se debe adicionar al prototipo de interfaz Gráfica el evento sobre el componente en donde se adiciona las instrucciones de programación necesarias para satisfacer el requerimiento. En la Figura 95.a se presenta la codificación del evento, el cual invoca el método que *updateCliente* (ver Figura 95.b) donde se codificó la lógica de programación necesarias para pedirle al dominio los datos del cliente y presentarlos en la interfaz gráfica.

A continuación, con los prototipos de la BUI preparados, se procede a validar la interfaz gráfica con el usuario, exponiendo los datos y las funcionalidades implementadas.

8.1.5 VALIDACIÓN DE LA BUI

En esta etapa se evalúan las BUI generadas. Dicha prueba se podrá hacer de muchas maneras; por ejemplo, aplicando pruebas de comprensibilidad (una prueba de usabilidad simplificada), se verifican los accesos e interacción a los elementos de las pantallas, los datos que se requieren para llevar a cabo cada tarea. En la Figura 96 se presenta el prototipo final de la Interfaz de usuario de negocio generada para el concepto Factura.

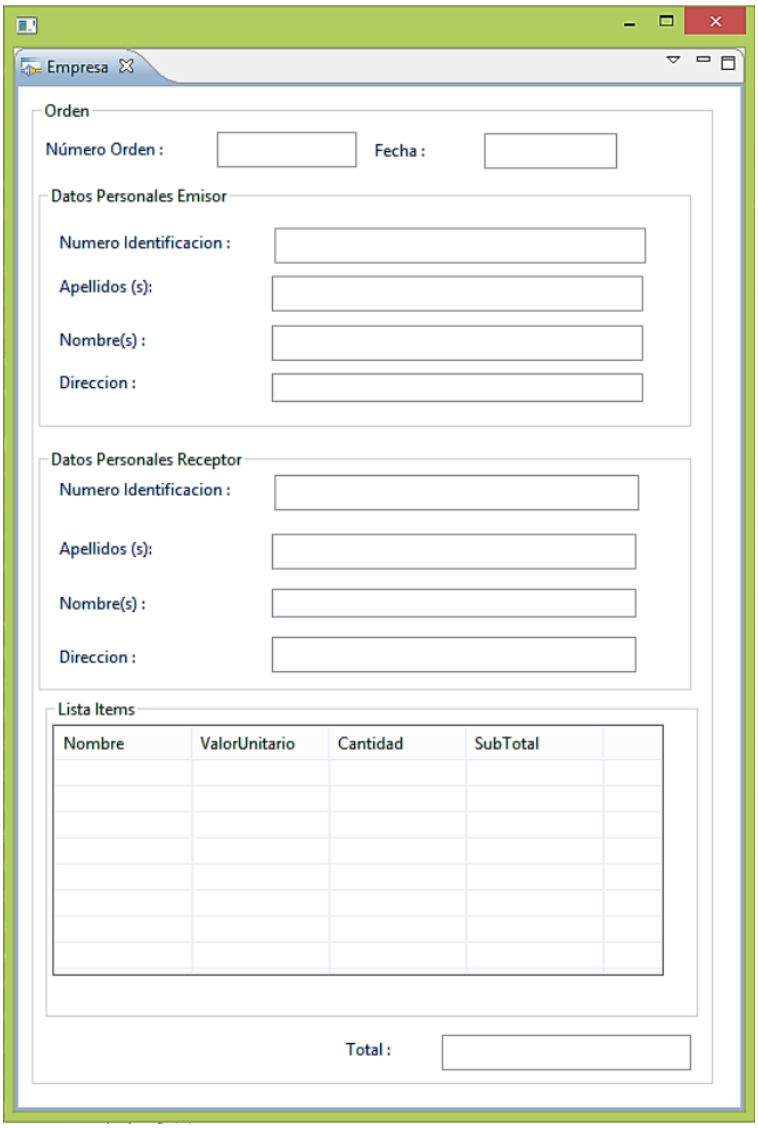

Figura 96 Interfaz de Usuario Generada

8.2 VALIDACIÓN CONCEPTUAL DEL ENFOQUE CON PATRONES

Para validar conceptualmente el enfoque de desarrollo de la interfaz de usuario a partir de patrones, se realizó una prueba con la participación de 4 estudiantes de último semestre del Programa de Ingeniería de Sistemas y Computación de la Universidad del Quindío los cuales han cursado los espacios académicos de ingeniería de software, en donde aplican los conceptos del patrón MVVM en el desarrollo de la interfaz de usuario y del lenguaje Data Form.

La prueba consistió en que, a partir de un Paper Form de una Factura (ver Figura 97) que representa el modelo mental de un usuario, los participantes deben identificar y definir la view, el viewmodel y domain model.

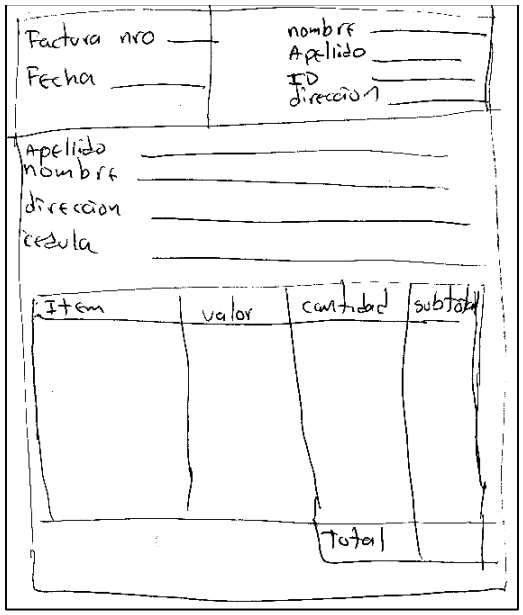

Figura 97 PaperForm en representación de modelo mental del usuario

8.2.1 Objetivo de la Prueba

El objetivo de la prueba es validar la aplicabilidad de los patrones como mecanismo para mejorar la calidad y tiempo en el desarrollo de prototipos de interfaz.

8.2.2 Criterios de Evaluación

Los criterios de evaluación utilizados para como base para la comparación e interpretación de los resultados fueron:

Efectividad: Tiempo necesario para la realización de las tareas, con este criterio se pretende evaluar el tiempo necesario por los participantes en efectuar la prueba.

Concordancia en los modelos: Con este criterio se pretende evaluar la concordancia de los modelos realizados por los participantes que conforman un grupo de prueba, lo cual permite analizar la subjetividad incluida por cada sujeto al realizar una la tarea.

8.2.3 Aplicación de la Prueba

Para la aplicación de la prueba los participantes fueron divididos de manera equitativa en dos grupos de prueba. Un grupo debía realizar el proceso mencionado haciendo uso del catálogo de patrones y el segundo grupo sin este recurso, al final de la prueba cada estudiante debe entregar el modelo de la view, del view model y del model (ver Figura 98).

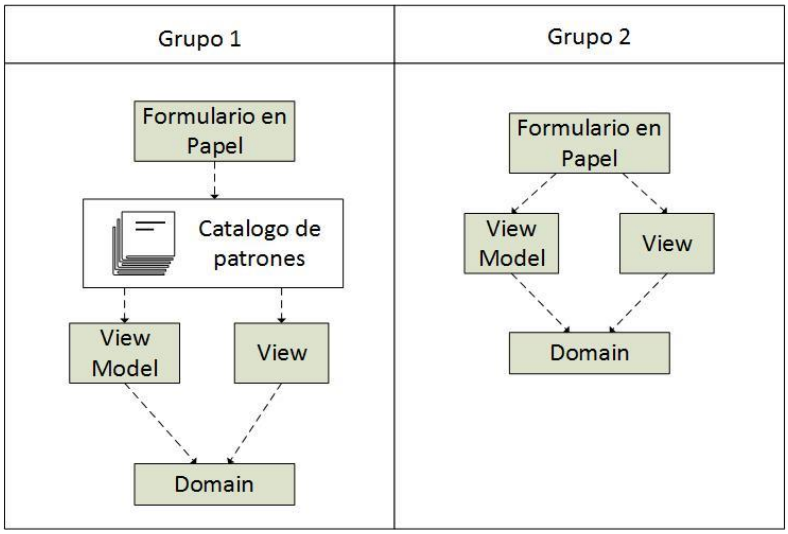

Figura 98 Proceso de la prueba aplicada

A continuación, se presentan el análisis de los resultados de esta prueba, los artefactos generados por cada estudiante se presentan en el ANEXO I ARTEFACTOS GENERADOS EN LA PRUEBA.

8.2.4 Análisis de Resultados

El objetivo de la prueba es comparar la calidad y homogeneidad de los artefactos generados al enfoque de patrones en el desarrollo de la interfaz de usuario de negocio. En consecuencia, se realiza un análisis en el cual se realiza un comparativo entre los artefactos generados en cada grupo de prueba. El resumen de las observaciones identificadas en el análisis de los artefactos resultantes de la aplicación de la prueba se describe en la Tabla 5.

Artefacto	Grupo 1	Grupo2
View	- Conceptos principales bien identificados - Ubicación homogénea de los contendores - Atributos ubicados en los contenedores adecuados	- Conceptos principales bien identificados - Diferencias en la ubicación de algunos los contenedores. - Error al ubicar el atributo total Factura dentro del contenedor de los productos de la factura.
Model	- Incluyen los datos presentes en el Paper Form y datos necesarios para la implementación de la lógica y reglas del negocio	- Solo incluye los datos presentes en el Paper Form
View model	- Grado de concordancia adecuado entre el modelo realizado por Participante	- Bajo grado de concordancia entre el modelo realizado por Participante
Tiempo Aproximado De duración	- Ambos integrantes en promedio se demoraron 10 minutos en la realización de la prueba	- Ambos integrantes en promedio se demoraron 40 minutos en la realización de la prueba

Tabla 5 Resumen del analisis de los artefactos generados por cada grupo de prueba

A continuación, se describe con mayor detalle la comparación y análisis de los modelos generados en cada grupo de prueba:

Grupo 1 – Enfoque de patrones:

El desarrollo de la prueba aplicando el catálogo de patrones como base para la definición de la view, view model y el model genero una mayor facilidad en la realización de esta tarea. Este grupo requirió menos tiempo (30 minutos aproximadamente) para la definición de los artefactos en comparación al grupo 2 (40 minutos aproximadamente), lo que demuestra que este enfoque facilita al desarrollador la interpretación del modelo mental y la construcción del prototipo de BUI.

En relación a la ***view***, ambos Participantes identificaron correctamente los elementos de la interfaz, la única diferencia consistió en los datos del emisor de la factura. El Participante 1 utilizó la variabilidad de persona natural para describir el concepto de emisor mientras que el Participante 2 aplicó la variabilidad de persona Jurídica, existiendo una similitud en el resto de las plantillas de presentación de las variabilidades de los patrones seleccionados.

Con respecto al ***Domain model***, los dos estudiantes presentaron un modelo de dominio adecuado, que representa de manera correcta el contexto de la Factura, incluyendo datos que, aunque no aparecen en el Paper Form, son necesarios para la aplicación (por ejemplo, el código del producto). La diferencia principal entre el modelo realizado por cada Participante radico en la aplicación directa del patrón por el Participante 2 (incluyo las clases abstractas de los patrones), mientras que, el Participante 1 aplico de los componentes concretos del patrón. A excepción del concepto emisor de la factura, ambos seleccionaron las mismas variabilidades de los patrones de datos.

La definición del componente ***view model*** presento una menor concordancia entre los dos Participantes, lo que indica que el nivel de subjetividad incluida por los participantes de la prueba fue alto, este pudo generarse en el hecho de que este elemento no está representado en el catálogo de patrones, en consecuencia, al no tener una plantilla o base preestablecida para su modelamiento, fue realizado totalmente basado en el modelo mental de los estudiantes que realizaron esta prueba, lo que demuestra la importancia de asociar al patrón un elemento (en nuestra propuesta el DataForm) que permita la generación de este componente.

Grupo 2 – Sin uso de patrones:

Este grupo realizó la definición del view, view model y del domain model sin el uso de patrones. En consecuencia, los estudiantes basados en su experiencia definieron estos elementos para lo cual el tiempo necesario por este grupo en la realización de la tarea fue aproximadamente 40 minutos, 10 minutos más en comparación al grupo 1 que utilizó el catálogo de patrones.

En relación a la **view**, los resultados obtenidos de los 2 alumnos fueron también similares, sin embargo, la ubicación de la información del emisor, de la fecha y numero de la factura fue distinta. Otro aspecto que resalta es que, a diferencia del grupo 1, ambos Participantes ubicaron el total de la factura en el mismo contenedor del listado de productos, siendo esto un error en el diseño debido, esto debido a que, el total de la factura es un atributo que pertenece al concepto factura y no a su detalle, en consecuencia, este atributo debe estar fuera de los contenedores.

Con respecto al **Domain model**, los estudiantes incluyeron solo la información visible en el Paper Form (modelo mental del usuario), en consecuencia, no incluyeron datos que, aunque no son necesarios en la interfaz de usuario si lo son para la lógica de la aplicación (por ejemplo, el código del producto y su fecha de creación). Esto demuestra la dificultad de definir el modelo de dominio basado totalmente en el modelo mental del usuario, esto debido a que el desarrollador debe inferir el resto de los datos

fundamentado en su experiencia y conocimiento del dominio, aumentando el nivel de subjetividad que puede incluirle al modelo.

Los **view model**, al igual que el grupo 1, presentaron un nivel alto de diferencia, demostrando que su modelamiento, aunque se tenga el Paper Form, depende de la experiencia y conocimiento del desarrollador.

8.2.5 Conclusiones de los resultados

Gracias a los resultados obtenidos, se pudo observar que el uso de los patrones aumento la calidad de los modelos view y domain, esto se evidencia en la similitud y en la calidad de los artefactos desarrollados (view y model) por los Participantes del grupo 1 en comparación al otro grupo de la prueba, asimismo el uso de patrones facilita la interpretación del modelo mental del usuario, esto se evidencia en que el grupo 1 (aplicando el enfoque de patrones) necesito menor tiempo en la elaboración del total de las tareas.

La dificultad en la definición del view – model permite identificar la importancia de contar con un artefacto que permita disminuir la complejidad y subjetividad en el modelado de este elemento o que permita su generación automática. En consecuencia, podemos sustentar lo valioso de la asociación del elemento DataForm, el cual, al incluir la información de la forma y datos de la interfaz, con los patrones propuesto logran la generación de este componente.

CAPÍTULO NOVENO

9. CONCLUSIONES Y TRABAJOS FUTUROS

La realización de este trabajo arroja como resultado un proceso metodológico para la realización de la interfaz de usuario de negocio basado en la utilización de patrones de negocios, soportada por una herramienta que soporta la realización de las actividades propuestas logrando generar muy rápidamente un prototipo de interfaz de negocio con funcionalidades mínimas y con datos de prueba reales que permiten la validación inmediata con el usuario.

Para la realización de este proceso se inició con la identificación de patrones de modelado de datos, la cual se hizo a través del estado del arte, garantizando así, que dichos patrones fueran reconocidos y validados previamente por sus autores. Los patrones por su naturaleza permiten su adaptación a múltiples contextos, estas variabilidades dependen de los dominios de cada problema, para identificarlas, se propone un método que permite basado en el modelo mental del usuario definir las variabilidades y la forma de presentación de sus datos, lo que puede disminuir problemas de usabilidad en los interfaces de usuarios generadas a partir los patrones.

Se logra con esto proponer una triada compuesta por la asociación entre la variabilidad de un patrón de datos, las plantillas de presentación y un modelo de interacción, los cuales en conjunto soportan la generación automática del prototipo de interfaz de usuario de negocios. A partir de esta triada y basado en los fundamentos de TD-MBUID se define un enfoque de ingeniería que haciendo uso de la metodología propuesta permite la generación de la interfaz de usuario de negocio, mejorando la calidad y tiempo necesario de los prototipos.

La adopción de TD-MBUID y del lenguaje DataForm fue fundamental para definir los componentes metodológicos propuestos en este trabajo, este último permite asociar a un mismo modelo de dominio múltiples formas de presentación, lo cual facilita separar la perspectiva del desarrollador de la del usuario, manteniendo la correlación de los datos necesarios en la visualización y los de la aplicación, la implementación del catálogo de patrones se realiza gracias a este elemento.

A partir de la realización de este trabajo, se puede ampliar el catálogo de patrones propuesto, involucrando patrones de interacción que permita mejorar la generación de las interfaces, asimismo, es

posible definir reglas que permitan transformar las plantillas de presentación a múltiples tecnologías como móviles y televisión, asimismo, se puede evolucionar esta colección a un lenguaje de patrones.

A nivel del componente tecnológico de esta propuesta se puede mejorar la generación de las interfaces a distintas tecnologías y lenguajes como plataformas móviles, web y lenguajes como .net y Android, por poner un ejemplo; el componente metodológico puede ser enriquecido y ampliado hasta llevarlo a la generación de la interfaz de usuario concreta.

Finalmente, como se contempla al inicio de este trabajo, se pretende continuar en la línea de trabajo del desarrollo de interfaz basada en modelo, con la construcción y utilización de métodos y herramientas que promueven su generación automática. Se espera que este trabajo pueda integrarse en esa línea para que las interfaces generadas sean usables.

REFERENCIAS

10. REFERENCIAS

Alexander, C. (1964). *Notes on the Synthesis of Form* (Vol. 5): Harvard University Press.

Alexander, C. (1975). *The Oregon Experiment*: Oxford University Press.

Alexander, C. (1979). *The Timeless Way of Building*: Oxford University Press.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*: OUP USA.

Bailey, G. (1993). *Iterative methodology and designer training in human-computer interface design*. Paper presented at the Proceedings of the INTERCHI '93 conference on Human factors in computing systems, Amsterdam, The Netherlands.

Balzert, H. (1996). From OOA to GUIs: The JANUS System. *Journal of Object-oriented Programming, 8*(9), 43-47.

Blaha, M. (2010). *Patterns of Data Modeling*: CRC Press.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (2013). *Pattern-Oriented Software Architecture, A System of Patterns*: Wiley.

Coad, P., & Mayfield, M. (1994). Object model patterns: workshop report. *SIGPLAN OOPS Mess., 5*(4), 102-104. doi: 10.1145/260060.260135

Coad, P., & Yourdon, E. (1991). *Object-oriented Design*: Yourdon Press.

Constantine, L. L., & Lockwood, L. A. D. (1999). *Software for use: a practical guide to the models and methods of usage-centered design*: ACM Press/Addison-Wesley Publishing Co.

Cross, H. (1952). *Engineers and Ivory Towers*: McGraw-Hill.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern Classification (2nd Edition)*: Wiley-Interscience.

Erickson, T. (2000). *Lingua Francas for design: sacred places and pattern languages*. Paper presented at the Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques, New York City, New York, USA.

Eriksson, H.-E., & Penker, M. (1998). *Business Modeling With UML: Business Patterns at Work*: John Wiley \\& Sons, Inc.

Erl, T. (2009). *SOA Design Patterns*: Prentice Hall PTR.

France, R., & Rumpe, B. (2007). *Model-driven Development of Complex Software: A Research Roadmap.* Paper presented at the 2007 Future of Software Engineering.

Frankel, D. S. (2004). An MDA Manifesto. *MDA Journal*. Retrieved from www.bptrends.com/publicationfiles/05-04%20COL%20IBM%20Manifesto%20-%20Frankel%20-3.pdf

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*: Pearson Education.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*: Addison-Wesley Longman Publishing Co., Inc.

Göransson, B., Lif, M., & Gulliksen, J. (2003). *Usability Design—Extending Rational Unified Process With A New Discipline.* Paper presented at the Lecture Notes in Computer Science.

Granollers, T. (2004). *Mpiu+A. Una Metodología que Integra la Ingeniería del Software, La Interacción Persona Ordenador y la Accesibilidad en el Contexto de Equipos de Desarrollo Multidisciplinares.* Doctorade, Lleida, Lleida. Retrieved from file:///C:/data/thesis/TesiToniGranollers.pdf

Gronback, R., & Roy, D. (2006). Tutorial GMF Retrieved 10-04, 2007

Gronback, R. C. (2009). *ECLIPSE MODELING PROJECT, A Domain-Specific Language Toolkit*. Boston: Addison Wesley.

Hay, D. C. (1996). *Data Model Patterns: Conventions of Thought*: Dorset House Pub.

Hay, D. C. (2010). *Data Model Patterns: A Metadata Map: A Metadata Map*: Elsevier Science.

IBM_Rational. (2003). *Too Navigator (Rational Unified Process), Concepts: Usability Engineering*.

Janeiro, J., Barbosa, S. D. J., Springer, T., & Schill, A. (2009). *Enhancing user interface design patterns with design rationale structures*. Paper presented at the Proceedings of the 27th ACM international conference on Design of communication, Bloomington, Indiana, USA.

Jaquero, V. M. L. (2005). *Interfaces de Usuario Adaptativas Basadas en Modelos y Agentes de Software.* Universidad Castilla - La Mancha.

Jouault, F., & Kurtev, I. (2006). *On the architectural alignment of ATL and QVT* Paper presented at the Proceedings of the 2006 ACM symposium on Applied computing, Dijon, France

Kaisler, S. H. (2005). *Software Paradigms*. London: John Wiley & Sons, Inc.

Lauesen, S. (2005). *User Interface Design:A Software Engineering Perspective*. Harlow: Addison-Wesley.

Lauesen, S., & Harning, M. B. (2001). Virtual Windows: Linking User Tasks, Data Models, and Interface Design. *IEEE Softw., 18*(4), 67-75. doi: http://dx.doi.org/10.1109/MS.2001.936220

Lea, D. (1994). Christopher Alexander: an introduction for object-oriented designers. *SIGSOFT Softw. Eng. Notes, 19*(1), 39-46. doi: 10.1145/181610.181617

Limbourg, Q. (2004). *Multi-Path Development of User Interfaces.* Phd. , Université catholique de Louvain, Louvain-la-Neuve. Retrieved from file:///D:/Data/thesis/PHD-Limbourg04-AfterPrinting.pdf

Limbourg, Q., & Vanderdonckt, J. (2004). UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence. In M. M & C. S (Eds.), *Engineering Advanced Web Applications* (pp. 325-338): Rinton Press, Paramus.

Lopez, O. P., Hayes, F., & Bear, S. (1992). *Oasis: An object-oriented specification language.* Paper presented at the 4th International Conference on Advanced Information Systems Engineering CAiSE-92.

Lopez, O. P., Insfr, E., Pelechano, V., Romero, J., & Merseguer, J. (1997). *OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods.* Paper presented at the Proceedings of the 9th International Conference on Advanced Information Systems Engineering.

Microsoft. (2012). The MVVM Pattern, 2015, from https://msdn.microsoft.com/en-us/library/hh848246.aspx

Molina, P. J., Belenguer, J., & Pastor, O. (2003). *Describing Just-UI Concepts Using a Task Notation* Paper presented at the Interactive Systems. Design, Specification, and Verification, 10th International Workshop, DSV-IS 2003, Madeira, Portugal.

Molina, P. J., Melia, S., & Pastor, O. (2003). *Describing Just-UI Concepts Using a Task Notation* Paper presented at the DSV-IS-2003.

Montero, F., López-Jaquero, V., & Molina, J. P. (2008). *Applying Usability Patterns in e-Commerce Applications*. Paper presented at the Proceedings of the 2008 conference on Techniques and Applications for Mobile Commerce: Proceedings of TAMoCo 2008.

Moore, B., Dean, D., Gerber, A., Wagenknecht, G., & Vanderheyden, P. (2004). *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*: ibm.com/redbooks.

Moreno, P. J. M. (2002). *Especificación de la Interfaz de Usuario: De los Requisitos a la Generación Automática.* Tesis Doctoral, Universidad Politecnica de Madrid.

Muñoz, R. A., & Orozco, W. J. G. (2014). Framework para la ejecución de modelos ejecutables específicos del dominio (pp. 33). Universidad del Quindío.

Nielsen, J. (1993). *Usability Engineering*: Morgan Kaufmann Publishers Inc.

Noble, R. J. (1997). GOF Patterns for GUI Design. *Proceedings of the European Conference on Pattern Languages of Program Design*.

OMG. (2008). Unified Modeling Language (UML) Version 2.0 Retrieved 11-2008, from http://www.uml.org/

Orozco, W. J. G. (2010). *Marco de Desarrollo de Sistemas Groupware Interactivos Basado en la Integración de Procesos y Notaciones.* Escuela Superior de Informática de Ciudad Real.

Rech, J., & Bunse, C. (2009). *Model-Driven Software Development: Integrating Quality Assurance*. Hershey, New York: Information Science Reference.

Riehle, D., Z, H., & llighoven. (1996). Understanding and using patterns in software development. *Theor. Pract. Object Syst., 2*(1), 3-13. doi: 10.1002/(sici)1096-9942(1996)2:1<3::aid-tapo1>3.0.co;2-#

Rossi, G., Schwabe, D., & Garrido, A. (1997). *Design reuse in hypermedia applications development*. Paper presented at the Proceedings of the eighth ACM conference on Hypertext, Southampton, United Kingdom.

Roth, S., & Schulz, C. (2015). *A pattern-based approach to presentations using slide facilities*. Paper presented at the Proceedings of the 18th European Conference on Pattern Languages of Program, Irsee, Germany.

Sarver, T. (2000). Pattern refactoring workshop. Position paper, OOSPLA 2000, , from http://www.laputan.org/patterns/positions/Sarver.html

Schmidt, D. C., Stal, M., Rohnert, H., & Buschmann, F. (2013). *Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects*: Wiley.

Silva, P. P. d. (2002). *Object Modelling of Interactive Systems: The UMLi Approach.* Phd. Thesis, University of Manchester, Manchester.

Silverston, L., & Agnew, P. (2009). *The Data Model Resource Book: Universal Patterns for Data Modeling*: Wiley Publishing.

Simarro, F. M. (2005). *Integración De Calidad Y Experiencia En El Desarrollo De Interfaces De Usuario Dirigido Por Modelos.* Doctor en Informática Doctoral, Universidad de Castilla -La Mancha.

Sowa, J. F., & Zachman, J. A. (1992). Extending and formalizing the framework for information systems architecture *IBM Syst. J*, 590-616

Steinberg, D., Budinsky, F., Paternostro, M., & Merks, E. (2009). *EMF: Eclipse Modeling Framework 2.0*: Addison-Wesley Professional.

Szekely, P. A., Sukaviriya, P. N., Castells, P., Muthukumarasamy, J., & Salcher, E. (1995). *Declarative interface models for user interface construction tools: the MASTERMIND approach.* Paper presented at the Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction.

Tamayo, M. T. (1999). *El Proyecto De Investigación* (Vol. 5). Bogotá.

Tidwell, J. (2010). *Designing Interfaces*: O'Reilly Media.

Toxboe, A. (2014). User Interface Design patterns, from http://ui-patterns.com/

Tran, V. (2010). *UI generation from task, domain and user models: the DB-USE approach*. Paper presented at the Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, Berlin, Germany.

Vanderdonckt, J. (1997). *Conception assistée de la présentation d'une interface homme-machine ergonomique pour une application de gestion hautement interactive.* Phd. PHD Thesis, Namur, Belgium.

Vanderdonckt, J., & Simarro, F. M. (2010). *Generative pattern-based design of user interfaces*. Paper presented at the Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems, Berlin, Germany.

Welie, M. v. (2008). Patterns in Interaction Design, from http://www.welie.com/patterns/

Zachman, J. A. (1987). A Framework For Information Systems Architecture. *IBM Ssystems Journal, 26*(3).

ANEXO I

ANEXO I ARTEFACTOS GENERADOS EN LA PRUEBA

A continuación, se presentan los artefactos generados por cada uno de los participantes en la realización de la prueba realizada.

A.1.1 ARTEFACTOS DEL GRUPO 1

El enfoque de patrones fue aplicado por el grupo 1, en el cual participaron 2 estudiantes de último semestre de ingeniería de Sistemas y Computación.

Modelos realizados por Participante 1:

Los artefactos definidos por el participante 1 se muestran en las Figura 99, Figura 100 y Figura 101

VIEW

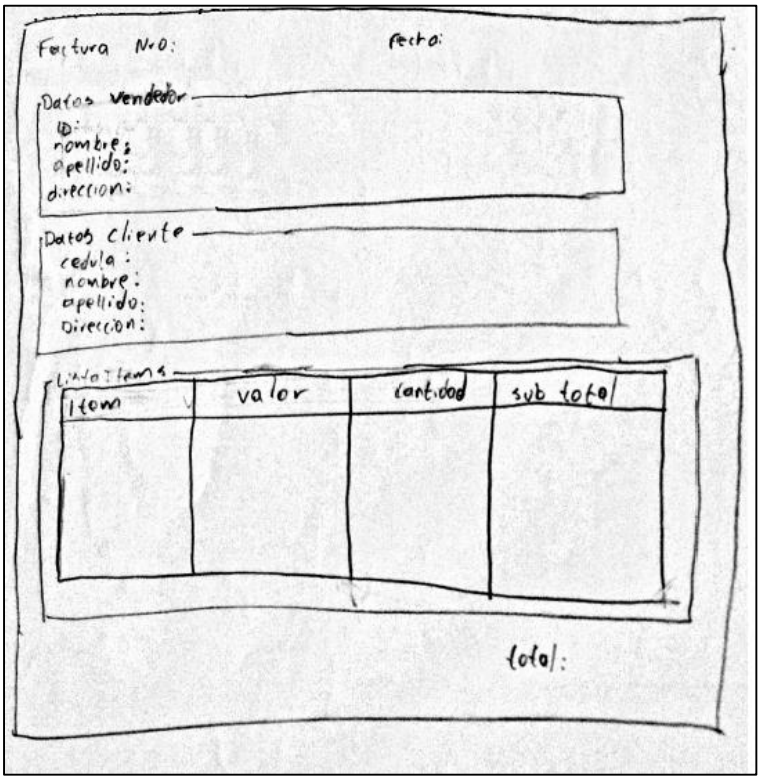

Figura 99 View realizada por el participante 1 del Grupo 1

DOMAIN MODEL

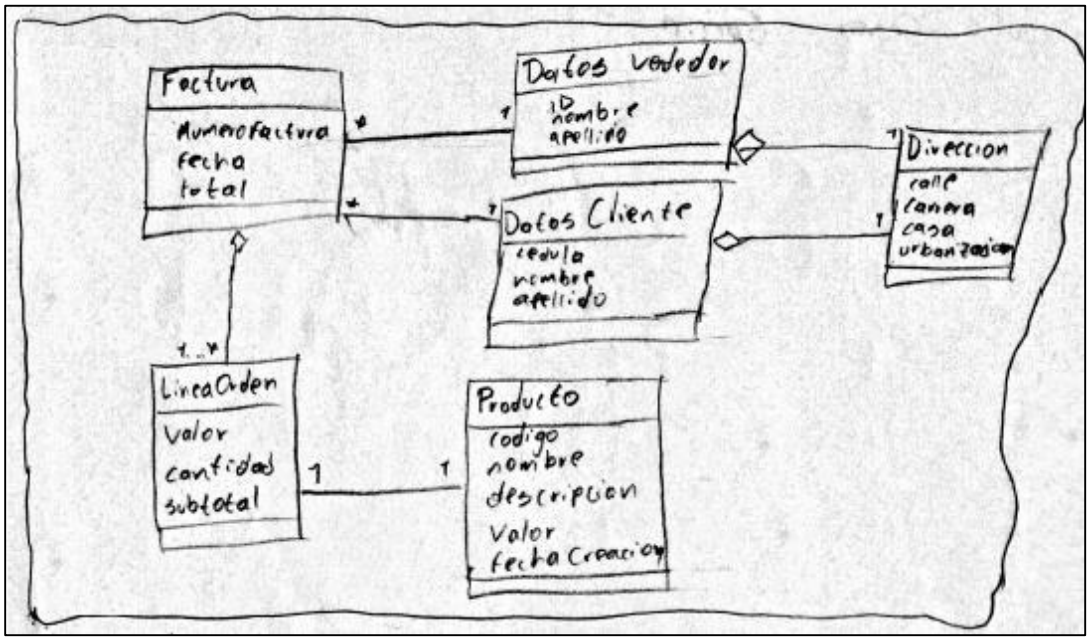

Figura 100 Domain model Realizado por el participante 1 del grupo 1

VIEW MODEL

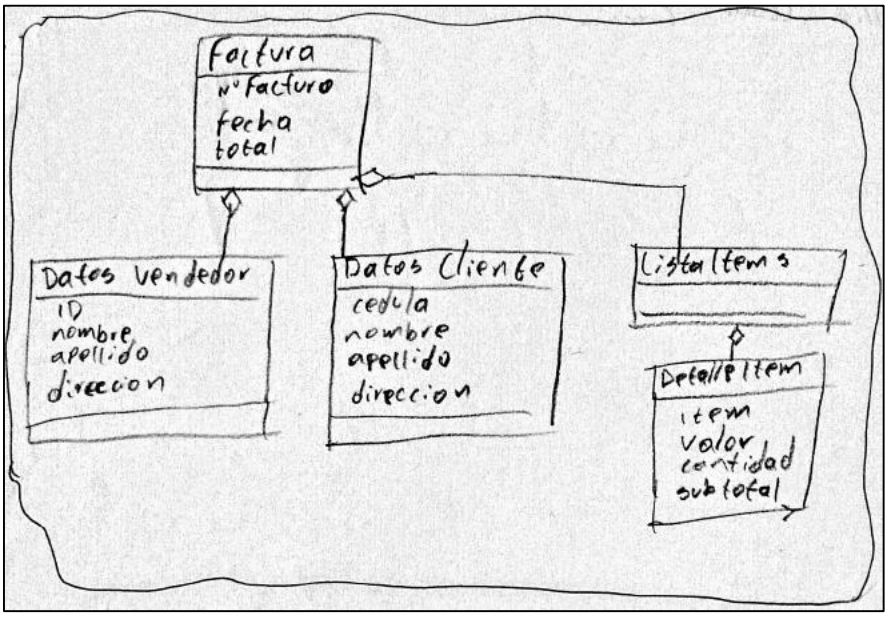

Figura 101 View realizada por el participante 1 del grupo 1

Modelos realizados por Participante 1:

Los artefactos definidos por el participante 1 se muestran en las Figura 102Figura 103Figura 104

VIEW

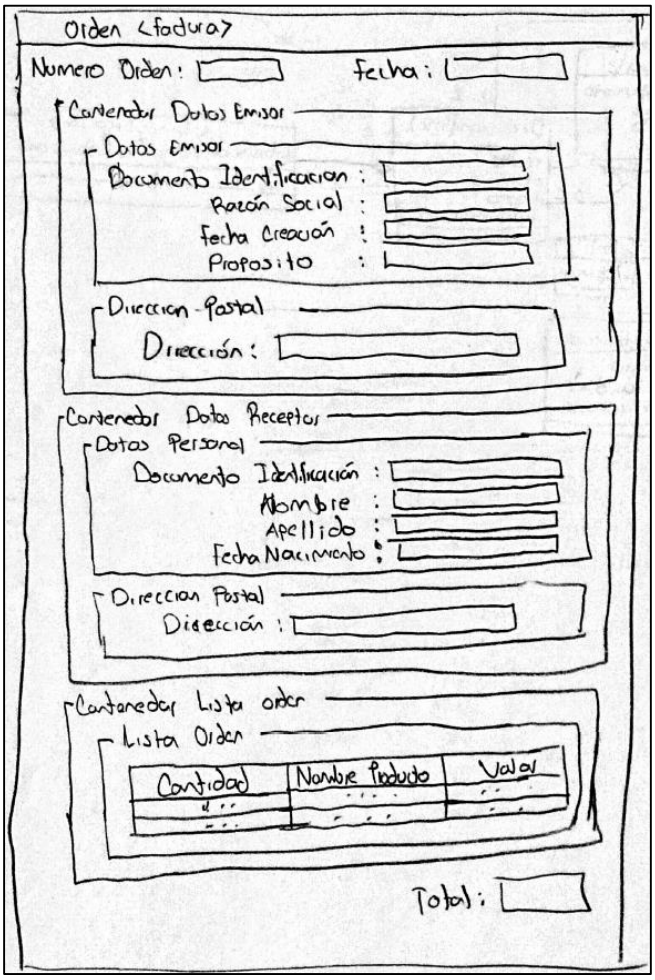

Figura 102 View realizada por el participante 2 del grupo 1

DOMAIN MODEL

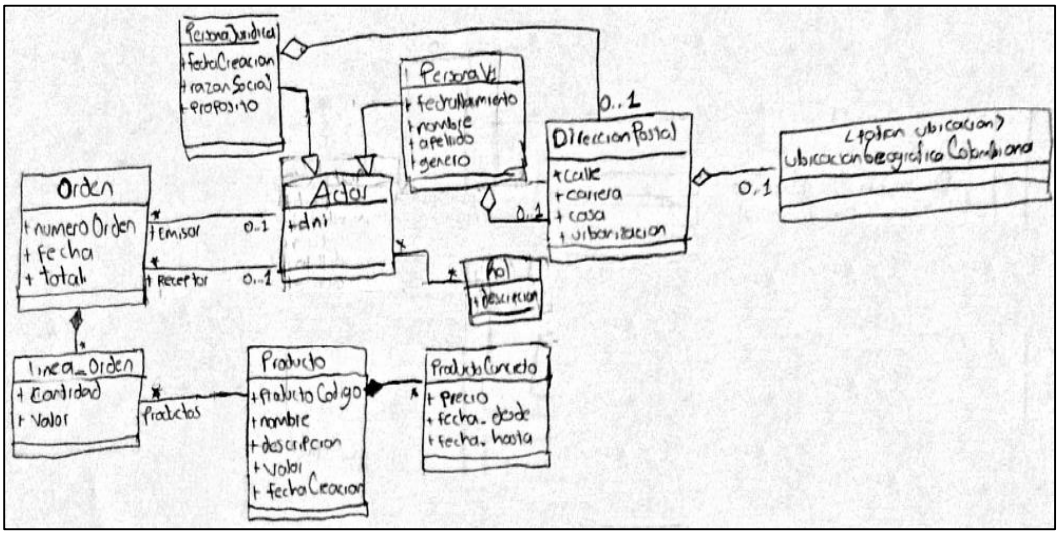

Figura 103 Domain model Realizado por el participante 2 del grupo 1

VIEW MODEL

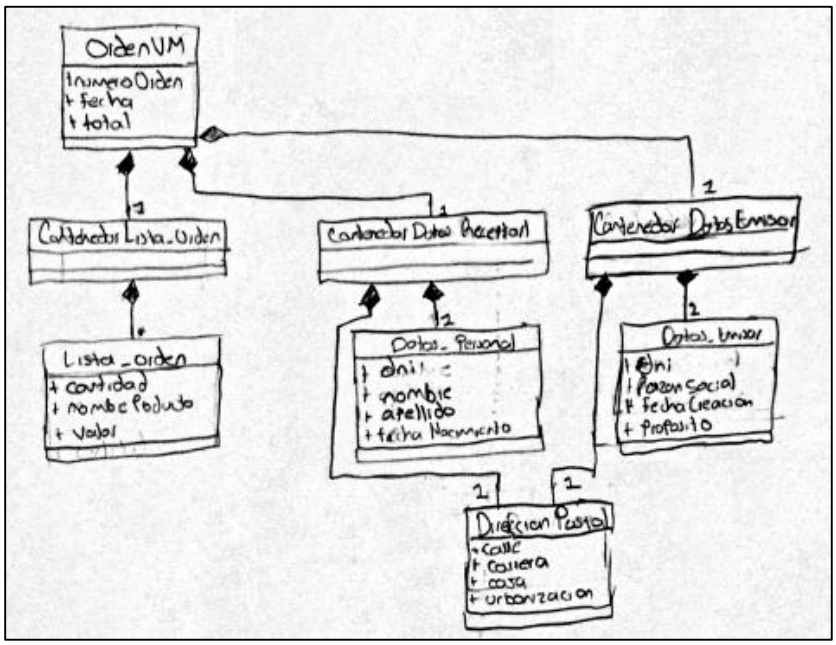

Figura 104 View model realizada por el participante 2 del grupo 1

A.1.2 ARTEFACTOS DEL GRUPO 2

El grupo 2 realizó la construcción de los modelos view, view model y model sin la aplicación de los patrones, en este grupo participaron 2 estudiantes de último semestre de ingeniería de Sistemas y Computación.

Modelos realizados por Participante 1:

Los artefactos definidos por el participante 1 se muestran en las Figura 105, Figura 106 y Figura 107.

VIEW

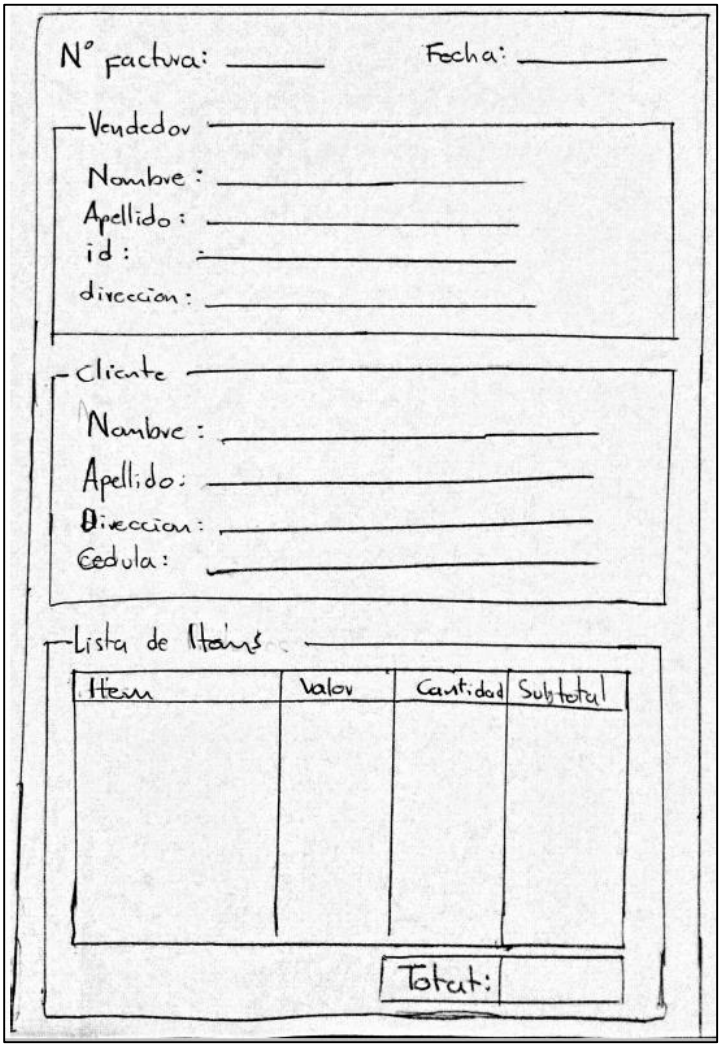

Figura 105 View realizada por el participante 1 del grupo 2

DOMAIN MODEL

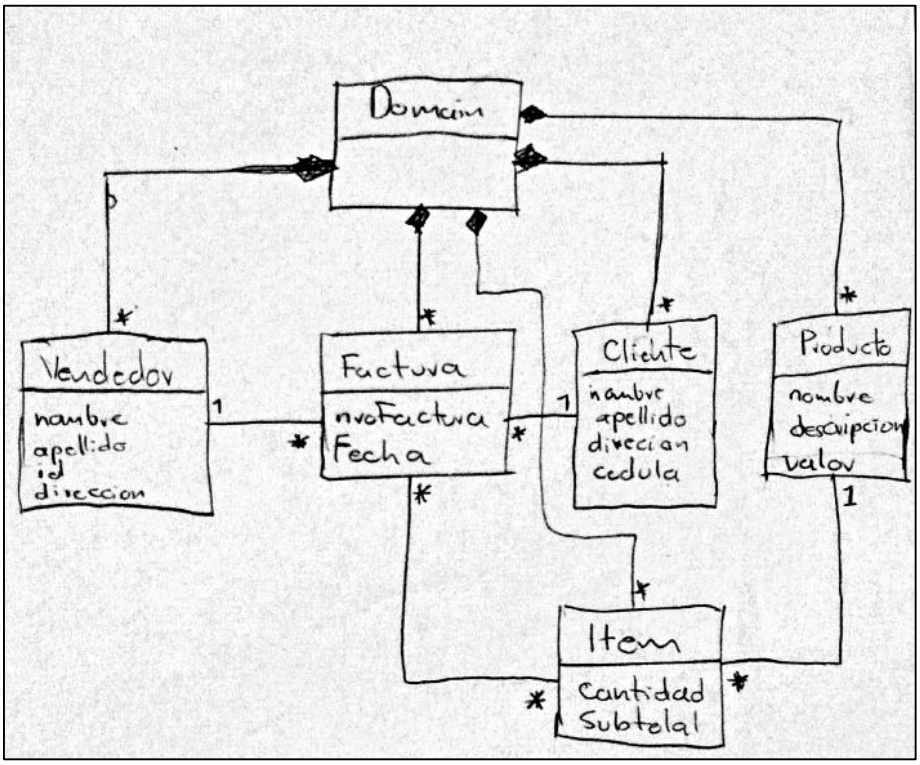

Figura 106 Domain model Realizado por el participante 1 del grupo 2

VIEW MODEL

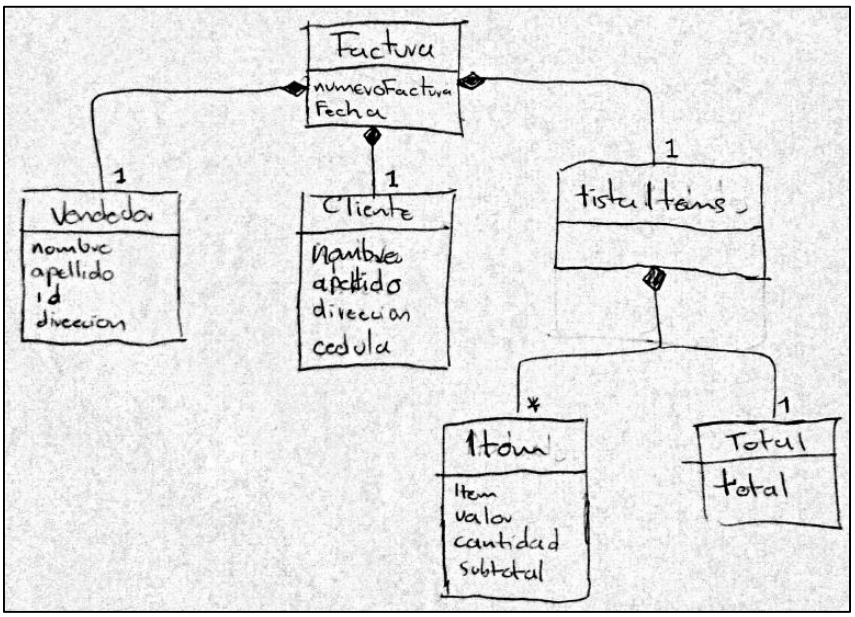

Figura 107 View model realizada por el participante 1 del grupo 2

Modelos realizados por Participante 1:

Los artefactos definidos por el participante 1 se muestran en las Figura 108Figura 109 y Figura 110

VIEW

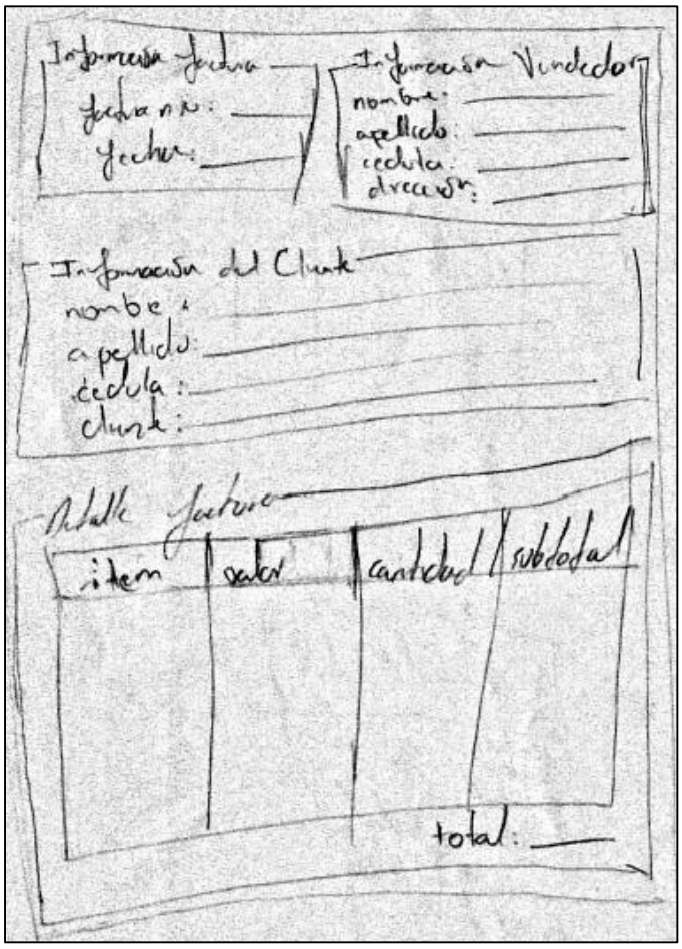

Figura 108 View realizada por el participante 2 del grupo 2

DOMAIN MODEL

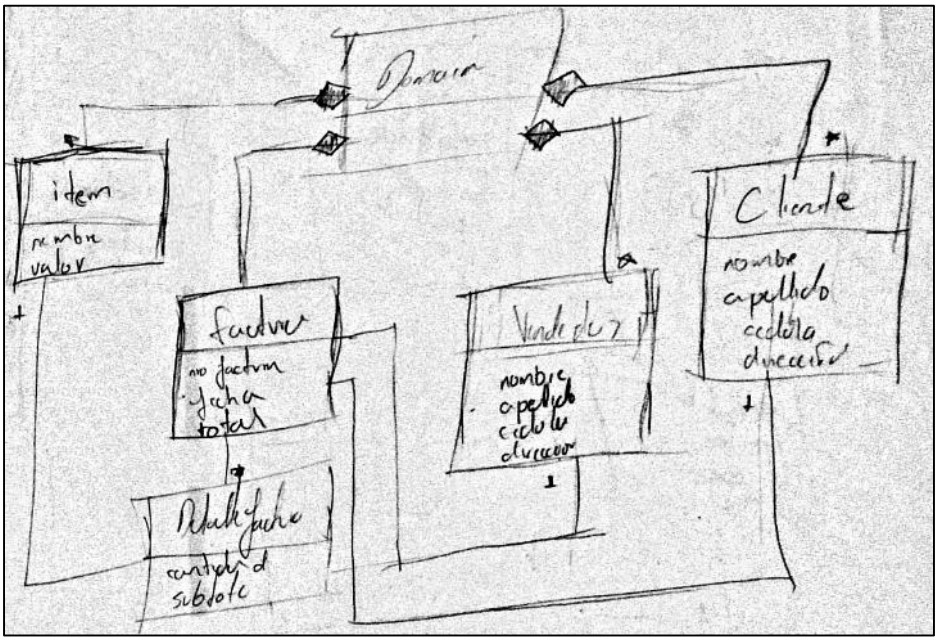

Figura 109 Domain model Realizado por el participante 2 del grupo 2

VIEW MODEL

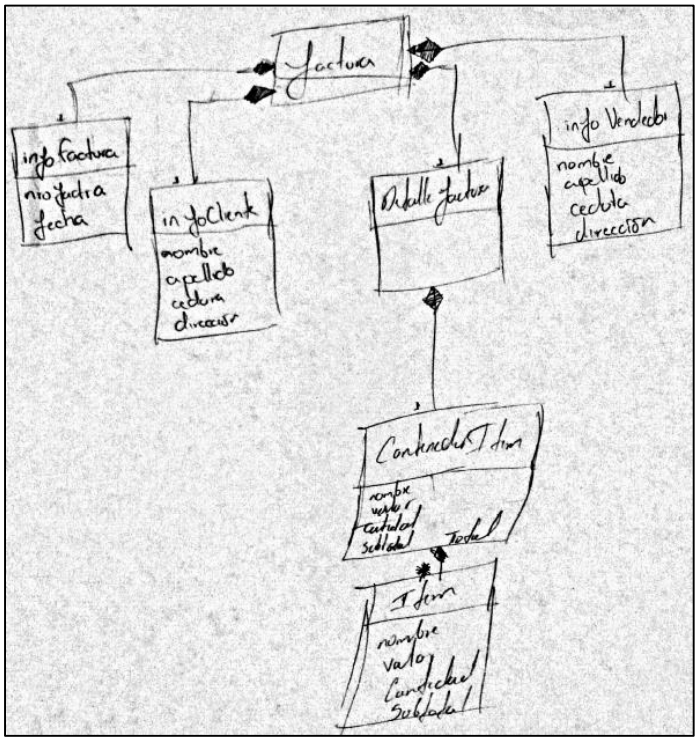

Figura 110 View model realizada por el participante 2 del grupo 2