Centro de Investigación en Matemáticas, A. C.

Localizando, Identificando y Midiendo la Brecha Representacional Entre el Modelo del Dominio y el Modelo del Diseño en el Proceso Unificado

Tesis
que para obtener el grado de

Maestro en Ciencias con Especialidad en
Ciencias de la Computación
y Matemáticas Industriales

presenta

Omar Posada Villarreal

Director de tesis:
Dr. Cuauhtémoc Lemus Olalde
Co-director de tesis:
M.C. Luis Felipe Fernández Martínez

Guanajuato, Gto., México, a 31 de mayo del 2004.

A mis queridos padres, Alfredo y Elva,
a mi hermano, Hugo Alfredo,
por su incondicional apoyo y constante cariño.
Gracias

Agradecimientos

Estoy convencido de que este trabajo es el fruto del apoyo de muchas personas que me alentaron, me enseñaron y me tuvieron confianza. Quiero expresar mi gratitud a todas ellas.

Al Dr. Cuauhtémoc Lemus Olalde y al M.C. Luis Felipe Fernández Martínez, mis directores de tesis, por su gran apoyo, su paciencia, sus consejos y por todas las tardes que dedicaron a orientar mi rumbo, gracias por este gran esfuerzo.

Al Dr. Miguel Ángel Serrano por ser parte del comité de evaluación, así como por la honestidad y claridad de su opinión para mejorar este trabajo.

Al M.C. Gerardo Padilla por sus sabios consejos en la realización del presente trabajo.

A todos mis profesores que me enseñaron y alentaron pacientemente durante mi estancia en el CIMAT. Cada uno de ellos dejó una experiencia útil en mi desarrollo profesional.

Al CONACYT y CONCYTEG por apoyarme con sus programas de beca que me permitieron estudiar una maestría de calidad.

A todas las personas de la comunidad del CIMAT por brindarme una casa de estudios con un ambiente agradable, trabajador y humano que nos tuvo la confianza para sentirnos libres en su uso adecuado. Aprecio el gran esfuerzo que se necesitó para ofrecernos un espacio y un equipo de cómputo que es de lo mejor. En especial, a Ma. Dolores Aguilera, Berenice Segovia, Ismael Olmos, Alicia Luna, Ricardo Martínez y José Castro. Así como al Profr. Francisco Mirabal (gracias por la bienvenida), Dr. Adolfo Sánchez y M.A. Rafael Ixta, por apoyarme académica y administrativamente. Al Dr. Arturo Hernández Aguirre por su tutoría y trato amable.

A mis amigos y compañeros de generación, Verónica Rodríguez, Rogelio Salinas, Daniel Alba, Joaquín Peña, Julio César Estrada y Alonso Ramírez, por todas las peripecias de la vida y académicas que enriquecieron el acervo para mis nietos. Sin mencionar a David Flores por no gustarle la trigonometría, ni su pedido especial.

Muy en especial a mis inseparables amigos Alonso Ramírez Manzanares y Julio César Estrada Rico por todo su apoyo, tiempo, retos, consejos y sentido del humor.

A Laura Rivera (por su alegre risa y su clásico del alce), Cynthia Álvarez (por sus suaves y delicados mimos), Ivete Sánchez, Brenda Tapia, Sandra Gaona, Verónica Rodríguez, David Flores, Alonso Ramírez, Patreck Chikiwanda y Luke Orawo, que cultivaron con ahínco en la porra maldita, la esperanza del espectacular equipo de Posgrado: Lenin Echavarria (fundador), Luis Chan, Geiser Villavicencio (por su humildad), Julio Estrada (domador de fieras), Rogelio Salinas (por el gol rodante), Daniel Alba (por sus prisas), Patricio Alba, Manuel Ramírez (porterazo) y José Luis Trejo (capitán). A Estela Ruiz por su aprecio y sus piquetes de aguja.

Además de las clases, quisiera darle las gracias a al Dr. Johan Van Horebeek por alentarme a entrar a la maestría y darse sus rondas para levantarnos la moral; Stephanie Dunbar por el aliento, consejo sabio y discusiones retadoras; a la Dra. Graciela González, por su paciencia en las asesorías y por su trato norteño; y al Dr. José Luis Abreu por su motivación. Gracias a ellos, seguiré preparándome.

A los problemáticos Víctor Muñiz, Ángel Muñoz y Jorge Cruz, así como al apacible Omar Tzec, Rosa Icedo, Yolanda de la Lira, Moisés Trejo y Rocky Bizuet. Tuve la fortuna de conocer a varias generaciones de la maestría y del doctorado, entre ellos, Ely Gallo, Teresa Alarcón, Juan Antonio Muñoz, Edgar Arce, Fernando Gómez, Arturo González, Cuauhtémoc López y Gerardo Padilla. Y a tantos amigos que compartimos muchas experiencias agradables y divertidas en el CIMAT, como Marco Bertani, Dan-El Vila y Marte Ramírez de la FAMAT por contagiarnos de la alegría y el humor en las situaciones de tensión. Así como a amigos de otros programas, Jerónimo Castro por trabajar con las configuraciones. A mis amigos estadísticos Yanink Caro, Pilar Arcos, Valentina Castellanos (un abrazo a Valery), Antonio Andrade, Daniel Fernández, Armando Flores, Antonio Murillo y Esteban Bracamontes.

A mis amigos de Chihuahua, Eduardo Ruiz, Andrés Bustamante, José Luis Gómez, Martha Villalobos, Jorge González, Francisco y David Andujo, por su ayuda y confianza.

A mis cuatachas de Guanajuato, por su carismático humor Rocío Espinoza, Noemí, Ruiz, Brenda Zavala, Rosalba Negrete e Ivonne Razo.

Gracias al M.C. Agustín Pacheco, mi padrino, fue que encontré el caminito a la escuela.

Quisiera reiterar mi agradecimiento a mis padres, Elva y Alfredo, mi hermano Hugo y al tremendo Erick. Así como a mi familia, en especial a la memoria de mis abuelitas Polita (QEPD), Tita (QEPD) y Lolita (QEPD).

Espero me entiendan aquéllas personas que por el momento, haya omitido su nombre, por lo que les pido disculpen mi distraimiento.

Atentamente, Omar

Índices

Resumen del contenido

Contenido

Figuras

Tablas

Resumen

Introducción

La necesidad de más y mejores aplicaciones de software exige un mejoramiento en el producto y proceso del desarrollo del software. Es ampliamente reconocido que la medición del software es un aspecto importante para el control de este mejoramiento. Es por ello que existe una demanda creciente de métricas de software y mejores paradigmas de desarrollo de software.

Generalmente, el desarrollo del software se compone de varias etapas: análisis, diseño, implementación, pruebas, etc. Varios autores han encontrado que entre más temprano se inyecta un defecto y más tarde se le repara, éste es más costoso. Es decir, los defectos en las etapas tempranas (e.g. análisis y diseño) son muy costosos.

Problema

Sin embargo, en la literatura hay pocas métricas para las etapas tempranas y mucho menos para la correspondencia entre los productos de distintas etapas. A este fenómeno de correspondencia, Larman lo llama Brecha Representacional, que es la diferencia entre la representación del modelo mental (problema) y su representación en software (solución). Él sólo identifica esta situación mencionando que es "difícil de cuantificar" y que intuitivamente una Brecha estrecha es útil. Sin embargo, no propone una solución a este problema.

Solución propuesta

En la presente investigación se propone un marco de trabajo que mide la correspondencia entre las Disciplinas (etapas) del desarrollo del software. En general, se caracteriza la Brecha Representacional entre las Disciplinas del Proceso Unificado, mediante las Discrepancias Ontológicas. En particular, esta caracterización se concreta en la localización, identificación y medición de la Brecha Representacional entre el Modelo del Dominio y el Modelo del Diseño, caracterizando ciertos elementos de modelado (paquetes, clases y atributos). Cabe mencionar que la medición de la Brecha Representacional requiere de su identificación, que a su vez necesita de su localización. d Coloquialmente, se detecta qué tanto corresponde el trabajo del analista con el trabajo del diseñador.

Para localizar las Discrepancias (i.e. establecer la correspondencia) se eligieron las dependencias de refinamiento; para identificarlas, la matriz de refinamiento (i.e. digrafo bipartito); y para medirlas, se propone el Conteo Ponderado. Para evaluar la eficiencia de ambos modeladores (e.g. analista y diseñador), se mide el cambio de las métricas de la Brecha Representacional. Para lograr la caracterización de la Brecha Representacional, se mejora la definición de las Discrepancias Ontológicas del modelo Bunge-Wand-Weber. Finalmente, la propuesta se explica con un ejemplo y se prueba con dos casos de estudio.

Article abstract

The need for more and better software applications demands improvement in software processes and products. As a result, there is an increasing demand for software metrics and better development approaches. This research focuses on characterizing (i.e. locating, identifying, and measuring) the representational gap, which is the interval between the mental model of the problem (e.g. domain model in the Unified Process) and the software representation of its solution (e.g. design model). These models were chosen because it is cheaper to identify defects in the early stages. The ontological discrepancies from an improved Bunge-Wand-Weber model, were used for characterizing this gap, and were represented as a bipartite digraph matrix. Also, the efficiency of the modelers was measured. Colloquially, this research detects how much the analyst work matches the designer work. Two cases of study were tested with an implementation of the proposed algorithm.

Keywords

Main keywords: Software Metrics, Representational Gap, Unified Process (UP), Ontological Discrepancies, Bipartite Graph Applications.
Other keywords: Software Engineering, Domain Model, Design Model, Unified Modeling Language (UML), Class Diagrams, Bunge-Wand-Weber Ontology, Relationship Matrix.

Resumen del artículo

La necesidad de más y mejores aplicaciones de software exige un mejoramiento en el producto y proceso del desarrollo del software. En consecuencia, hay una demanda creciente de métricas de software y mejores paradigmas de desarrollo. Esta investigación se enfoca en la caracterización (i.e. localización, identificación y medición) de la Brecha Representacional que es el intervalo entre el modelo mental del problema (e.g. Modelo del Dominio en el Proceso Unificado) y la representación en software de su solución (e.g. Modelo del Diseño). Estos modelos fueron elegidos porque es más barato identificar defectos en las etapas tempranas. Las Discrepancias Ontológicas de un modelo Bunge-Wand-Weber mejorado, se emplearon para caracterizar esta brecha y fueron representados mediante una matriz de un digrafo bipartito. Además, la eficiencia de los modeladores es medida. Coloquialmente, esta investigación detecta qué tanto el trabajo del analista corresponde con el trabajo del diseñador. Dos casos de estudio fueron probados con la implementación del algoritmo propuesto.

Palabras clave

Palabras clave principales: Métricas de Software, Brecha Representacional, Proceso Unificado (UP), Discrepancias Ontológicas, Aplicaciones de Grafos Bipartitos.
Otras palabras clave: Ingeniería de Software, Modelo del Dominio, Modelo del Diseño, Lenguaje de Modelado Unificado (UML), Diagramas de Clases, Bunge-Wand-Weber Ontology, Relationship Matrix.

Parte 1. Problema

Capítulo 1. Introducción

1.1 Necesidad

"No puedes controlar lo que no puedes medir" DeMarco[1]

La necesidad de más y mejores aplicaciones de software exige un mejoramiento en el proceso de desarrollo del software ("SPI, Software Process Improvement"). Es ampliamente reconocido que la medición es un aspecto importante para el control de este mejoramiento. Esto ha tenido dos efectos: la necesidad de mejores paradigmas para el desarrollo del software (el más prominente ha sido la Orientación a Objetos); y una demanda creciente de métricas de software con que administrar al proceso de desarrollo del software [Chidamber94]. En la práctica, existe la iniciativa de aplicar algunas metodologías exitosas de mejoramiento de la calidad provenientes de la manufactura, en la industria del software [Siviy01, Tayntor03].
Cabe mencionar que encontrar los defectos en las etapas tempranas evita su propagación a otros productos de la etapa actual y de las posteriores, evitando aumentos en tiempo y costo del desarrollo del software [Humphrey95] (ver "Costos de la identificación de defectos de software" en la página 43). Por lo que una medición en las etapas tempranas puede ayudar en la reducción de recursos.

[1] "You cannot control what you cannot measure". [Fenton97]
[2] Por sus siglas "Define, Measure, Analyze, Improve, Control"

1.2 Vista general e índice temático

1.2.1 Vista general

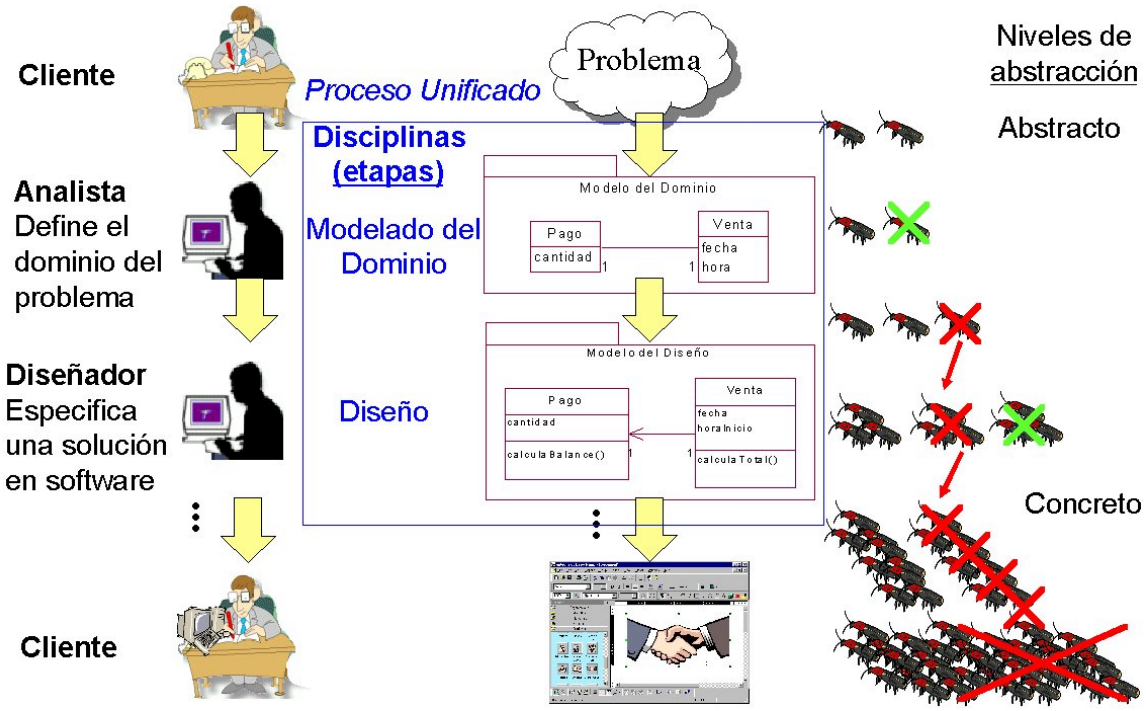


Figura 1.1. Vista general de la tesis

Cuando un cliente desea resolver un problema mediante una aplicación informática, se lo encarga a los desarrolladores de software. Generalmente, el cliente le comunica sus requerimientos al analista de software. En base a esta información, el analista define el dominio del problema y pasa esta información al diseñador de software, quien especifica una solución en software de acuerdo al problema. Esta especificación se transfiere al programador y así, sucesivamente hasta que se entrega la aplicación al cliente. Este proceso es conocido como desarrollo en cascada, que se muestra en la columna izquierda de la Figura 1.1.

Aunque la metodología del Proceso Unificado es un desarrollo iterativo (es decir, se hacen varios desarrollos en cascada para aproximarse a la solución), la presente investigación se enfoca a describir las relaciones de una iteración entre dos Disciplinas (etapas): el Modelado del Dominio y el Diseño. En estas Disciplinas, el analista, en base al problema presentado por el cliente, define el Modelo del Dominio y el diseñador especifica el Modelo del Diseño. Después de pasar por las demás Disciplinas (al final de la iteración), el cliente recibe una aplicación informática que se supone satisface sus necesidades [Larman01]. Esto se muestra en la columna central de la Figura 1.1.

Conforme los modelos abstractos (i.e. de alto nivel de abstracción) se van concretando, se va acercando a una solución real de software. Sin embargo, también se van acumulando defectos. Es decir, en cada paso de un modelo a otro modelo, los modeladores van introduciendo defectos desapercibidamente. Además, cada defecto que se deja pasar a la siguiente Disciplina usualmente genera otros [Humphrey95]. De alguna forma, se supone que los defectos se deben de corregir en cada Disciplina. La presente investigación ofrece una forma de identificar los defectos que se generan entre el Modelo del Dominio y el Modelo del Diseño, para evitar su propagación a las siguientes Disciplinas. Esto se representa en la columna derecha de la Figura 1.1, con las cruces sobre los bichos (eliminación de defectos) unidas por flechas.

1.2.2 Temas específicos de la tesis

Los roles de los interesados directos de la tesis son el verificador de los modelos (quien se encarga de identificar los defectos entre las Disciplinas), el analista y el diseñador de sistemas de software.

El verificador evalúa la correspondencia entre los trabajos de los modeladores para asegurar que la solución esté de acuerdo con la definición del problema. Para ello, se caracterizará la Brecha Representacional que es "el intervalo entre el modelo mental del dominio que pensamos y su representación en software" [Larman01]. En el Proceso Unificado, el modelo mental es el Modelo del Dominio y su representación en software, el Modelo del Diseño. Las métricas de software proveen un marco que sirve para la evaluación cuantitativa de este tipo de problemas.

Aunque es "difícil de cuantificar" la Brecha Representacional [Larman01], se propone una caracterización mediante un análisis de Discrepancias Ontológicas. Las Discrepancias son las diferencias entre las representaciones de las "cosas" del dominio [Opdahl02]. Este análisis incluye la localización, identificación y medición de las Discrepancias mediante una matriz de refinamiento (i.e. digrafo bipartito). Para la identificación, se mejora el modelo Bunge-Wand-Weber, que enmarca las consideraciones del alcance del presente trabajo.

Una vez que se midieron y analizaron las Discrepancias, los modeladores deberán elegir si justificarlas o eliminarlas. A diferencia de [Larman01], se considera que una Brecha estrecha o nula no necesariamente es lo mejor, por lo que se propone entenderla y controlarla sistemáticamente. Después de las correcciones, se vuelven a medir las Discrepancias y se comparan las métricas para evaluar la eficiencia (i.e. cuántos cambios fueron necesarios para corregir los modelos) de los modeladores. Todo esto con la finalidad de reducir los defectos, así como su propagación a otros elementos (e.g. clases) y Disciplinas.

Para saber más sobre estos temas se recomienda consultar los títulos de la tesis de la Tabla 1.1.

Tabla 1.1. Índice por aparición en los temas específicos de la tesis

1.3 Problema

1.3.1 Alcance de la tesis

El alcance del presente trabajo se acota a la metodología del Proceso Unificado (ver "Selección del Proceso Unificado" en la página 46). Debido a la importancia de las Disciplinas tempranas se estudiarán los artefactos del Modelado del Dominio y del Diseño (ver "Selección de los Artefactos" en la página 60). Dado que el único artefacto del Modelado del Dominio es un diagrama de clases (sin métodos), no se consideran otras especificaciones (e.g. diagramas de secuencia). La importancia de los elementos seleccionados se puede ver en "Selección de los elementos" de la página 62. En resumen:

Metodología de desarrollo del software: Proceso Unificado (UP)
⊃ Disciplinas: Modelado del Dominio, Diseño
⊃ Artefactos: Modelo del Dominio, Modelo del Diseño
⊃ Especificaciones: Diagramas de clases
⊃ Elementos: Clases, paquetes y atributos[3]

[3] En este contexto, el símbolo de subconjunto "⊃" se refiere a una composición de uno a muchos. Es decir, una Disciplina está compuesta por uno o más Artefactos.

1.3.2 Justificación

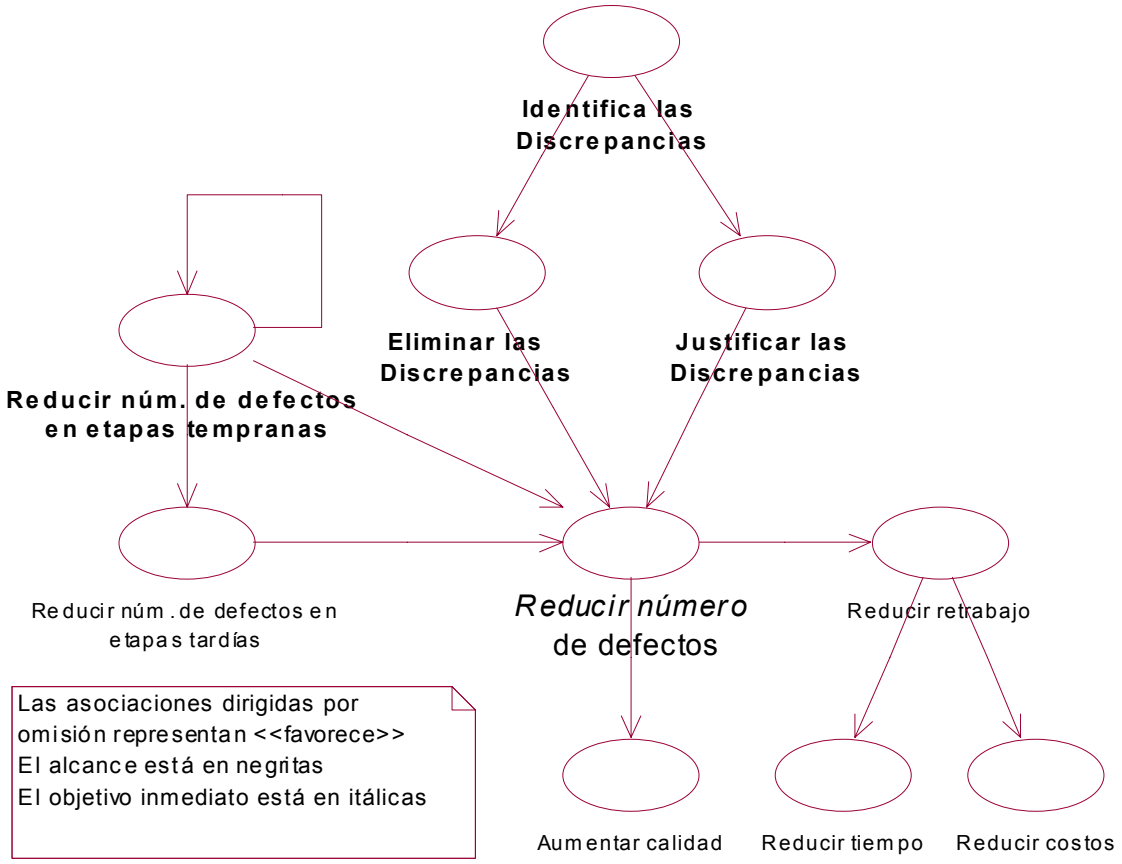


Figura 1.2. Justificación del trabajo

Uno de los beneficios que se desea lograr es la reducción del número de defectos. En especial, en las Disciplinas tempranas pues la reducción impacta a la Disciplina actual y a las posteriores. Cuando se analizan las Discrepancias Ontológicas, se está en condiciones de justificarlas y eliminarlas, con lo que se reducen defectos potenciales. Con ello, se espera aumentar la calidad, evitar aumentos en el tiempo de entrega y los costos involucrados en el desarrollo del software.

Se tienen dos áreas de oportunidad (a) y (b) para la investigación:

(a) Sería deseable medir la correspondencia entre el Modelo del Dominio (conceptualización del problema) y el Modelo del Diseño (su solución) porque:
   1. Las Disciplinas tempranas tienen un impacto crítico en el desarrollo del software pues un defecto en ellas afecta a otros elementos en la misma y en otras Disciplinas (ver "Costos de la identificación de defectos de software" en la página 43).
   2. Hay pocas métricas de software para las Disciplinas tempranas del Proceso Unificado (ver "Métricas de software de la literatura" en la página 57).
   3. Es deseable una Brecha Representacional estrecha, pero es difícil de cuantificar.

[4] Este no es propiamente un Diagrama de Casos de Uso, sino una visualización de causa-efecto.

(b) Sería deseable medir elementos de diferentes Disciplinas pero de naturaleza similar porque:
1. La mayoría de las métricas son Intradisciplinarias o se tienen razones entre elementos de distinta naturaleza –Multidisciplinarias– (ver "Métricas Intradisciplinarias, Interdisciplinarias y Multidisciplinarias" en la página 74).
2. No se encontraron métricas para elementos de misma o similar naturaleza y de diferente Disciplina (ver "Métricas de naturaleza similar" en la página 74).

1.3.3 Definición del problema

La pregunta general que se tratará de responder brevemente es:

- ¿Cómo se evalúa la correspondencia entre el trabajo del analista y el diseñador?

Y en particular,

- ¿Cómo se caracteriza la Brecha Representacional entre el Modelo de Dominio y el Modelo de Diseño?
  - Localizando los elementos participantes en las Discrepancias Ontológicas mediante las dependencias de refinamiento
  - Identificando el tipo, categoría y orden de las Discrepancias mediante la matriz de refinamiento (i.e. dígrafo bipartito)
  - Midiendo las Discrepancias mediante el Conteo Ponderado
- ¿Cómo detecto defectos potenciales en el Modelo de Dominio y el Modelo de Diseño?
  - Analizando las Discrepancias Ontológicas entre los elementos del Modelo del Dominio y del Modelo del Diseño. Es decir, cuestionando al analista y al diseñador la existencia de las Discrepancias
- ¿Cómo se mide la eficiencia del trabajo conjunto del analista y el diseñador?
  - Comparando las métricas del modelo inicial con las del modelo corregido

1.4 Organización de la tesis

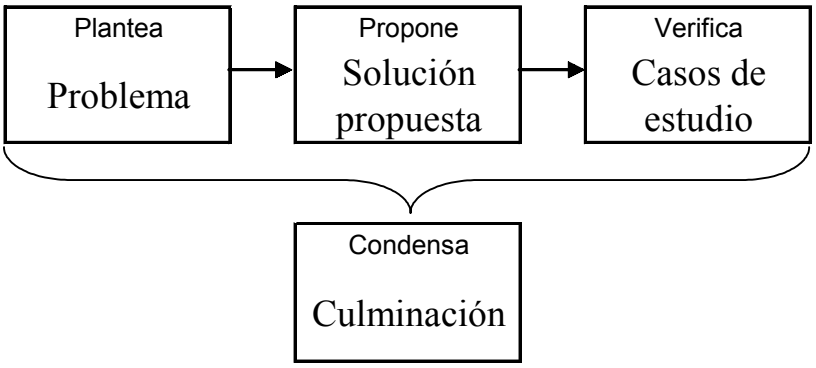


Figura 1.3. Partes de la tesis

Para establecer una distinción clara entre el conocimiento existente y el nuevo, la tesis se divide en cuatro partes: problema, solución propuesta, casos de estudio y la culminación. En la Tabla 1.2, se muestran las convenciones para la presentación de los temas.

Tabla 1.2. Convenciones para la organización de la tesis

Estilo de letra	**Descripción**
Negritas y fondo gris	Contribución
Itálicas	Breve introducción
Regular	Argumentación o título

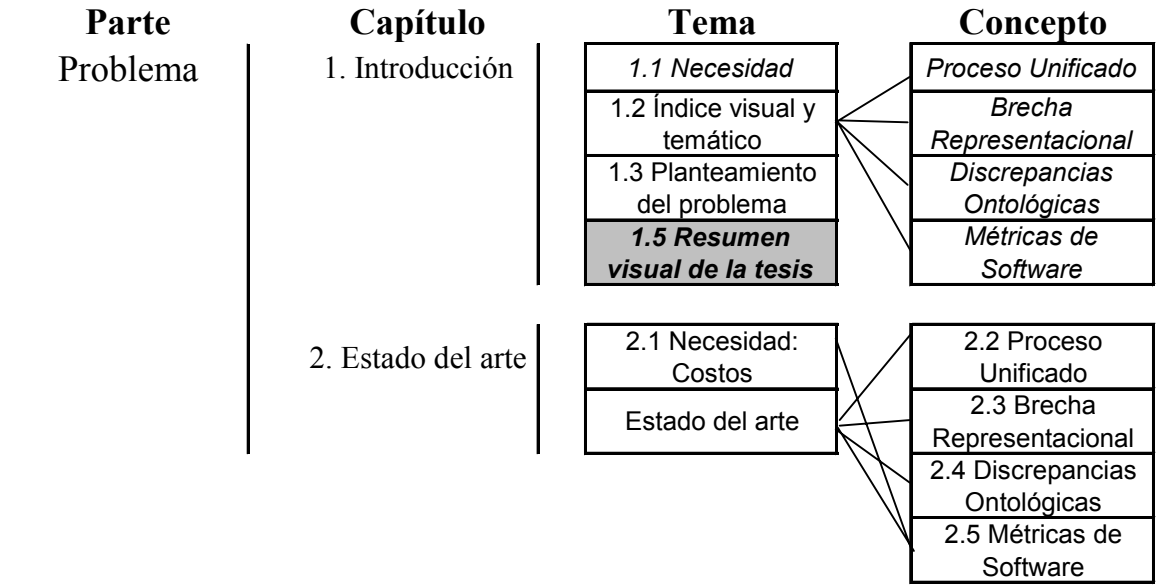


Figura 1.4. Problema

En el Capítulo 1, se plantea el problema proporcionando una idea inicial sobre los conceptos usados y la solución propuesta. En este resumen se incluyen las páginas de los temas detallados. La justificación detallada, los conceptos y el estado del arte se explican en el Capítulo 2. Básicamente se manejan cuatro temas: Proceso Unificado, Brecha Representacional, Discrepancias Ontológicas y Métricas de Software. El lector familiarizado con estos temas puede omitir la lectura del Capítulo 2.

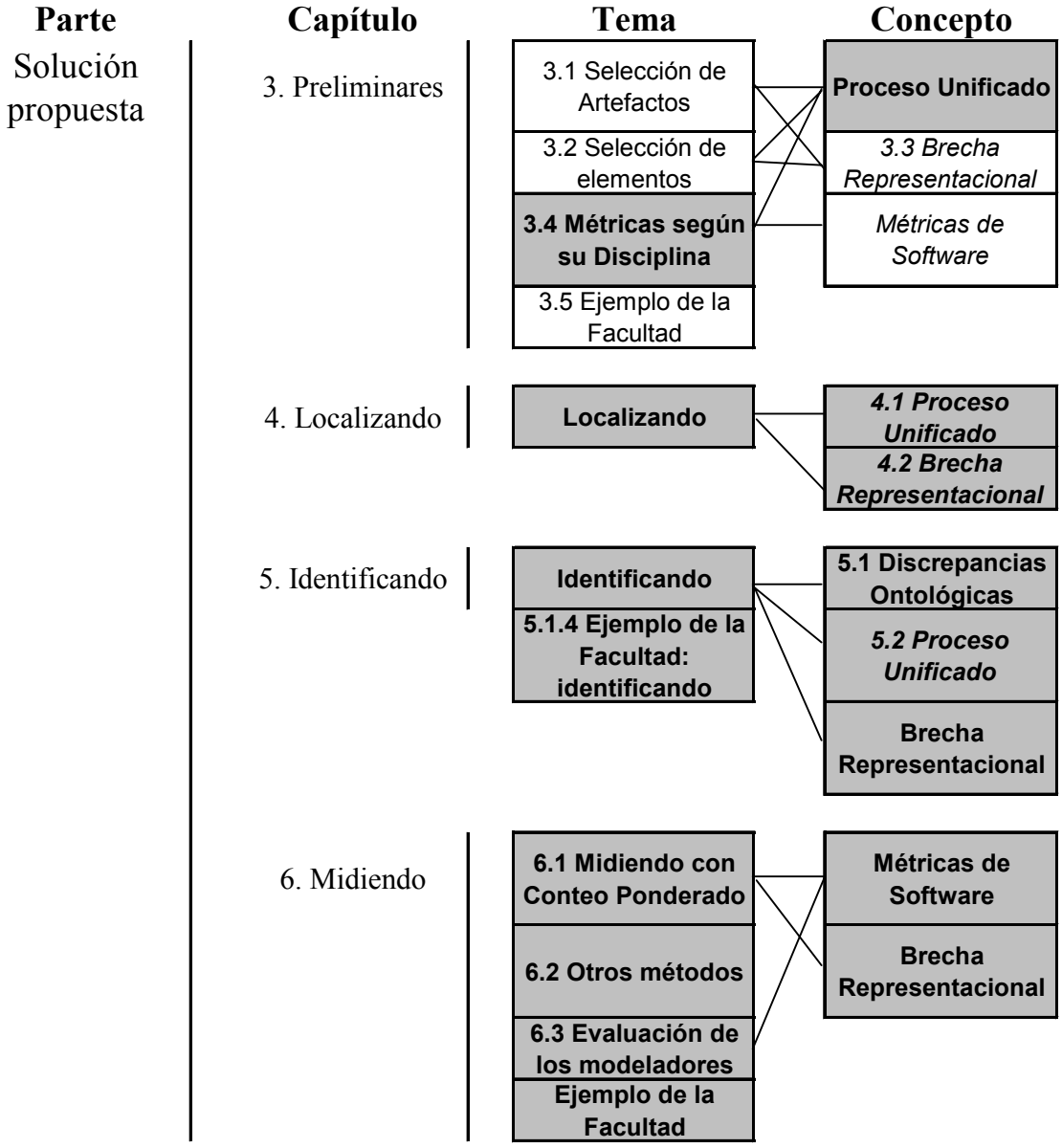


Figura 1.5. Solución propuesta

En el Capítulo 3, se muestran ciertos preparativos para explicar la solución propuesta. Para ello, se justifica el uso de ciertos Artefactos (i.e. Modelo de Dominio y Modelo de Diseño) y elementos (i.e. paquetes, clases y atributos); se introduce una nueva

clasificación de métricas por su dependencia a las Disciplinas que ayuda a justificar la originalidad de las métricas propuestas; se ofrece una explicación inicial sobre la medición de la Brecha Representacional y se detalla un ejemplo con el que se practicará en el Capítulo 4, Capítulo 5 y Capítulo 6. En estos capítulos es dónde se caracteriza a la Brecha Representacional. . Si bien el Capítulo 4, es relativamente corto, es la base para los demás. En el Capítulo 5, se profundiza en el conocimiento sobre las Discrepancias Ontológicas, mejorando el modelo de Bunge-Wand-Weber. El Capítulo 6 es más detallado pues además de explicar y justificar el procedimiento de la medición de la Brecha Representacional (i.e. Conteo Ponderado), se explican otros métodos menos apropiados (lectura optativa), la forma de evaluar la eficiencia de los modeladores, algunos beneficios del análisis de las Discrepancias y algunos comentarios críticos sobre las afirmaciones de [Larman01].

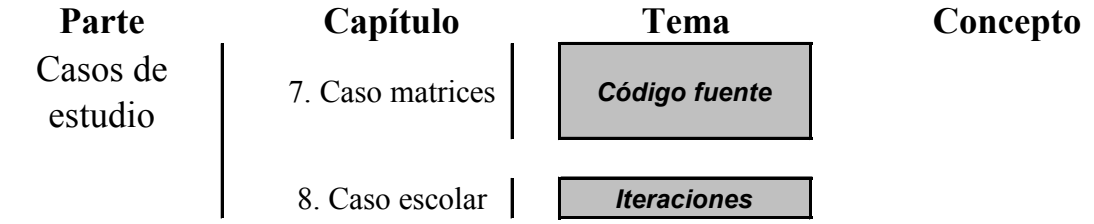


Figura 1.6. Casos de estudio

En el Capítulo 7, se detalla el caso de estudio de un proyecto académico de matrices. La idea es comparar el modelo conceptual de las matrices con el diseño extraído del código fuente. Mientras que en el Capítulo 8, se practica con el desarrollo de las Disciplinas tempranas en tres iteraciones. Para cada iteración se muestran los casos de uso, los diagramas de clases (i.e. Modelo de Dominio y Modelo de Diseño) y algunos comentarios sobre el desarrollo y el establecimiento de la correspondencia. En ambos casos de estudios se calcularon e interpretaron las métricas. También, se propusieron soluciones a diversos problemas que surgieron en la práctica.

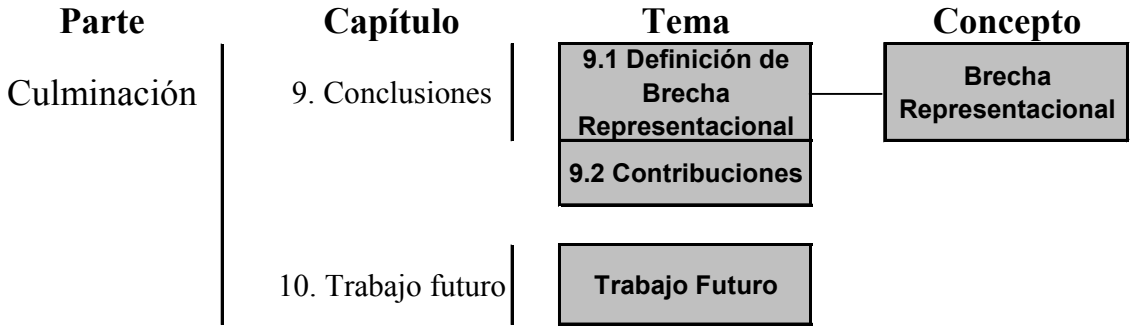


Figura 1.7. Culminación

En el Capítulo 9, se propone una definición formal de Brecha Representacional y se desglosan los objetivos cumplidos y las contribuciones de la investigación. Mientras que en el Capítulo 10, se enlista las líneas de investigación del trabajo futuro. Sección El apartado de anexos incluye apéndices, índices y bibliografía.

1.4.1 Convenciones

Por convención, las palabras y siglas en inglés, así como las referencias textuales a citas y elementos de los modelos, se encierran entre comillas. Se usa inglés en algunas partes como en el metamodelo de UML y el caso de estudio sobre matrices (el código fuente estaba en inglés).

- "Modelador" es una generalización de analista, diseñador, implementador (programador), etc. En la aplicación al Proceso Unificado, se empleará para el analista y el diseñador. En la generalización de la investigación, al analista se le llama "modelador de lo general" y al diseñador, "modelador de lo particular".
- "Disciplina" es el término empleado en el Proceso Unificado que en otras metodologías sería "etapa". Cuando se desea generalizar la idea o se hace referencia a un trabajo de diferente metodología, se usará "etapa" como en los temas: "Introducción" (página 16), "Costos de la identificación de defectos de software" (página 43) y "Métricas de software" (página 54). Estos términos son equivalentes.

1.5 Resumen visual de la tesis

1.5.1 Actores

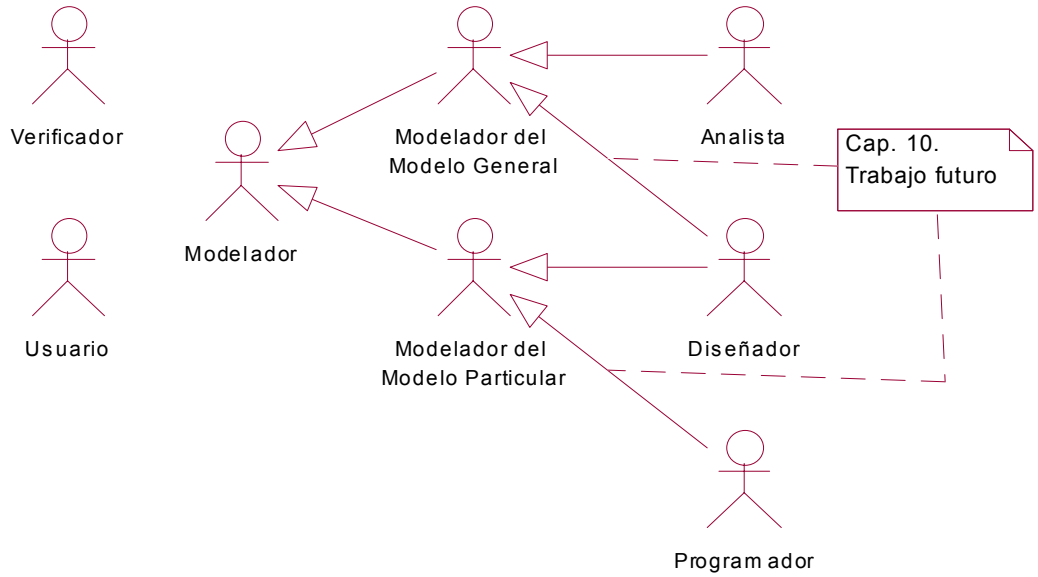


Figura 1.8. Actores de la Brecha Representacional

Los actores beneficiados directamente de la investigación son el verificador y el modelador de la Figura 1.8. Indirectamente, el usuario se beneficia con un producto (i.e. sistema) de mejor calidad. Ahora bien, cada modelador tiene un trabajo distinto: el analista define el dominio del problema, el diseñador especifica la solución en objetos de software y el programador implementa estas especificaciones.

Para caracterizar la correspondencia entre un modelo general (i.e. artefacto con de mayor nivel de abstracción) y un modelo particular (i.e. artefacto que refina al modelo general) se les denomina a sus autores, respectivamente, "modelador del modelo general" (o "modelador de lo general" para abreviar) y "modelador del modelo particular" (o "modelador de lo particular").

La presente investigación caracteriza el refinamiento del Modelo del Dominio en el Modelo del Diseño. En ambos, se trazan diagramas de clases con diferentes niveles de abstracción (e.g. el Modelo del Dominio es más abstracto que el Modelo del Diseño). Por lo que el modelador de lo general es el analista y el modelador de lo particular, el diseñador. En el "Trabajo Futuro" (página 192), se plantea la correspondencia entre el diseñador (i.e. modelador de lo general) y el programador (i.e. modelador de lo

particular). La elección del analista y el diseñador se basa en que la detección de defectos en etapas tempranas es menos costoso que la detección en etapas tardías[5].

1.5.2 Proceso Unificado

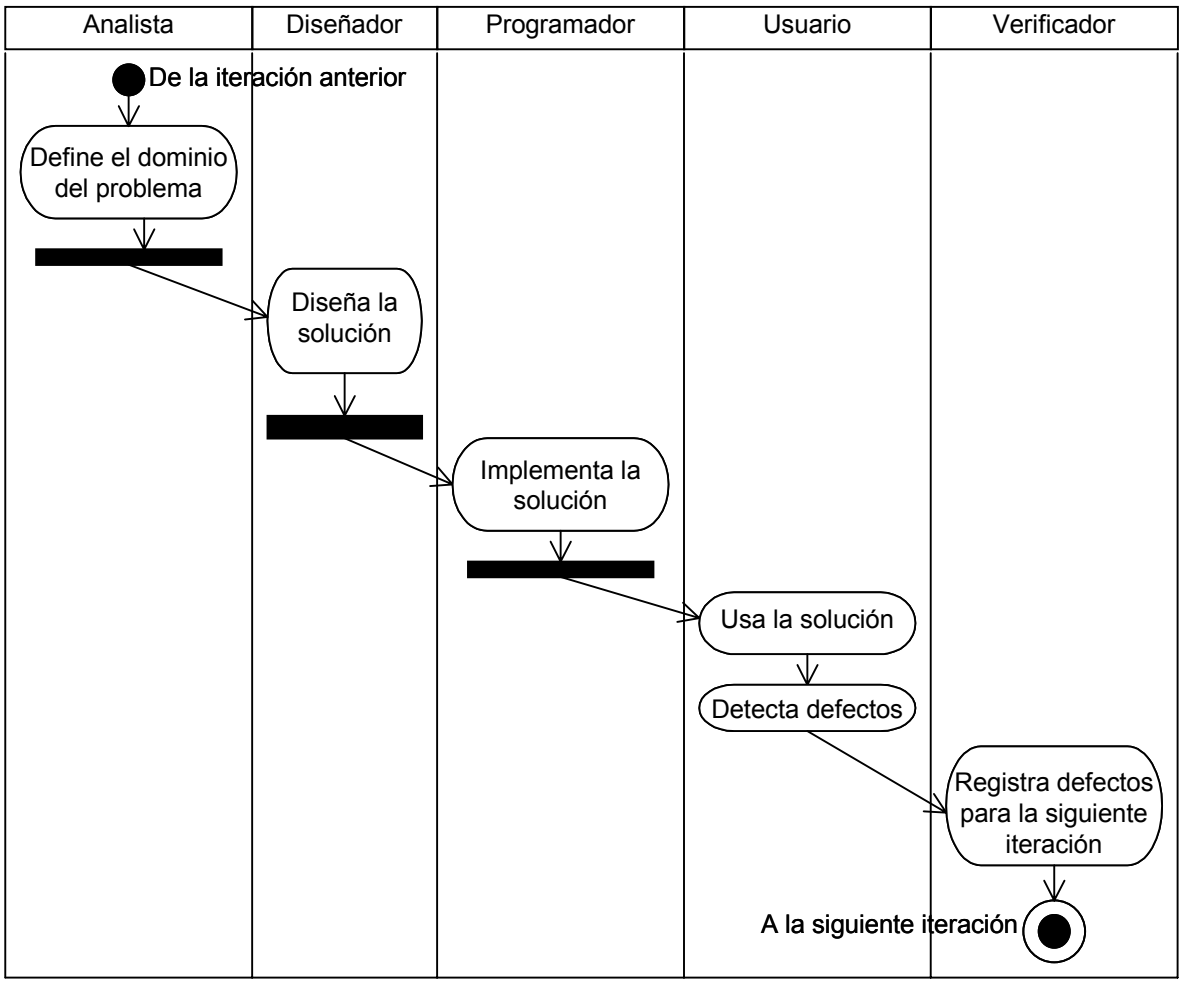


Figura 1.9 Iteración simplificada del Proceso Unificado (Diagrama de Actividades)

Las actividades básicas de cada iteración del Proceso Unificado son como un desarrollo en "cascada" pero en pequeña escala. El analista define el dominio del problema especificando el vocabulario del tema de interés (en el Modelo del Dominio). Luego, el diseñador usa este modelo[6] para producir una especificación (i.e. Modelo del Diseño) para la implementación del sistema, es decir, la descripción lógica de cómo un sistema trabajará[7]. De igual forma, el programador traduce estas especificaciones en un sistema (i.e. Modelo de Implementación) que se espera cumpla con los requerimientos del

[5] Ver "Costos de la identificación de defectos de software" en la página 43
[6] Propiamente se usan varios productos del analista que se incluyen en los Requerimientos: Modelo de Casos de Uso, Visión, Especificación Suplementaria y Glosario.
[7] Ver "Apéndice A. Glosario" en la página 194

usuario. En caso contrario, el verificador deberá encauzar los defectos detectados para ser corregidos en la siguiente iteración.

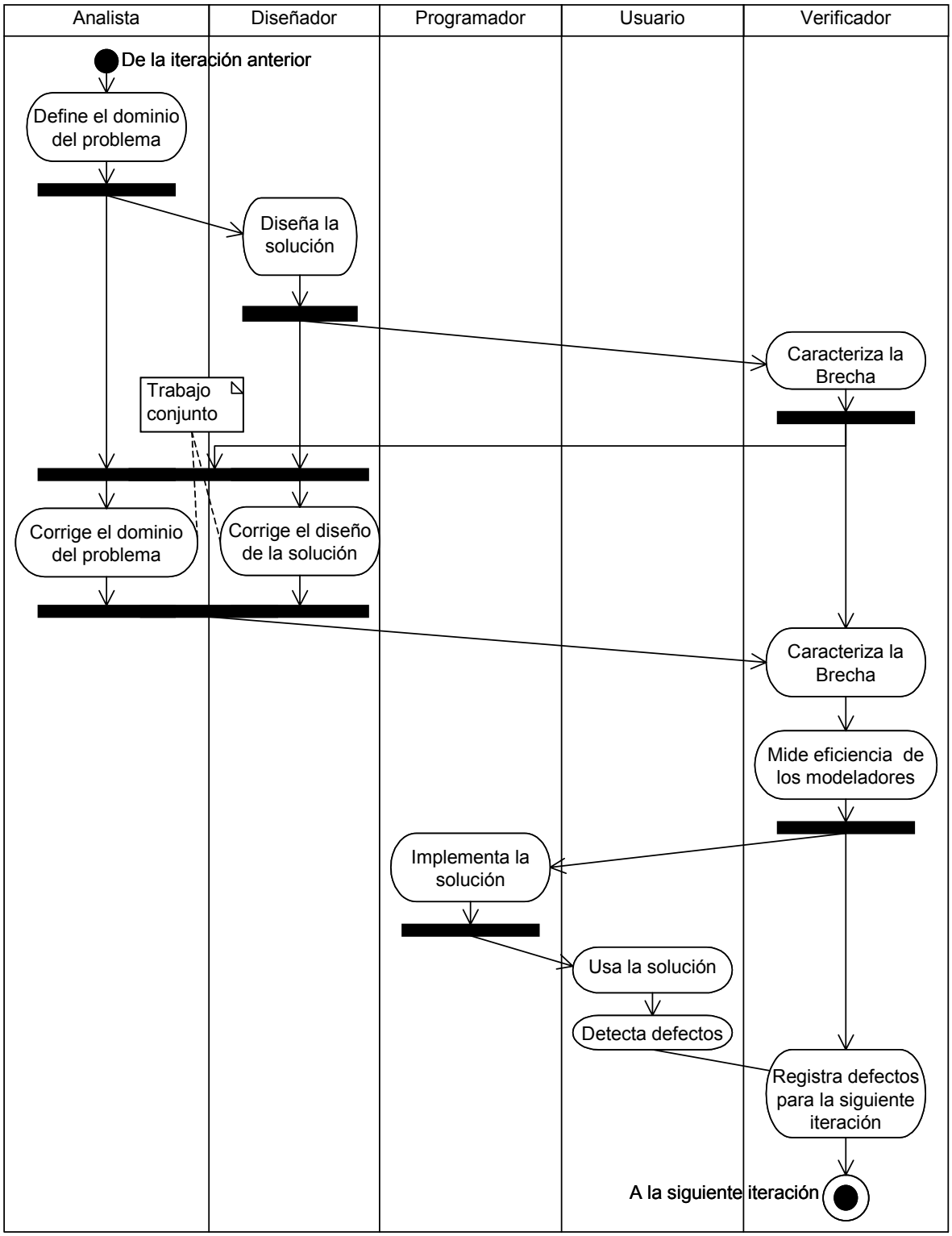


Figura 1.10. Iteración simplificada del Proceso Unificado con la propuesta (Diagrama de Actividades)

En la Figura 1.9 y Figura 1.10, se muestra una iteración simplificada con fines de comparación, pues no incluyen las actividades de otras Disciplinas como los Requerimientos, la Administración de Proyectos y el Ambiente. En la Disciplina de Pruebas se incluye las actividades de pruebas del verificador ("tester") o de éste en conjunto con el usuario. También, no se mostraron en estas figuras, el flujo de los objetos entre las actividades para hacerlos más legibles.

Si bien, los modeladores (e.g. analista y diseñador) detectan defectos no se expresa esta tarea en la Figura 1.10, ya que es una actividad interna (i.e. revisión). En cambio, el verificador y el usuario al detectar un defecto, se comunican con los modeladores para su corrección.

Al tener varios niveles de abstracción, se tiene una dependencia entre las Disciplinas (i.e. etapas) del desarrollo del software. Es decir, para implementar una solución, se necesita la elección de un diseño particular que se encuentra en un dominio definido del problema. Aunque estas actividades pueden realizarse de forma implícita (i.e. sin documentación), el Proceso Unificado requiere de la creación de varias especificaciones.

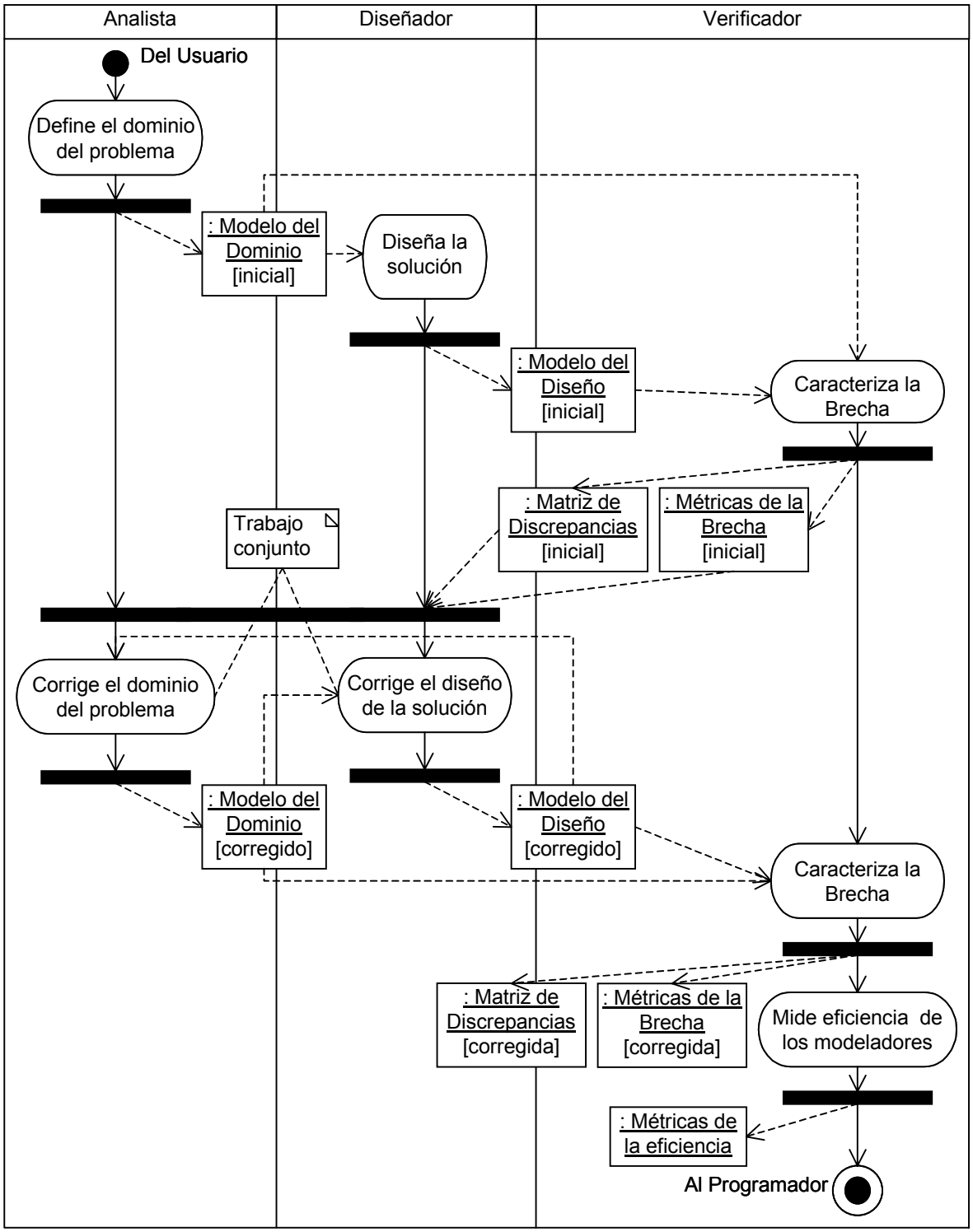


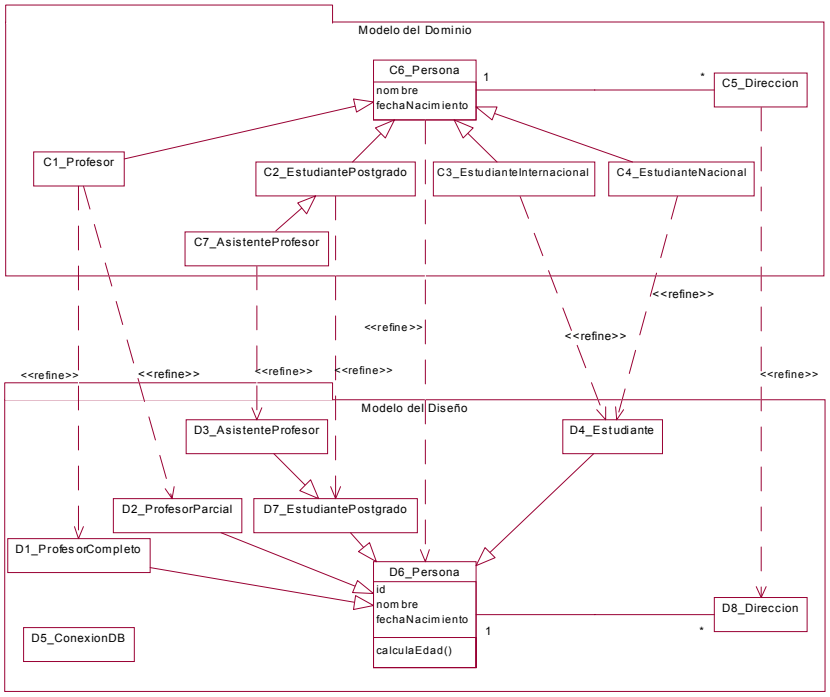
Figura 1.11. Iteración simplificada del Proceso Unificado con la propuesta y flujo de objetos (Diagrama de Actividades)

Al conjunto de diferencias entre las representaciones (i.e. modelos) del problema y de su solución, se le conoce como "Brecha Representacional"

Por todo esto, el análisis de las dependencias de refinamiento entre modelos (i.e. caracterización de la Brecha Representacional), ayuda a detectar defectos en las Disciplinas tempranas. Por ello, se propone caracterizar la Brecha entre el Modelo del Dominio y el Modelo del Diseño para que el analista y el diseñador, en conjunto, corrijan los defectos de sus modelos, como se muestra en la Figura 1.11. La matriz de Discrepancias permite localizar e identificar las diferencias entre los modelos que puedan originar defectos. Al comparar las métricas de la Brecha de las dos parejas de modelos (e.g. Modelo del Dominio y Modelo del Diseño), es decir, de la inicial y la corregida, se puede evaluar la eficiencia de los modeladores (i.e. analista y diseñador). De esta forma, se detectan defectos que previenen que el programador inserte otros defectos y sobretodo, que lleguen al usuario.

Ejemplos

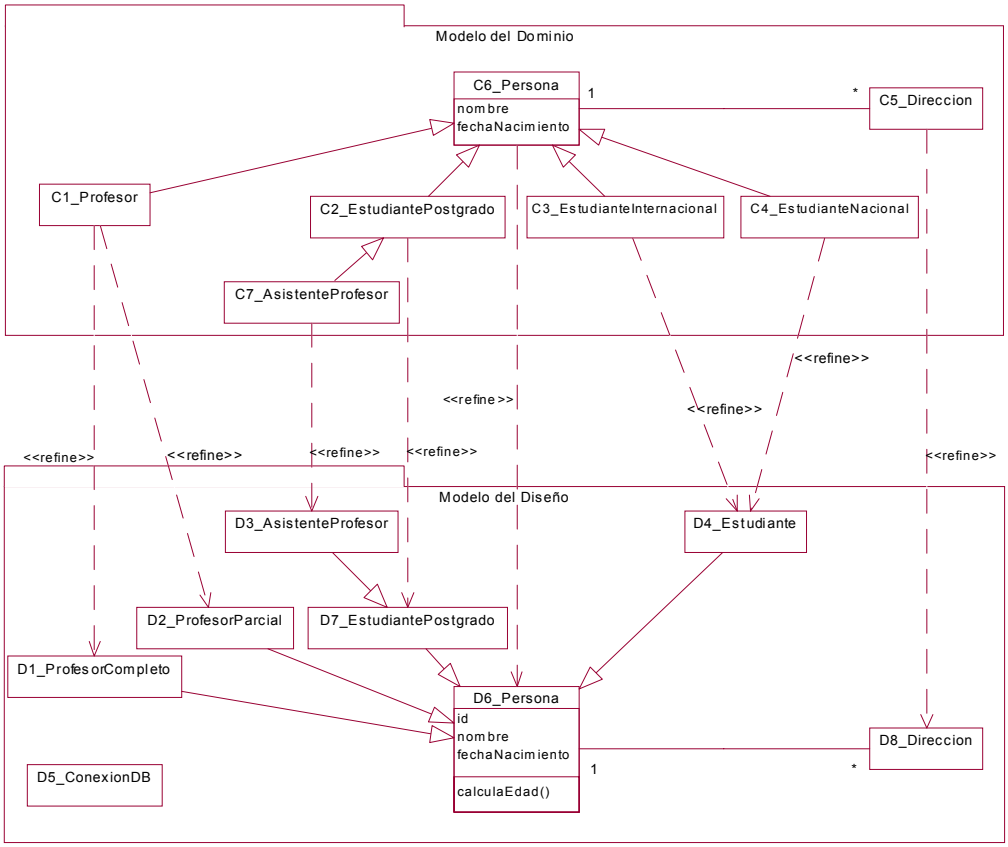
En el diagrama de actividades de la Figura 1.11 se pasan entre las actividades algunos objetos: Modelo del Dominio, Modelo del Diseño, matriz de Discrepancias, métricas de la Brecha y métricas de la eficiencia. Sólo como una guía visual, en la Figura 1.12, Figura 1.13 y Figura 1.14, se muestran estos objetos. Estas muestras de ejemplo se desarrollarán en la investigación.



C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	Def
c[1]	R3	R3	O2R3	-	-	-	-
c[2]	-	-	O2	-	-	-	-
c[3]	-	-	-	O2	-	-	-
c[4]	-	-	-	O2	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
Exc	-	-	-	-	E	-	

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	1	6	17%	11%	26%
Exceso	1	6	17%	6%	13%
Sobrecarga	2	6	33%	20%	46%
Redundancia	1	6	17%	5%	12%
Unión	6	36	17%	2%	4%
Modelo	**5**		**42%**	**43%**	**100%**

Figura 1.12. Par inicial de los modelos: diagramas de clases, matriz de identificación de Discrepancias y tabla de las métricas de la Brecha



C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]	d[8]	Def
c[1]	R2	R2	-	-	-	-	-	-	-
c[2]	-	-	-	-	-	-	1To1	-	-
c[3]	-	-	-	O2	-	-	-	-	-
c[4]	-	-	-	O2	-	-	-	-	-
c[5]	-	-	-	-	-	-	-	1To1	-
c[6]	-	-	-	-	-	1To1	-	-	-
c[7]	-	-	1To1	-	-	-	-	-	-
Exc	-	-	-	-	E	-	-	-	

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	0	7	0%	0%	0%
Exceso	1	8	13%	4%	25%
Sobrecarga	1	7	13%	8%	45%
Redundancia	1	8	14%	4%	26%
Unión	4	56	7%	1%	4%
Modelo	**3**		**20%**	**17%**	**100%**

Figura 1.13. Par corregido de los modelos: diagramas de clases, matriz de identificación de Discrepancias y tabla de las métricas de la Brecha

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	0%	117%	0%	0%	0%
Exceso	100%	133%	75%	75%	195%
Sobrecarga	50%	117%	38%	38%	98%
Redundancia	100%	133%	86%	86%	223%
Unión	67%	156%	43%	43%	111%
Modelo	**60%**		**47%**	**38%**	**100%**

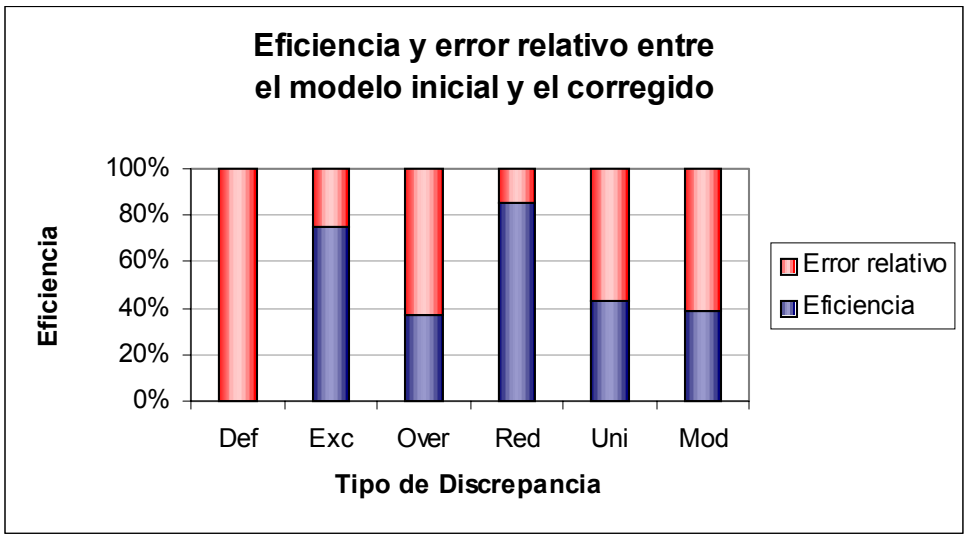


Figura 1.14. Tabla de las métricas de la eficiencia de los modeladores y gráfica de la eficiencia (según el peso de la Discrepancia)

1.5.3 Caracterización de la Brecha Representacional

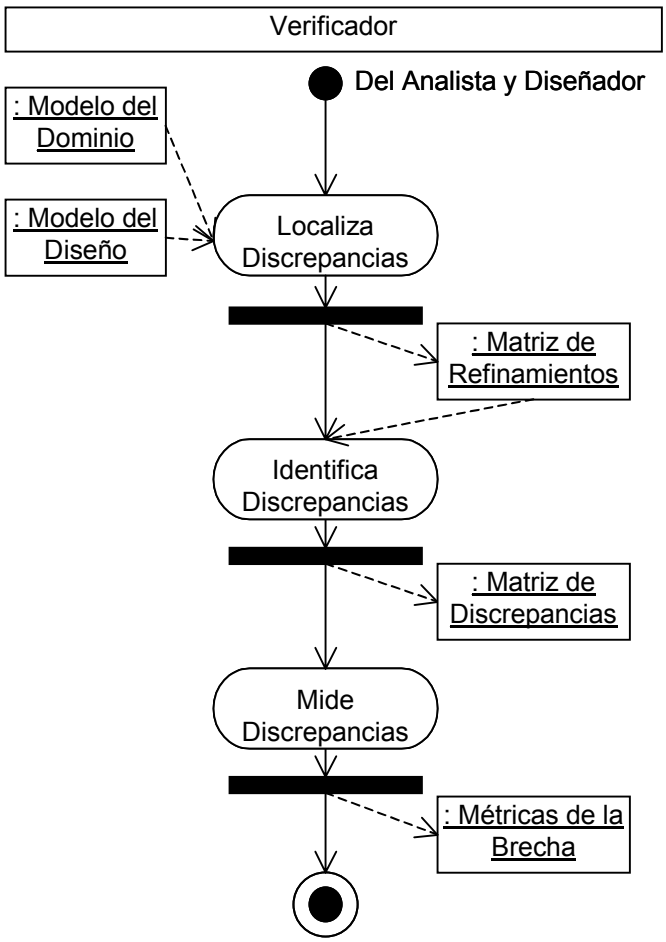


Figura 1.15. Caracterización de la Brecha Representacional mediante las Discrepancias Ontológicas (Diagrama de Actividades)

El análisis de la Brecha Representacional se caracteriza por medio de la localización (Figura 1.16), la identificación (Figura 1.17) y la medición (Figura 1.20) de la Brecha mediante las Discrepancias Ontológicas, como se muestra en la Figura 1.15. Para calcular las métricas de la Brecha, se requiere de la identificación de las Discrepancias mediante la matriz de Discrepancias, que a su vez necesita la matriz de refinamientos, obtenida en la localización de las Discrepancias entre el Modelo del Dominio y el Modelo del Diseño.

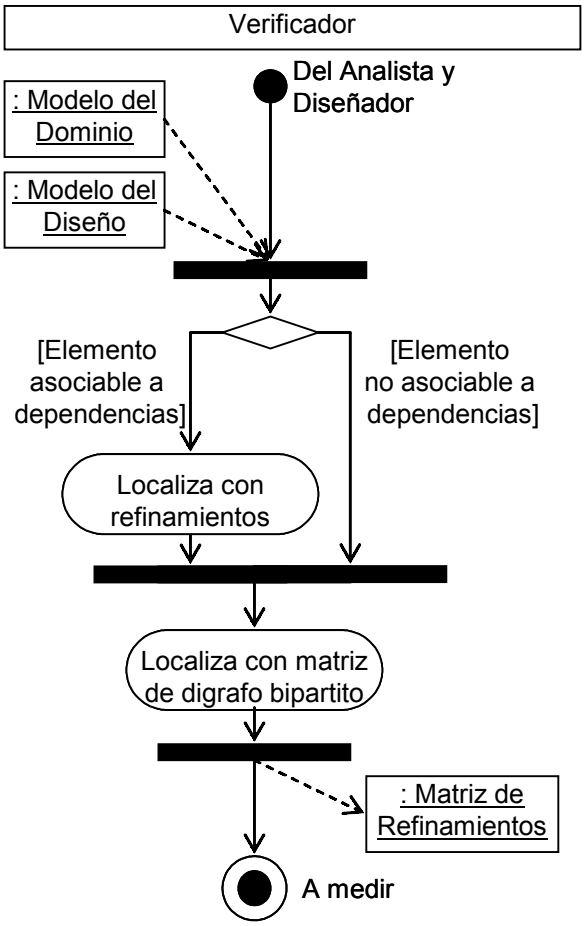


Figura 1.16. Métodos para localizar Discrepancias (Diagrama de Actividades)

Cuando el refinamiento entre elementos de modelado (e.g. clases) se traza con dependencias de UML (estereotipadas con "refine"), se le denomina como localización con refinamientos. Cuando no se pueden trazar (i.e. asociar) dependencias entre los elementos (e.g. atributos), se representan mediante una matriz de refinamiento que puede ser empleada a todos los elementos con ciertas variaciones[8]. También se puede obtener esta matriz mediante los modelos con elementos asociables a dependencias, como se muestra en la Figura 1.16. Esta matriz es la representación de un digrafo bipartito[9] donde los vértices representan a los elementos y los arcos a los refinamientos.

[8] Ver "Selección de los elementos" en la página 62
[9] Ver "La Brecha Representacional como un grafo" en la página 83

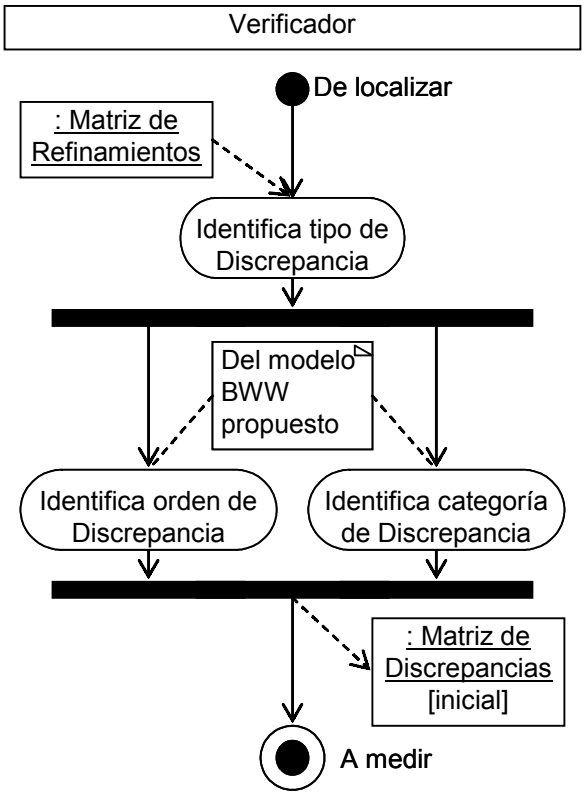


Figura 1.17. Identificación del tipo, orden y categoría de las Discrepancias (Diagrama de Actividades)

Mediante la matriz de refinamientos se identifica el tipo de Discrepancia (Figura 1.20) con el que se puede identificar su orden y categoría (Figura 1.19). Esta información se condensa en una matriz de Discrepancias. Cabe mencionar que el orden y la categoría, son parte de la versión mejorada del modelo Bunge-Wand-Weber, que se propondrá.

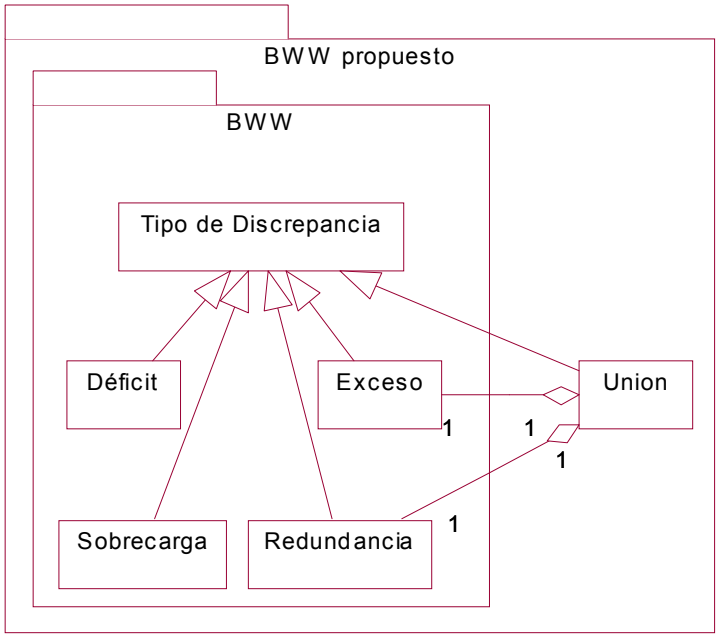


Figura 1.18. Tipos de Discrepancias para la identificación de las Discrepancias

En la Figura 1.18, se tienen los tipos de Discrepancias Ontológicas del modelo de Bunge-Wand-Weber: Déficit, Exceso, Sobrecarga y Redundancia. En el modelo BWW propuesto, se detecta el fenómeno de la Unión que se compone de la fusión de un Exceso y una Redundancia.

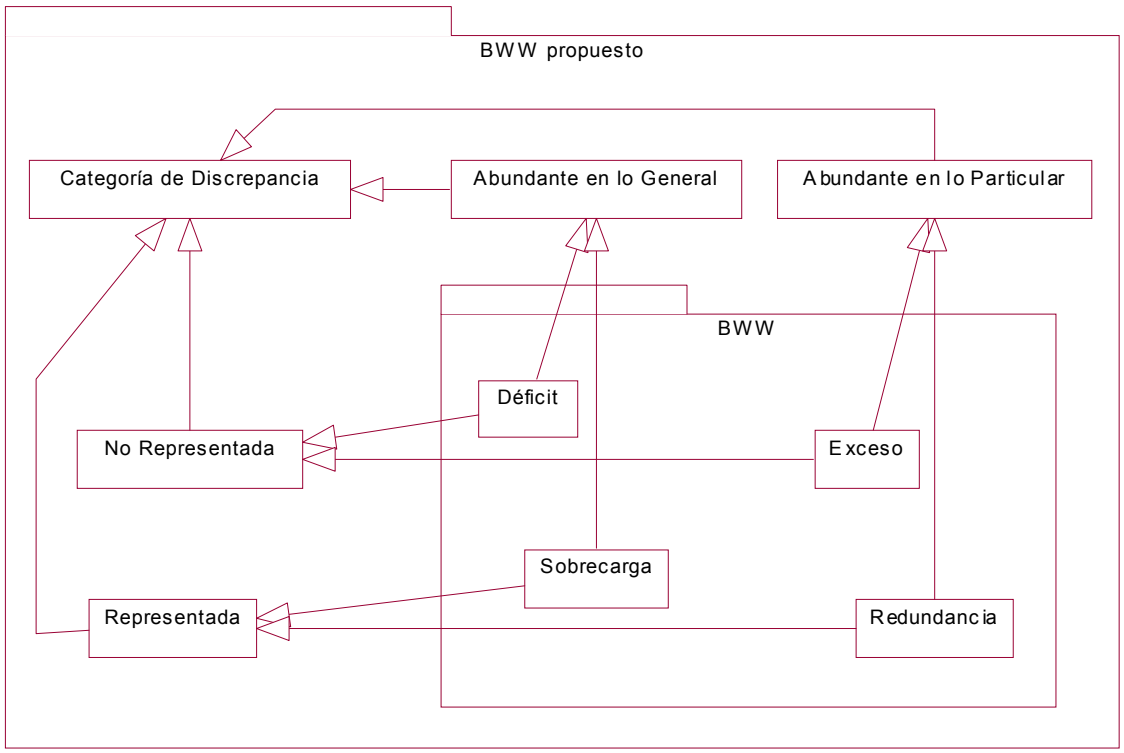


Figura 1.19. Categorías de las Discrepancias para la la identificación de las Discrepancias

Como parte del modelo BWW propuesto, en la Figura 1.19 se muestran las categorías de los tipos de Discrepancias: Abundante en lo General, Abundante en lo Particular, Representada y No Representada. Cabe notar que en estas categorías se refleja la complementariedad de los tipos de Discrepancias. En la Tabla 5.3, página 95,se muestra un resumen de los tipos y categorías de las Discrepancias.

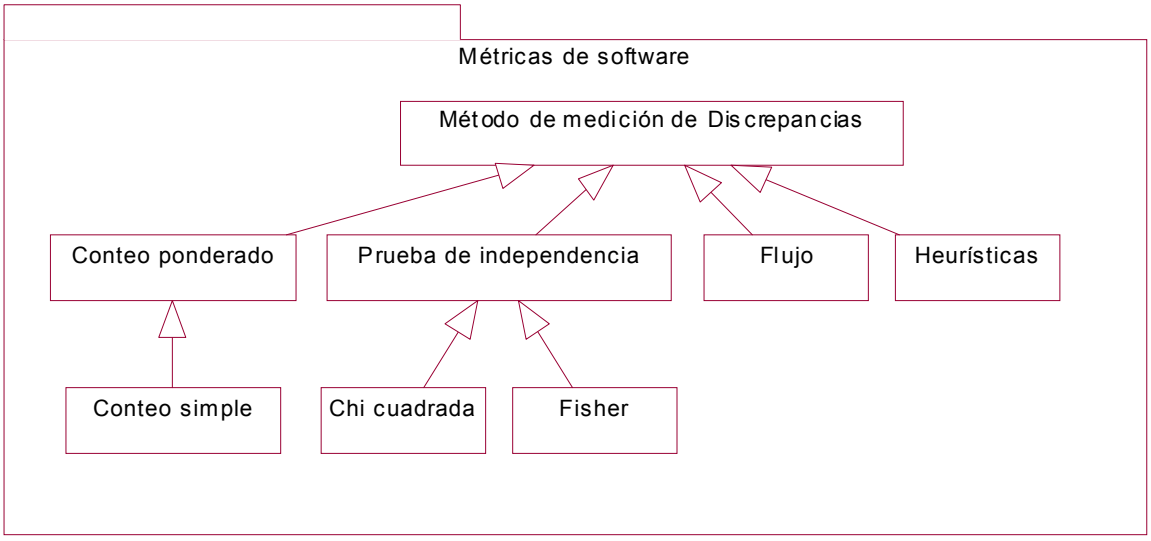


Figura 1.20. Métodos de medición de las Discrepancias probados

En la Figura 1.20, se muestran los métodos de medición de las Discrepancias (así como algunas variaciones) con los que se experimentó para evaluar cuantitativamente la Brecha: Conteo Ponderado (Conteo Simple), Prueba de Independencia (Chi Cuadrada, Prueba Exacta de Fisher), Flujo y Heurísticas[10]. El método seleccionado y más explicado es el Conteo Ponderado[11].

[10] Ver "Otros métodos para medir la Brecha Representacional" en la página 119
[11] Ver "Método de medición seleccionado: Conteo Ponderado" en la página 104

Capítulo 2. Estado del arte

2.1 Costos de la identificación de defectos de software

La justificación de encontrar los defectos lo más temprano posible se basa en algunas conclusiones de [Humphrey95]:

> "La revisión e inspección de programas, inmediatamente después de que se producen, minimiza el número de defectos en el producto en cualquier etapa. Al hacer esto, también se minimiza la cantidad de retrabajo y los costos involucrados. Además, habrá más propensión a reducir los costos de encontrar los efectos en primer lugar. […] Aunque no se puede asegurar que una reparación implicará a todo elemento de costo, [se puede decir que] mientras más dure un defecto en el producto, posiblemente, se tendrán más elementos involucrados."

De forma similar, [Opdahl02] cita a los trabajos de Davis y Pohl para argumentar que las etapas tempranas son críticas para el éxito y la eficiencia del costo del desarrollo de los sistemas de información.

A continuación se muestran algunos datos sobre defectos en el software [Humphrey95]

- IBM. Regla intuitiva no publicada de IBM para los costos relativos para identificar defectos de software: durante diseño, 1.5; antes de la codificación, 1; durante la codificación, 1.5; antes de las pruebas, 10; durante las pruebas, 60; en uso, 100.
- TRW. Tiempo relativo para identificar los defectos: durante los requerimientos, 1; durante el diseño, de 3 a 6; durante la codificación, 10; en pruebas de desarrollo, de 15 a 40; en pruebas de aceptación, de 30 a 70; durante la operación, 40 a 1000.
- IBM. Tiempo relativo para identificar los defectos: durante revisiones de diseño, 1; durante inspecciones de código, 20; durante pruebas de máquina, 82.
- JPL. Costo (en dólares USD) promedio por defecto: $90 a $120 en inspecciones y $10,000 en pruebas.
- Freedman y Weinberg. Reportaron que los proyectos que emplean revisiones e inspecciones tenían reducciones de diez veces en el número de defectos encontrados en prueba. Así como una reducción del 50% al 80% en costos de prueba, incluyendo los costos de las revisiones y las inspecciones.

Tabla 2.1. Comparación sobre identificación de defectos de software por referencia

Organización	IBM	JPL		IBM	TRW	
Unidades	**Costos relativos**	**Dólares USD /defecto**		**Tiempos relativos**	**Tiempos relativos**	
Etapa común	**Prom.**	**Mín.**	**Máx.**	**Prom.**	**Prom. Mín.**	**Máx.**
Requerimientos					1	
Diseño	1.5				3	6
Revisiones de diseño	1			1		
Codificación	1.5				10	
Inspecciones de código	10	90	120	20		
Pruebas	60		10000	82	15	40
Pruebas de aceptación					30	70
Operación	100				40	1000

Tabla 2.2. Proporciones entre etapas de costos y tiempos relacionados con los defectos

Organización	IBM	JPL		IBM	TRW	
Unidades	**Costos relativos**	**Dólares USD /defecto**		**Tiempos relativos**	**Tiempos relativos**	
Etapa	**Prom.**	**Mín.**	**Máx.**	**Prom.**	**Mín.**	**Prom. Máx.**
Etapa 1	0.67	111.11	83.33	20.00	3.00	6.00
Etapa 2	1.50			4.10	3.33	1.67
Etapa 3	6.67				1.50	4.00
Etapa 4	6.00				2.00	1.75
Etapa 5	1.67				1.33	14.29
Prom.	**3.30**	**97.22**		**12.05**	**3.89**	
Mín.	0.67	83.33		4.10	1.33	
Máx.	6.67	111.11		20.00	14.29	

Para facilitar su comprensión y comparación, se han reorganizado algunos de estos datos en la Tabla 2.1, para mostrar que mientras más tarde se identifica un defecto es más costoso. Mientras que en la Tabla 2.2, se efectuaron algunos cálculos para obtener una (vaga) regla acerca de la identificación de los defectos.

En la Tabla 2.2, se muestran las proporciones de recursos relacionados con los defectos. Por ejemplo, los tiempos relativos en IBM de dejar pasar un defecto, en promedio, se cuadruplican. Estos tiempos van desde 133% (1.33 sombreado en la Tabla 2.2) veces hasta más de 14 veces (14.29 sombreado en la Tabla 2.2). Cabe mencionar que la alta proporción en JPL y en los tiempos relativos de IBM, se puede deber a las etapas tardías. El mínimo de 67% (0.67 sombreado en la Tabla 2.2) de los costos relativos de IBM, parece raro, pero se refiere a una revisión que se dedica a detectar los defectos, no a producir artefactos.

De forma general, se puede decir que el costo y el tiempo (en promedio) de dejar pasar un defecto a la siguiente etapa (i.e. Disciplina); al menos se triplican. En definitiva, mientras más pronto se logre detectar y corregir los defectos, se aumentan las oportunidades de

entregar un sistema en la fecha establecida y dentro del presupuesto. Sin embargo, lograrlo no es un asunto trivial pues requiere de métodos, herramientas y orden.

2.2 Proceso Unificado (UP)

2.2.1 Selección del Proceso Unificado

Se seleccionó la metodología del Proceso Unificado ("UP") debido a que representa un esfuerzo muy difundido, maduro y estandarizado para el enfoque Orientado a Objetos. Debido a que se pretende medir los productos en las Disciplinas tempranas, y que UML es un lenguaje de modelado de propósito general que ha llegado a ser un estándar de facto [Opdahl02], se optó por el Proceso Unificado. Además existe una amplia disponibilidad de recursos (literatura, herramientas, etc.).

Otra razón importante fue la existencia de artículos que avalan la teoría detrás de la representación del mundo real [Opdahl02], su correspondencia con UML y la explicación de una problemática ontológica (Discrepancias Ontológicas).

2.2.2 Distinción entre UML y el Proceso Unificado: perspectivas múltiples

Para establecer la diferencia básica entre el modelo de UML y el Proceso Unificado, se citará a [Larman01]:

> "UML sólo describe tipos ordinarios de diagrama, como diagramas de clases y diagramas de secuencia. No impone un método o perspectiva de modelado en estos. En vez de ello, un proceso (como UP) aplica UML de forma ordinaria en el contexto de los modelos definidos en la metodología".

Esto es, la misma notación se debería de usar para tres perspectivas y tipos de modelos:

- **Perspectiva Conceptual (o Esencial)**. Los diagramas son interpretados como la descripción de las cosas del mundo real o del Dominio de interés.
- **Perspectiva de Especificación**. Estos mismos diagramas son interpretados como la descripción de las abstracciones de software, pero no comprometidas a una implementación particular.
- **Perspectiva de Implementación**. Estos mismos diagramas son interpretados como la descripción de la implementación del software de una tecnología y lenguaje en particular (como Java o C++).

2.2.3 Fases del Proceso Unificado

Las Fases del Ciclo de Desarrollo del Proceso Unificado son:

- Iniciación
- Elaboración
- Construcción
- Transición

Cada Fase tiene una o varias iteraciones. Por lo general, la Fase de Iniciación tiene sólo una iteración, la de Transición, dos y las demás varían.

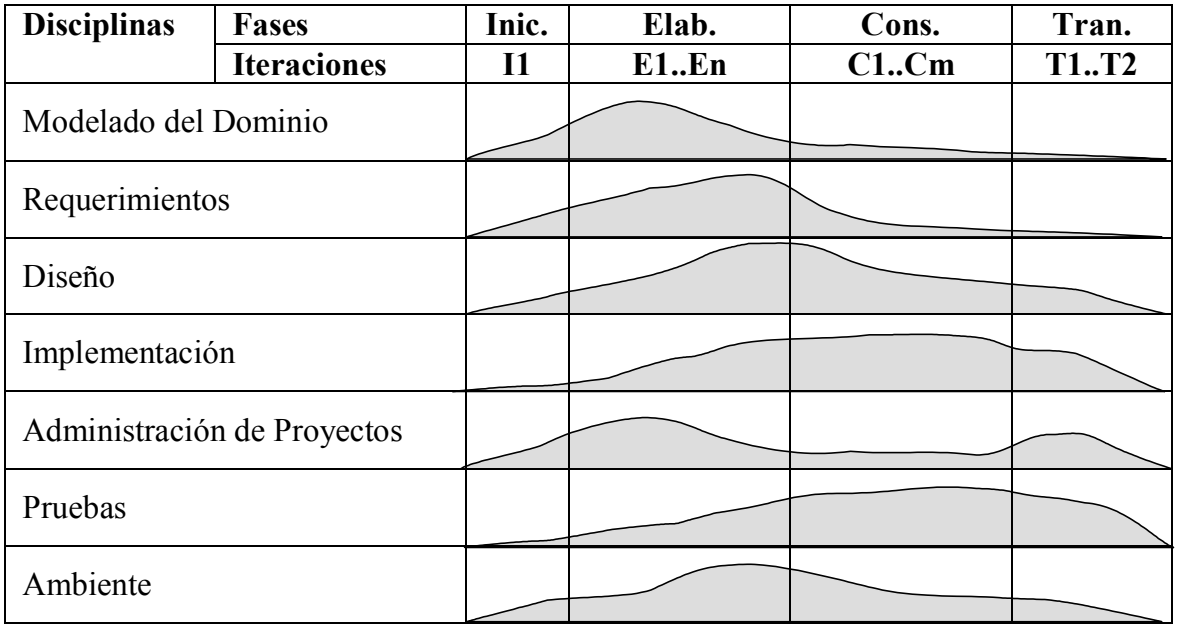


Figura 2.1. Esfuerzo relativo a través del tiempo de las Disciplinas del Proceso Unificado

Tabla 2.3. Disciplinas, Artefactos y Fases del Proceso Unificado ("C" comienza; "R" refinación)

Disciplinas	Artefactos	Fases	Inic.	Elab.	Cons.	Tran.
		Iteraciones	I1	E1..En	C1..Cm	T1..T2
Modelado del Dominio	***Modelo del Dominio***			***C***		
Requerimientos	Modelo de Casos de Uso		C	R		
	Visión		C	R		
	Especificación Suplementaria		C	R		
	Glosario		C	R		
Diseño	***Modelo del Diseño***			***C***	***R***	
	Documento de la Arquitectura SW			C		
	Modelo de Datos			C	R	
Implementación	Modelo de la Implementación			C	R	R
Administración de Proyectos	Plan de Desarrollo SW		C	R	R	R
Pruebas	Modelo de Pruebas			C	R	
Ambiente	Caso del Desarrollo		C	R		

* En sombreado se señalan las áreas donde se propondrán métricas.

En la Figura 2.1, se muestra el esfuerzo relativo dependiendo de la iteración y la Disciplina. Con estas curvas se muestra el enfoque iterativo: en cada iteración se pasa por todas las Disciplinas, aunque en ciertas iteraciones se dedica mayor esfuerzo a algunas Disciplinas. Esto es, el esfuerzo se adapta dependiendo de las necesidades y

requerimientos de la iteración. Aunque una Disciplina usualmente comienza en ciertas Fases y se refina en algunas más. Esto se aprecia en la Tabla 2.3. Por ejemplo, uno de los artefactos de la Disciplina de Diseño, el Modelo del Diseño, comienza en la Fase de Elaboración y se refina en la de Construcción, pero como se vio en la Figura 2.1, puede estar presente en las demás Fases.

2.2.4 Diagramas de UML

Tabla 2.4. Áreas, vistas y diagramas de UML

Área mayor	**Vista**	**Diagrama de**
Estructural	***Estática***	***Clases***
	Casos de uso	Casos de uso
	Implementación	Componentes
	Despliegue	Despliegue
Dinámico	Máquinas de estado	Estados
	Actividad	Actividad
	Interacción	Secuencia
		Colaboración
Administración del modelo	Administración del modelo	Clases
Extensibilidad	Todas	Todos

En la Tabla 2.4, se muestra (en sombreado) la posición de los diagramas de clase con respecto a los demás diagramas existentes en UML. Los diagramas de clases representan la estructura estática de un sistema. En general, los diagramas de clases emplean los elementos ("conceptos de UML"): clase, asociación, generalización, dependencia, realización, interfaz. [Booch99]

2.2.5 Categorías de las relaciones y tipos de dependencias

Más adelante (ver "Localizando las Discrepancias" en la página 81) se emplearán las dependendecias de refinamiento. Este tipo particular de dependencia se sitúa en una categoría de relaciones. A continuación se muestran las categorías de las relaciones de UML [Rumbaugh99]:

- **Abstracción**
- Asociación
- Atadura
- Flujo
- Generalización
- Inclusión
- Metarelación
- Permiso
- Uso

Dentro de la categoría de abstracción (en negritas) existen cuatro variedades: derivación, realización, refinamiento y rastreo. Enseguida, se muestran todos los tipos de dependencias estándar de UML resaltando (en negritas) el tipo que se usa en el presente trabajo.

Tabla 2.5. Tipos de dependencias [Rumbaugh99]

Estereotipo (palabra clave)	**Dependencia**
access	Acceso
bind	Atadura
call	Llamada
derive	Derivación
friend	Amigo
import	Importación
instantiate	Instanciación
parameter	Parámetro
realize	Realización
refine	**Refinamiento**
send	Envío
trace	Rastreo
use	Uso

2.3 Brecha Representacional

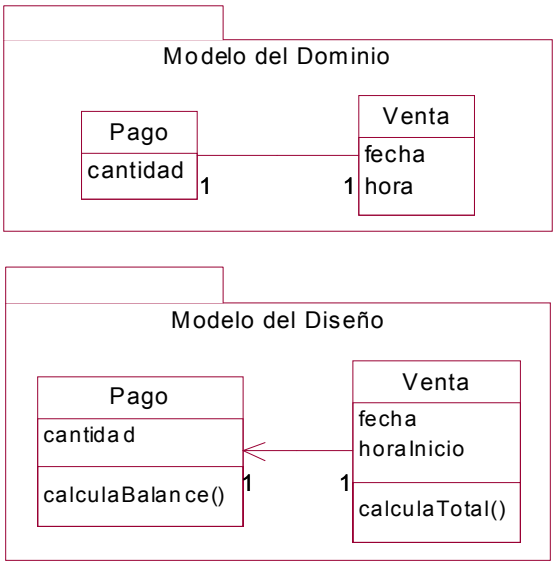


Figura 2.2. Ejemplo de la Brecha Representacional [Larman01]

De forma general, [Larman01] percibe una Brecha Representacional entre la conceptualización del problema (Modelo de Dominio) y su solución (Diseño). Una Brecha Representacional ("Representational Gap" o "Semantic Gap") es citada como: "El intervalo entre el modelo mental del dominio que pensamos y su representación en software". Este fenómeno se puede apreciar en la Figura 2.2. En el Modelo del Dominio de ventas, se tiene que para cada "Venta" se relaciona con un "Pago". Si bien, no hay una relación uno a uno entre el Modelo del Dominio y el Diseño, sería deseable que estos modelos fueran similares, es decir, que esta Brecha fuera estrecha . Cabe notar que el Modelo del Diseño tiene métodos y que la relación difiere por la navegabilidad (i.e. la flecha de "Venta" a "Pago").

De igual forma se señala que una Brecha Representacional reducida es útil. Mientras la correspondencia se acerque a una relación uno a uno, entre el vocabulario del dominio y nuestro vocabulario de software, así como su agrupación en unidades más manejables, reducirá esta Brecha. La mayoría de los ingenieros de software saben que esto es cierto, "aunque es difícil de cuantificar" [Larman01]. Este trabajo propone una caracterización de la Brecha Representacional (que incluye la cuantificación).

[Larman01] especifica que:

> "El Modelo del Dominio provee un diccionario visual del vocabulario y de los conceptos del dominio, del que se obtiene inspiración para el nombrado de algunas cosas en el diseño del software"

Esto se relaciona con la Brecha Representacional, pues es el intervalo entre nuestro modelo mental del dominio y su representación en el software. El hecho de separar el problema (conceptos en el Modelo del Dominio) de la solución (clases de software en el Modelo del Diseño) reduce (intuitivamente) la Brecha Representacional, y se considera "una de las grandes ideas en la tecnología de [orientada a] objetos" [Larman01].

2.3.1 Niveles de abstracción

Tabla 2.6. Niveles de abstracción de una clase

Clase	Entidad	Énfasis	Orientación a objetos
Clase Conceptual		Concepto del dominio	OOA
Clase de Diseño	Persona nombre fechaNacimiento daNombre() calculaEdad()	Visualización del concepto del dominio	OOD

Clase de Implementación	`class Persona {`
	`  String nombre;`
	`  Date fechaNacimiento;`
	`  Public String daNombre()`
	`  {…}`
	`  public int calculaEdad()`
	`  {…}`
	`}`
		Representación en un lenguaje orientado a objetos	OOP

En el Proceso Unificado, ciertos elementos tienen varios niveles de abstracción. En la Tabla 2.6, se ejemplifican los tres niveles de abstracción de las clases. En el análisis orientado a objetos ("OOA" por sus siglas en inglés) hay un énfasis para encontrar y describir los objetos (i.e. conceptos) del dominio del problema. Mientras que la intención del diseño orientado a objetos ("OOD"), es definir los objetos de software y cómo colaboran entre ellos para cumplir los requerimientos. Finalmente, en la implementación, los objetos de diseño se programan en algún lenguaje orientado a objetos ("OOP").

2.4 Discrepancias Ontológicas

El modelo Bunge-Wand-Weber (BWW) ha sido aplicado al análisis y evaluación de métodos de diseño de sistemas de información, diagramas de flujo de datos, diagramas entidad-relación, una , nueve lenguajes soportados por la herramienta "Upper CASE-toolset Excelerator", cuatro lenguajes soportados por las herramientas ARIS para el modelado de negocios y Al igual que Wand y Weber, la base teorética elegida fue la ontología de Bunge empleada también en [Chidamber94]

[Opdahl02] se basa en las Discrepancias para establecer una correspondencia de los conceptos de la Ontología de BWW con las construcciones de UML.

Por lo anteriormente mencionado, se eligió el modelo BWW. Además, se cuenta con artículos que avalan la teoría detrás de la representación del mundo real [Opdahl02], su correspondencia con UML y la explicación de una problemática ontológica (Discrepancias Ontológicas).

Un Concepto Ontológico es categoría de ideas o cosas del mundo real que se supone existen en alguna área de interés. Mientras que una Construcción del Modelado es la especificación formal explícita que representa a un Concepto Ontológico [Opdahl02]. Coloquialmente, el Concepto Ontológico es una abstracción del usuario y la Construcción del Modelado es la especificación empleada por el desarrollador (e.g. analista) para modelar la solución.

Wand y Weber identifica cuatro Discrepancias Ontológicas (en adelante llamadas sólo Discrepancias) que pueden deteriorar la claridad ontológica de las construcciones y del lenguaje [Opdahl02]. A continuación se describen las Discrepancias del modelo BWW[12]:

- **Sobrecarga de Construcciones** ("Construct Overload")
  - Cuando una construcción representa a varios conceptos.
  - Usualmente, es una situación problemática.
- **Redundancia de Construcciones** ("Construct Redundancy")
  - Cuando varias construcciones representan a un mismo concepto.
  - No es necesariamente una situación problemática, mientras que el traslape de las Construcciones del Modelado, representen subtipos disyuntivos del Concepto Ontológico. De hecho, se menciona que puede ser de ayuda para clasificar un Concepto Ontológico en varios subtipos. Sin embargo, puede resultar en un Déficit, si los subtipos en conjunto, no cubren completamente al Concepto Ontológico.
    También se observa que existen dos tipos de redundancia en UML:
  - **De Subtipo**
    - Dos o más Construcciones de UML pueden representar al mismo Concepto BWW en el dominio del problema porque son subtipos de una construcción más general (y no redundante).

[12] La apreciación gráfica de las Discrepancias se muestra en la Figura 5.2, dentro del tema "Mejoras propuestas al modelo BWW", página 87.

        - No necesariamente es un problema.
        - Los subtipos pueden ser útiles porque:
            - Proveen a los modeladores y a los usuarios de los modelos de vocabularios más precisos y orientados al problema.
            - Ofrecen definiciones más precisas de atributos y relaciones en el metamodelo de UML.
    - **Genuina**
        - Dos o más Construcciones de UML pueden ser genuinamente redundantes porque representan al mismo Concepto BWW en el dominio del problema.
        - Indica debilidad en la definición y en la estructura de UML que debería de ser mejorada en versiones futuras de UML:
            - Cada grupo con construcciones que sean semánticamente semejantes deberían de ser colapsadas en una única Construcción UML.
            - Cada grupo con construcciones que sean semánticamente distintas deberían de separarse (especializarse) en una metaclase más general en el modelo de UML.
- **Exceso de Construcciones** ("Construct Excess").
    - Cuando una construcción no representa a un concepto. Es problemática, si la Construcción del Modelado claramente pretende (al menos en parte) representar al fenómeno o algún aspecto del problema del dominio.
    - Hay dos tipos generales de excesos:
        - **No orientado al dominio del problema**
            - Construcciones UML no orientadas al dominio que no tienen una contraparte en el modelo BWW, pero pueden todavía ser partes necesarias o útiles de UML en relación al a los mundos de sistemas o del desarrollo.
        - **Genuino**
            - Construcciones UML claramente intencionadas para representar el dominio del problema, pero que no tienen una contraparte en el modelo BWW.
- **Déficit de Construcciones** ("Construct Deficit").
    - Cuando no hay construcción para representar a un concepto. Usualmente, es una situación problemática.

2.5 Métricas de software

2.5.1 Objetivos de las Métricas de Software

A continuación se citan algunos ejemplos de las preguntas principales desde la perspectiva de los administradores y los desarrolladores para entender y controlar el desarrollo de un proyecto de software [Fenton97].

Administrador

1. ¿Cuánto cuesta cada proceso?
   - Se mide el tiempo, el dinero y el esfuerzo de los procesos.
2. ¿Qué tan productivo es el personal?[13]
   - Se mide el tiempo para desarrollar el sistema. Útil para conocer el costo y la duración de los cambios.
3. ****¿Qué tan bueno es el código que se está desarrollando?**
   - Registro de defectos, fallas y cambios para medir la calidad del software.
4. ***¿Qué tan satisfecho estará el usuario con el producto?**
   - Características que sugieran que el cliente esté satisfecho. La funcionalidad se puede medir con los requerimientos implementados adecuadamente.
5. ¿Cómo se puede mejorar?
   - Se puede tomar el tiempo para cada actividad principal de desarrollo y calcular su efecto en la calidad y la productividad, para elegir la mejor práctica.

Desarrollador

6. ¿Se pueden probar los requerimientos?
   - Se pueden analizar para determinar si su satisfacción puede ser expresada de forma medible y objetiva.
7. ****¿Se han encontrado todos los defectos?**
   - Se refiere a la contabilización de defectos en la especificación, diseño, código y plan de prueba, así como a su rastreo para encontrar sus causas iniciales.
8. ****¿Se han cumplido nuestras metas del producto o del proceso?**
   - Estas características nos pueden decir si se ha satisfecho un requerimiento, cumplido con los estándares o con una meta del proceso.
9. ¿Qué pasará en el futuro?
   - Se pueden medir los atributos de productos y procesos actuales para predecir su valor en el futuro, tanto de las métricas como de las versiones futuras del producto.

El presente trabajo pretende dar algunas recomendaciones razonadables orientadas a las Disciplinas del Modelado del Dominio y Diseño tomando como guía a las preguntas 3, 7 y 8 (**), con lo que se pudiera tener alguna sugerencia para la pregunta 4 (*). Dado que las preguntas 3 y 7 se refieren a los defectos, la propuesta tratará sobre la eliminación o la justificación de las Discrepancias que pueden representar situaciones problemáticas (i.e.

[13] Por ejemplo, con los datos de [Smith99] se puede estimar, que en promedio, la productividad mensual de un programador es de 233 LOC.

que originen defectos). Al reducir el número de defectos se esperaría una mayor satisfacción del usuario (pregunta 4).

A continuación, se responden brevemente las preguntas seleccionadas:

3. ¿Qué tan bueno es el código que se está desarrollando?
   - Entre más controlada se tenga la Brecha Representacional (i.e. todas las Discrepancias justificadas), se tendrá un código más completo, consistente y simple con lo que se mejora la corrección y la facilidad del mantenimiento (ver "Métricas de software propuestas dentro del modelo de calidad de McCall" en la página 135).
7. ¿Se han encontrado todos los defectos?
   - La localización e identificación de las Discrepancias resalta los elementos (e.g. clases) que posiblemente originen defectos. La revisión de estos elementos conducirá a la justificación o eliminación de las Discrepancias. La medición de las Discrepancias ofrece una idea sobre la cantidad de defectos potenciales (i.e. Discrepancias) en el análisis y el diseño (ver "Parte 2. Solución propuesta" en la página 60).
8. ¿Se han cumplido nuestras metas del producto o del proceso?
   - Se pudiera establecer un estándar de comparación para la correspondencia deseada entre los productos (i.e. Modelos del Dominio y del Diseño). Si las métricas obtenidas de la Brecha sobrepasan estos estándares, se pudiera pensar en que no se tiene una solución que corresponde al problema. Cabe mencionar que un Brecha estrecha no necesariamente es lo mejor (ver "Comentarios sobre la Brecha Representacional de Larman" en la página 136).
   - De igual forma, se puede tener un estándar de comparación para la eficiencia de los modeladores, que indicará qué tantas Discrepancias se tuvieron que eliminar (ver "Métricas de la eficiencia del modelador" en la página 128).

2.5.2 La Teoría Representacional de la Medición

En cualquier actividad de medición, se necesitan seguir ciertas reglas. Estas reglas son útiles tanto para tener una forma consistente de medir como para la interpretación de los datos. También son útiles para codificar el entendimiento inicial del problema y luego definir otras métricas más sofisticadas conforme se tiene mayor conocimiento del fenómeno. A continuación, se resume la clara exposición de [Fenton97] sobre la Teoría Representacional de la Medición.

Relaciones empíricas

La teoría representacional de la medición busca formalizar la intuición que se tiene de la forma en que trabaja el mundo real. Es decir, los datos que se obtienen representan atributos de las entidades observadas. Entonces, la manipulación de estos datos debería de preservar las relaciones que observamos entre las entidades.

Dado que se tiende a entender las cosas comparándolas entre sí, no mediante la asignación de números, las "relaciones empíricas" son la base de la medición. En una

relación empírica binaria[14] hay un consenso razonable sobre las diferencias entre un par que sean similares a las relaciones del mundo real.

Estas relaciones se pueden ver como una conversión del mundo empírico y real a un mundo formal matemático. Esta conversión se puede hacer entre varios atributos y varias métricas. Entonces, el mundo real es el dominio de la conversión (i.e. función) y el mundo matemático es el rango (i.e. contradominio). Por ello, se tienen muchas opciones para esta conversión como números reales, enteros y hasta símbolos no numéricos. De esta manera, una medida debe especificar el dominio, el rango y las reglas de conversión.

La "condición de representación" sostiene que una conversión de la medición μ debe convertir las entidades en números y las relaciones empíricas en relaciones matemáticas que preserven el orden de las relaciones.

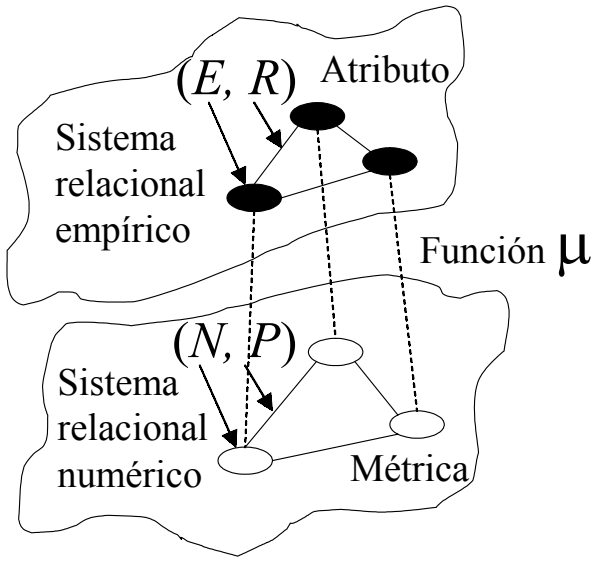


Figura 2.3. Modelo de la representación de las relaciones

Por ejemplo, en la Figura 2.3, se muestra como las relaciones empíricas *R* entre las entidades *E* del mundo real corresponden con las relaciones numéricas *N* de las entidades del mundo matemático *P*. Dado que la función μ convierte el sistema relacional empírico (*R, E*) al numérico (*N, P*) se cumple la condición de representación. Se puede considerar una función binaria ε de las relaciones empíricas y ν de las relaciones numéricas. Entonces formalmente, se puede establecer que $\varepsilon(x, y) \Leftrightarrow \nu(\mu(x), \mu(y))$ donde *x* y *y* son entidades del mundo real.

En resumen, puede haber muchas diferentes formas de medir un atributo dado. Cualquier medida que cumpla la condición de representación es una medida válida. Además, mientras más rico sea el sistema empírico relacional, habrá menos medidas válidas en el

[14] Una relación empírica binaria compara uno o varios atributos, entre dos entidades. Puede haber relaciones con más entidades (e.g. ternaria, enaria, etc.)

sistema relacional numérico. Se considera un sistema relacional rico aquél que tiene un número grande de relaciones que puedan ser definidas [Fenton97] y [Purao03].

Medida, métrica y medición

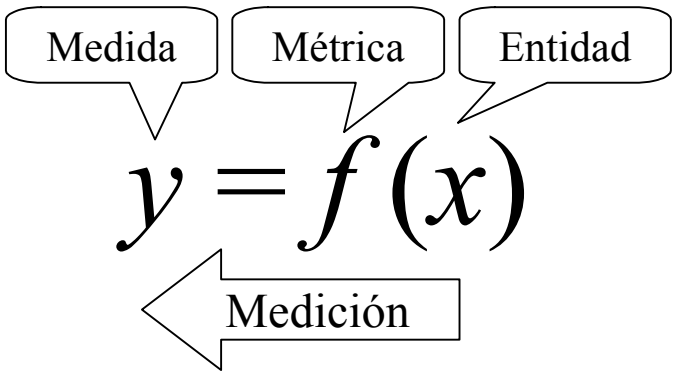


Figura 2.4. Diferencias entre medida, métrica y medición[15]

A continuación, se pueden apreciar las distinciones entre estos términos similares [Fenton97]:

- Métrica
  - Es una función que asigna un número o símbolo a una entidad para caracterizar uno o varios atributos.
- Medida
  - Valor de la métrica.
- Medición
  - Proceso en el que la métrica es calculada a partir de los atributos de las entidades del mundo real empleando reglas claras y definidas.

2.5.3 Métricas de software de la literatura

Se pueden consultar algunos “Ejemplos de clasificación de métricas” en la página 76.

Estudio sobre las métricas existentes

Abreu

Si bien, [Abreu99] reportó la existencia de más de 300 métricas, la cantidad de ellas para las etapas tempranas es relativamente muy poca. Esto es, hay más métricas de software para la Implementación que para el Diseño, y menos para el Análisis.

Hogan

Una de las métricas más cercanas al presente trabajo es la mención de [Hogan97] sobre el conteo de dependencias entre los diagramas de módulos. Sin embargo, no ofrece mayor detalle sobre su utilidad. Además se refiere a todos los tipos de dependencias (e.g. con estereotipos “friend”, “import”, “trace”, para la lista completa ver “Categorías de las relaciones y tipos de dependencias” en la página 48), mientras que la solución propuesta sólo se refiere a las dependencias de refinamiento entre los elementos (e.g. paquetes,

[15] La traducción empleada fue medida para “measure” y medición para “measurement”

clases y atributos) de los diagramas de clases. Esta encuesta selecciona 117 métricas de software OO enfocadas al reuso (o reutilización), calidad y productividad.

Kim

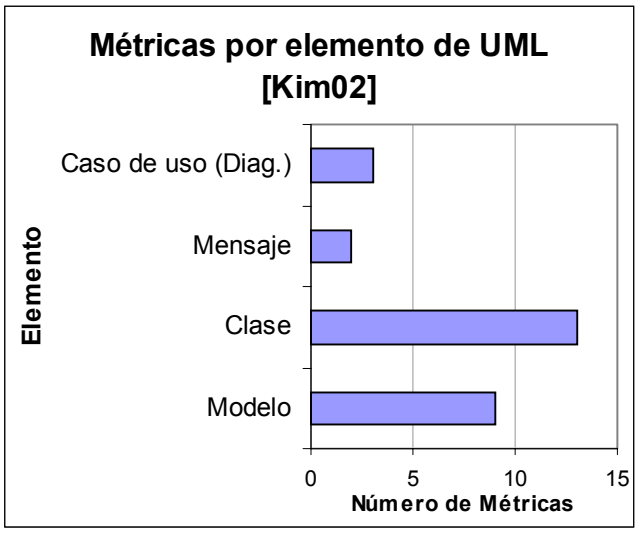


Figura 2.5. Cantidad de métricas por elemento de UML

Tabla 2.7. Cantidad de métricas por elemento de UML [Kim02]

Métricas de UML	Cantidad	Relativo
Caso de uso (Diagramas)	3	11%
Mensaje	2	7%
Clase	13	48%
Modelo	9	33%
Total	**27**	**100%**

En la Figura 2.5 y en la Tabla 2.7, se reorganizó el conjunto de 27 métricas para medir los elementos de UML propuestas por [Kim02]. Se puede apreciar que hay pocas métricas para las etapas tempranas (i.e. casos de uso, 11%) con respecto a las demás. Las métricas de clase y modelo, no se consideran métricas del Modelo del Dominio porque consideran a los métodos, que no forman parte del Modelo del Dominio.
Parte de su trabajo fue la implementación de una herramienta (UML Metrics Producer, UMP[16]) para recabar sus métricas. Una implementación similar pero para las métricas propuestas se considera como parte del "Trabajo Futuro", página 192.

De las 27 métricas de [Kim02] (para elementos de UML), sólo hay dos métricas que a pesar de ser las más cercanas a las propuestas, están más alejadas que las de [Hogan97].

- Acoplamiento entre clases ("Coupling between classes, CBC")

[16] Se empleó BasicScript de Rational Rose. El tamaño fue de 1152 LOC con comentarios (in-line comments)

  - También se le conoce como “canales de entrega de mensajes”. Esta métrica cuenta el número de asociaciones en una clase y los atributos, para los cuales los parámetros son del tipo de la clase. Los mensajes pueden ser enviados cuando un objeto de una clase mantiene una referencia a otro objeto de la clase
- Número de clases en el modelo (“Number of the classes in a model, NCM”)
  - Esta métrica es comparable al tradicional LOC (líneas de código)

Originalidad

Se puede afirmar que hay más métricas en etapas tardías que en etapas tempranas. Esto se basa en que si [Abreu99] reporta más de 300 métricas tres años antes de las 27 métricas de diseño recopiladas por [Kim02], se puede decir que al menos, existen aproximadamente diez veces[17] más métricas en etapas tardías (e.g. codificación) que en etapas tempranas (e.g. diseño).

Se realizaron búsquedas bibliográficas sobre los artículos que hayan citado a Larman y que hablaran de la Brecha Representacional, pero no se encontraron referencias relacionadas. Por las anteriores referencias, se infiere que pueden existir muy pocas métricas en la misma área, por lo que se concluye que el presente trabajo contribuye de manera significativa a las métricas en las etapas tempranas. Por ejemplo, se puede decir que si sólo una de las 117 métricas [Hogan97] es parecida, entonces muy probablemente las métricas propuestas son novedosas.

[17] 10.11 veces = (300 [Abreu99] – 27 [Kim02]) / 27 [Kim02]. Se supone que las métricas de [Abreu99] incluyen de todas las etapas, mientras que las de [Kim02] son sólo de diseño. Además, se puede considerar que para el año 2002, habría más de 300 métricas.

Parte 2. Solución propuesta

Capítulo 3. Preliminares de la solución propuesta

3.1 Selección de los Artefactos

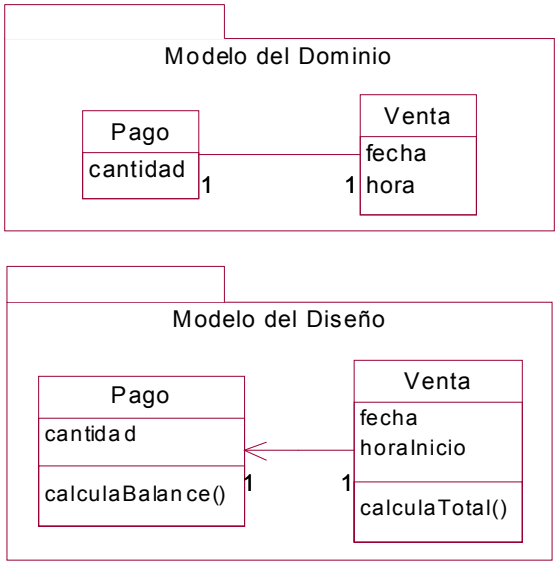


Figura 3.1. Ejemplo de la Brecha Representacional [Larman01]

A continuación se enlistan varias de las razones por las que se eligieron los Artefactos del Modelo del Dominio y del Modelo del Diseño:

- Se identificó que de acuerdo a la definición de Brecha Representacional de [Larman01], el modelo mental se representa con el Modelo del Dominio y la solución con el Modelo del Diseño.
  - Para la representación del Modelo de Dominio sólo se cuenta con diagramas de clases (como en la Figura 3.1). Mientras que para la representación del Modelo de Diseño se tienen los diagramas de clases entre otros, como los diagramas de secuencia.
- Ambos son parte de las Disciplinas tempranas.
  - Reducir defectos en Disciplinas tardías, es más costoso que en Disciplinas tempranas.
- Ambos contienen elementos de similar naturaleza.
  - Los diagramas de clases del dominio (análisis) y los diagramas de clases de diseño, tienen los mismos elementos y siguen las mismas reglas, aunque representan distintos enfoques: el de dominio representa conceptos del mundo real, mientras que el de diseño, objetos de software.

Cabe mencionar que, usualmente, el Modelo del Dominio está a cargo del analista y el Modelo del Diseño, del diseñador.

3.2 Selección de los elementos

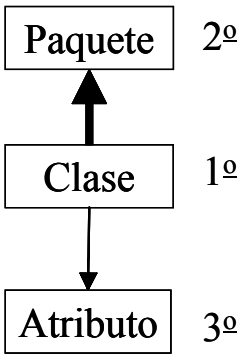


Figura 3.2. Análisis de Discrepancias centrado en las clases

Cabe resaltar que la Brecha Representacional primordialmente se refleja en las clases. Ahora bien, de los elementos usuales de los Diagramas de Clases, se explicarán las razones por las que fueron o no seleccionados. En la Figura 3.2, se muestran las prioridades en la selección del elemento para el análisis de las Discrepancias. Los elementos seleccionados fueron los paquetes, clases y atributos. Aunque el resto se considera como no seleccionado se justifica la exclusión (para el presente trabajo) de las asociaciones, generalizaciones y métodos.

3.2.1 Seleccionados

Del metamodelo de UML

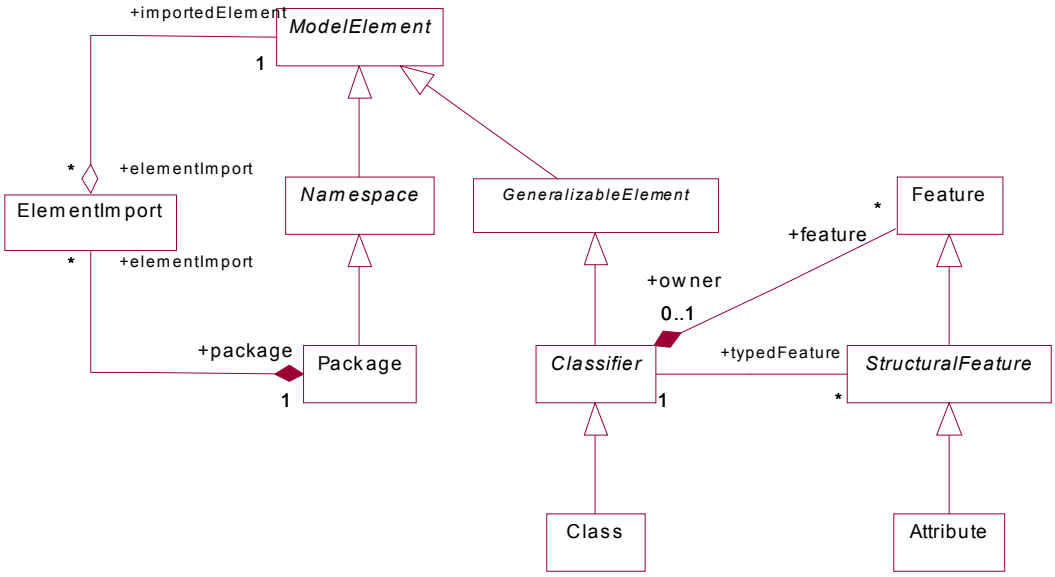


Figura 3.3. Integración de extractos del metamodelo de UML para mostrar la composición de los elementos seleccionados

Los elementos seleccionados tienen la facultad de participar en una composición jerárquica. La justificación se sostiene del metamodelo de UML. En la Figura 3.3, se muestra la integración de varios extractos del metamodelo de UML [OMG03a][18].

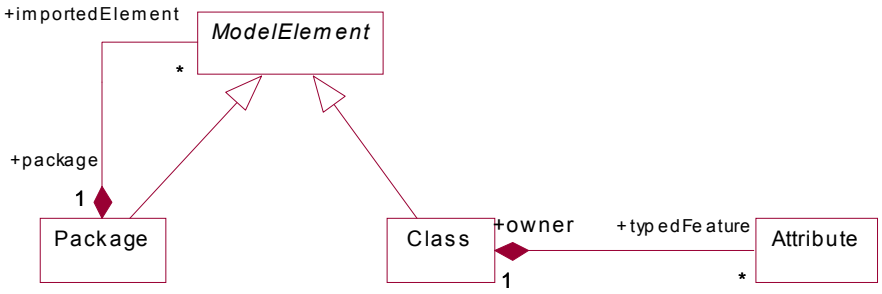


Figura 3.4. Composición de los elementos seleccionados

De la Figura 3.3, se puede deducir la composición jerárquica de los elementos seleccionados que se muestra en la Figura 3.4. Esto es, se puede decir que un paquete se puede componer de varios paquetes y clases; así como que una clase se puede componer de varios atributos. Estos elementos de modelo son independientes pues no necesitan de otros elementos para existir.

[18] [OMG03a] Se tomaron extractos de las figuras 5-2, 5-4, 5-21 (ver " Apéndice C. Metamodelo de UML")

Clases

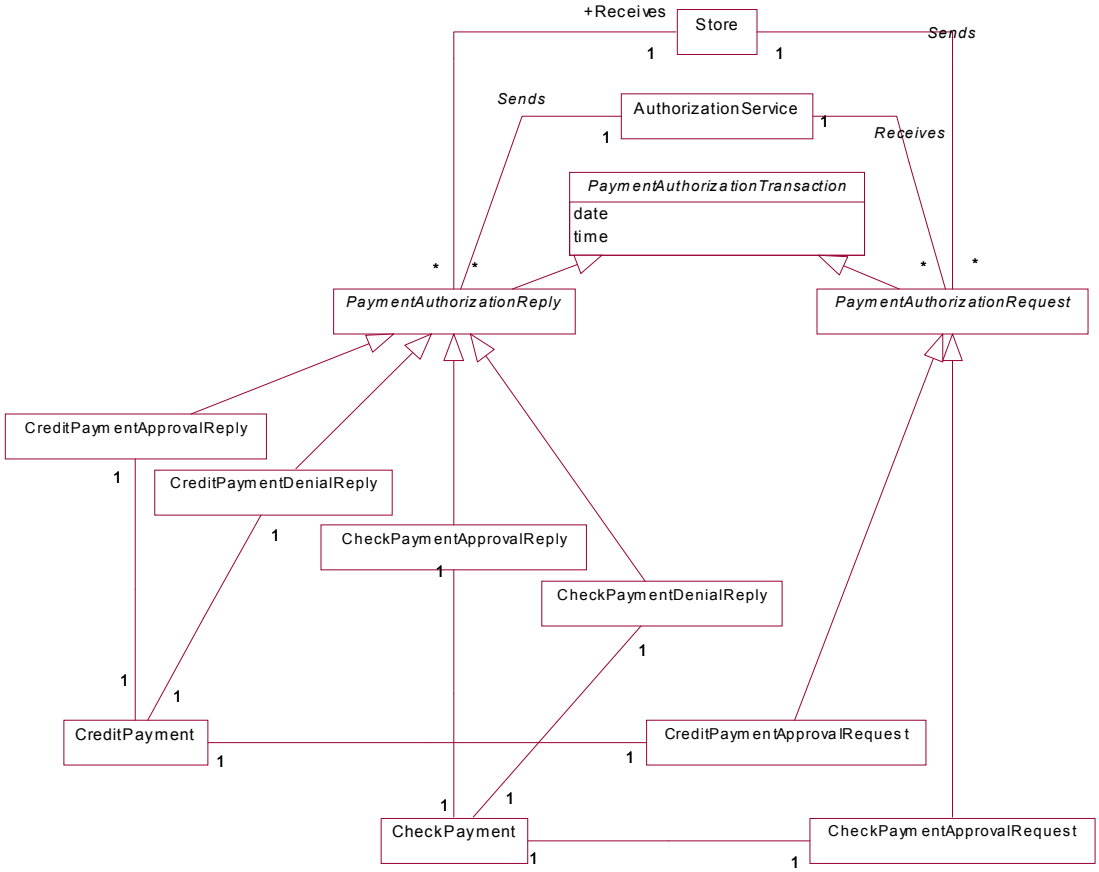


Figura 3.5. Diagrama de clases del paquete "AuthorizationTransaction", en el Modelo del Dominio "NextGen POS"

La correspondencia entre clases es la más importante en este análisis por varias razones:

- Las clases son la esencia de los modelos seleccionados. En el Modelo del Dominio son la base para la representación de los conceptos. De igual forma, las clases son la modelación directa de las entidades de software. Sería (casi[19]) ilógico tener un diagrama de clases sin clases.
- Generalmente, el Modelo del Dominio inicial está sólo compuesto por Clases Conceptuales [Larman01].
- Usualmente, las Clases Conceptuales tienen menos atributos que las Clases de Diseño. Esto se puede apreciar en la Figura 3.5 y en el 'Apéndice D. Modelo del Dominio "NextGen POS (Point of Sale)"', página 206.

[19] Los elementos usuales de la vista del Modelo de Administración (en UML) no emplean clases sino paquetes, subsistemas y modelos [Booch99 p.24]

- El enfoque del Proceso Unificado es orientado a objetos, no a paquetes, ni a atributos u otro tipo de elemento. A final de cuentas, un objeto es una instancia de una clase.

Paquetes

La correspondencia entre los paquetes permite separar los conjuntos de Discrepancias y por tanto, las métricas. Con ello, se puede tener un nivel de medición con mayor detalle, promoviendo la modularidad y un mejor mantenimiento, por el hecho de tener conjuntos aislados.

La primera actividad de las nueve mostradas en [Pressman01], es la partición del modelo del análisis (i.e. Modelo del Dominio) en subsistemas (i.e. paquetes). Con ello, se pretende definir colecciones altamente cohesivas de clases, relaciones y comportamiento. Estos paquetes se caracterizan por sus responsabilidades, es decir puede ser identificado por los servicios que se proveen [Pressman01]. Los paquetes deben adaptarse a los siguientes criterios de diseño:

- Deberían tener una interfaz bien definida en la que ocurra toda la comunicación con el resto del sistema.
- Con la excepción de un número reducido de "clases de comunicación", es decir, las clases que dentro del paquete deban colaborar sólo con otras clases dentro del paquete.
- El número de paquetes debería mantenerse bajo.
- Pueda ser particionado internamente para ayudar a reducir la complejidad. Como consecuencia de ello, se incurre en el "diseño de capas" que organiza a los paquetes y a sus elementos (e.g. clases, otros paquetes) de forma sistemática y jerárquica.

El análisis de las Discrepancias ayuda a organizar los paquetes alineando el Modelo del Diseño con el Modelo del Dominio para tener una mayor cohesión. Los refinamientos de los elementos que componen a los paquetes, resaltan las partes con las que se comunica el paquete, que una vez señaladas, el modelador deberá decidir si son o no son "clases de comunicación".

Con fines didácticos, se anticipará sin detalles la forma gráfica y tabular para representar un ejemplo teórico. Las mayúsculas corresponden al Modelo de Dominio y las minúsculas, al Modelo de Diseño.

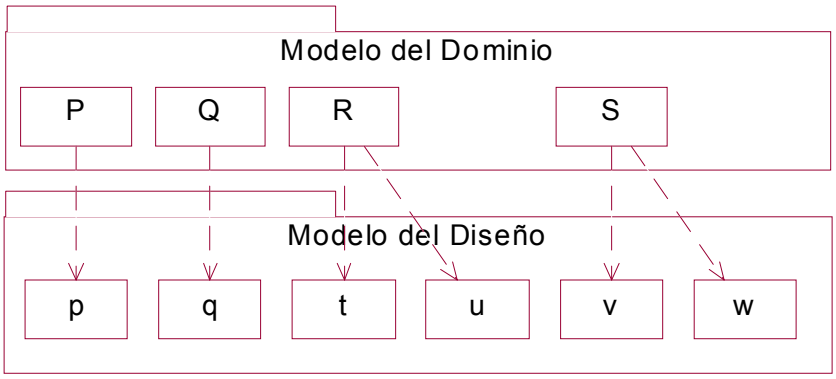


Figura 3.6. Modelo de Dominio y Modelo de Diseño sin paquetes

Tabla 3.1. Matriz de relaciones sin paquetes

Paquete.Clase Dominio\Diseño	p	q	t	u	v	w
P	1	0	0	0	0	0
Q	0	1	0	0	0	0
R	0	0	1	1	0	0
S	0	0	0	0	1	1

$0 < \mu < 1, \mu \approx 0.5$

Se pudiera tener la siguiente situación. Dado un modelo donde la mitad de las clases tienen relaciones uno a uno y la otra mitad, redundancias; la razón del modelo sería alrededor de 0.5 (ver Figura 3.6 y Tabla 3.1). Para tener una idea, convendría estimar intuitivamente una métrica normalizada, $\mu \in [0, 1]$. Se pudiera suponer que hay dos conjuntos separados de paquetes, {A, a} y {B, b}. Si no se considerara a los paquetes, posiblemente se ocultarían las situaciones siguientes:

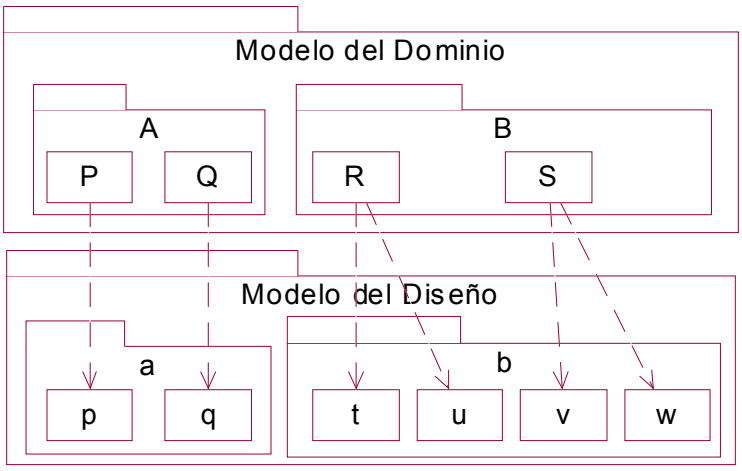


Figura 3.7. {A, a} tiene una Brecha más estrecha que {B, b} (situación 2)

Tabla 3.2. Los paquetes {A, a} no tienen Discrepancias mientras que los paquetes {B, b}, tienen varias Discrepancias (redundancias)

Paquete.Clase Dominio\Diseño	a.p	a.q
A.P	1	0
A.Q	0	1

$\mu_A = 0$

Paquete.Clase Dominio\Diseño	b.t	b.u	b.v	b.w
B.R	1	1	0	0
B.S	0	0	1	1

$\mu_a < \mu_B, \mu_B > 0.5$

- **Situación 1.** Los paquetes {A, a} tienen las relaciones uno a uno, mientras que los paquetes {B, b}, tiene las demás (ver Figura 3.7 y Tabla 3.2). Los paquetes {A, a} requerirán de poco análisis pues su Brecha es nula (métrica mínima), mientras que los paquetes {B, b} necesitarán más tiempo para analizar las correspondencias (métrica alta).

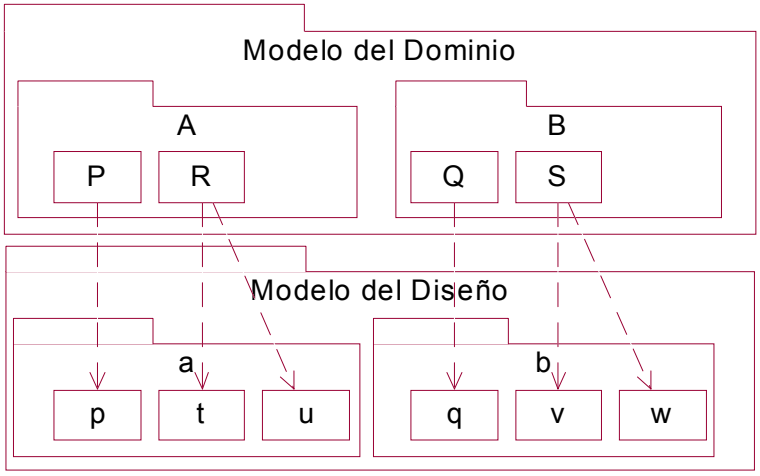


Figura 3.8. Parece que la Brecha de {A, a} es similar que la de {B, b} (situación 2)

Tabla 3.3. Los paquetes {A, a} y {B, b} tienen la misma métrica. Ambos tienen una relación uno a uno y una Discrepancia (redundancia)

Paquete.Clase Dominio\Diseño	a.p	a.t	a.u
A.P	1	0	0
A.R	0	1	1

Paquete.Clase Dominio\Diseño	b.q	b.v	b.w
B.Q	1	0	0
B.S	0	1	1

$\mu_A = \mu_B$

- **Situación 2.** Los paquetes {A, a} y {B, b} tienen las mismas métricas de correspondencia (ver Figura 3.8 y Tabla 3.3). Se pudiera pensar que fueron analizadas y diseñadas de forma similar o consistentemente, posiblemente por una misma persona.

Atributos

La correspondencia entre atributos es un análisis más detallado que entre clases. Con el propósito de modelar la parte más relevante, manteniendo la simpleza, se propone establecer la correspondencia sólo entre la existencia de los atributos, no se consideran aspectos como la visibilidad y el tipo de datos.

De cualquier forma, para establecer los refinamientos entre clases (y por tanto, las posibles Discrepancias), es conveniente poner atención a los atributos de las mismas. Aunque el nombre de una Clase Conceptual corresponda con el de una Clase de Diseño, puede suceder que alguna de ellas tenga un atributo que se encuentre en una tercera clase, lo cual puede pasar inadvertidamente. Esto, debería de reflejarse con un refinamiento extra (i.e. que equivaldría a una Sobrecarga o una Redundancia). De forma similar, para eliminar una Discrepancia (como la anterior) se debería de mover (o eliminar) el atributo a una clase más apropiada.

Dado que un Modelo del Dominio pudiera tener muy pocos atributos con respecto al Modelo del Diseño, se incurría en muchas Redundancias (ver 'Apéndice D. Modelo del Dominio "NextGen POS (Point of Sale)"'); lo cuál no necesariamente es erróneo, pero puede ocultar las Redundancias problemáticas (i.e. sería difícil identificar las Redundancias problemáticas con 100 atributos del Dominio vs. 500 atributos del Diseño y 250 Redundancias). De los elementos seleccionados, los atributos no cuentan con una representación gráfica (i.e. dependencias de refinamiento[20]). Para establecer la correspondencia entre los atributos se debe crear una tabla (independiente de la notación UML) como se muestra en la Tabla 5.1, página 93. A pesar de ello, el análisis entre atributos es útil para distinguir las correspondencias complejas por tener mayor detalle.

3.2.2 No seleccionados

En el presente trabajo, no se considera explícitamente la correspondencia entre los siguientes elementos: asociación, generalización, dependencia, realización e interfaz. En realidad, se consideran como parte de un trabajo futuro, que convendría experimentar para probar su utilidad . En general, no se consideran a estos elementos por ser dependientes (ver "Apéndice A. Glosario") de otros elementos. Por ejemplo, una generalización requiere para existir una clase padre y una clase hijo. Otras razones para esta exclusión son:

- Faltaría una forma gráfica para establecer la correspondencia. Al menos, las dependencias de refinamiento serían ilegales para los elementos de generalización, dependencia y realización[21]. Aun si se propusiera una forma gráfica, los diagramas serían más complejos (i.e. más flechas).
- Se necesitaría una tabla externa. Si el mantenimiento de los modelos, muchas veces es descuidado por su complejidad y dificultad, el mantenimiento de una tabla más compleja que la de atributos, sería probablemente relegada. La complejidad radicaría en que la columna y renglón de los encabezados estarían

[20] Al menos en Rational Rose, no se puede establecer una dependencia entre los atributos.
[21] Esto fue probado en Rational Rose, aunque en el metamodelo de UML (ver figuras 5-2 y 5-4) es válido establecer una dependencia entre cualquier relación.

compuestas por varios elementos (usualmente entre uno y tres) que se aprecia en la Tabla 3.4.

Tabla 3.4. Posible tabla para un elemento no seleccionado (asociación). Usualmente, *n* y *m* valen 1, 2 o 3

Asociación	**Diseño**	**Clases de Diseño**	**d[1]**	DA	DD	…	DA
Dominio			**d[2]**	DB			DX
			…	…			…
Clases Conceptuales			**d[m]**	DC			DY
c[1]	**c[2]**	**…**	**c[n]**				
CA	CB	…	CC	1	0	…	1
CD	CF			0	1		0
…				…			…
CE	CG	…	CH	1	0	…	1

Sin embargo, a la hora de establecer las correspondencias se debe prestar atención a que los elementos participantes sean similares en ambos modelos. De lo contrario, se pudieran establecer correspondencias parcial o incorrectamente.

Del metamodelo de UML

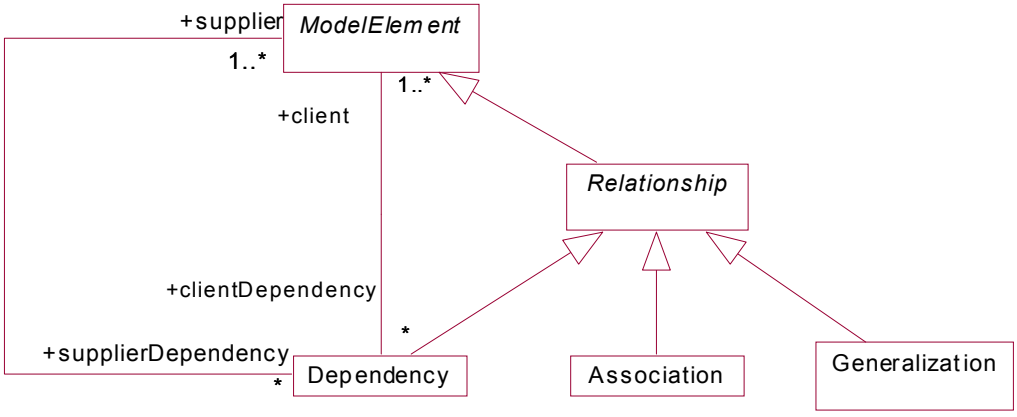


Figura 3.9. Los elementos no seleccionados son relaciones. Se pueden tener dependencias entre relaciones

Los elementos no seleccionados son relaciones, por lo que no pueden existir de forma independiente [OMG03a][22]. Aunque usualmente las dependencias no se emplean para las relaciones, en el metamodelo de UML son válidas, como se puede apreciar en la Figura 3.9.

Asociaciones

Si bien es posible[23] establecer un refinamiento entre dos asociaciones, puede resultar muy complicado. Establecer la correspondencia implicaría considerar, no sólo los elementos

[22] [OMG03a] Se tomaron extractos de las figuras 5-3 y 5-4
[23] Esto fue probado en Rational Rose

de la asociación (e.g. clases), sino también las características adicionales[24] como multiplicidad, navegabilidad, agregación y estereotipo, así como las combinaciones entre ellos.

Intuitivamente, las asociaciones son muy similares entre el Modelo del Dominio y el del Diseño. Las Discrepancias de relaciones por regla son similares a las de clases. Hay que recordar que se debe prestar atención a las relaciones en el momento de la verificación.

Generalizaciones

Aunque no se recomienda establecer correspondencias entre las generalizaciones, se sugiere que se tomen en cuenta en el establecimiento de las correspondencias. El prestar atención a los atributos puede ayudar a identificar alguna herencia.

Para ilustrarlo, se supone este modelo de ejemplo: se tienen las Clases Conceptuales CA, CB y CC, así como las Clases de Diseño DA y DB. Además, CA y CB heredan de CC, mientras que DA y DB no tienen herencia. Al establecer las correspondencias, se tendrían las parejas (CA, DA) y (CB, DB). La herencia nos sugiere establecer una correspondencia (CC, DA) y (CC, DB) pues posiblemente, DA y DB sean similares (i.e. tengan varios atributos en común). Esta situación se presentó en el caso de estudio sobre las matrices, entre las Clases Conceptuales, "TriangularMatrix", "LowerMatrix" y "UpperMatrix" con la Clase de Diseño "TriangularMatrix" (ver 'Caso de estudio de matrices: "AbstractMatrix"' en la página 137).

Métodos

Por definición, las Clases Conceptuales no pueden tener métodos. Esto, debido a que el Modelo del Dominio es un diccionario visual de los conceptos, no existe para modelar el comportamiento.

3.2.3 Utilidad de los elementos y análisis central de las Discrepancias

.

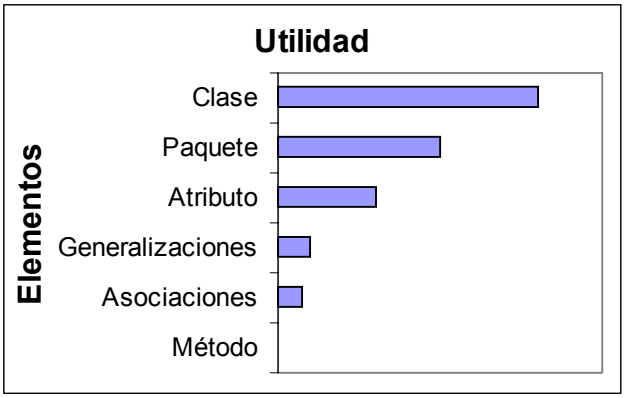


Figura 3.10. Apreciación de la utilidad intuitiva de los elementos usuales del diagrama de clases

[24] Por claridad, se empleó "características adicionales" en vez de "adornos" que sería la traducción literaria de los "adornements" de UML

Por lo expuesto para cada elemento, se muestra en la Figura 3.10, la apreciación de la utilidad de los elementos usuales en los diagramas de clases. Aunque las barras no representan un valor numérico y la apreciación es un tanto subjetiva, tratan de resumir la justificación expuesta en una cierta utilidad. En particular, al analizar las Discrepancias entre las clases se cubre buena parte de la problemática. Si se incluyen a los paquetes y a los atributos (en ese orden) se podría esperar que se cubra la mayor parte del problema.

Como aplicación de este orden, se sugiere un análisis centrado en las clases. Esta recomendación se puede apreciar en la Figura 3.2 y en la Figura 3.10[25]. También se sugiere que se inicien los establecimientos de los refinamientos entre las clases, para luego organizarlas en paquetes de alta cohesión entre las clases y bajo acoplamiento entre los modelos (i.e. evitar Discrepancias Representadas, ver "Categorías propuestas de Discrepancias"). Si aún se desea mayor detalle, se puede incluir el análisis de Discrepancias entre los atributos para cuestiones de eliminación y justificación de Discrepancias entre clases..

[25] Si por cuestiones de recursos o porque no se desea mucho detalle, el análisis de Discrepancias entre las clases puede ser suficiente

3.3 Entendiendo la Brecha Representacional mediante conjuntos

La Brecha Representacional puede ser comprendida mediante los Diagramas de Venn, para ayudar a entender intuitivamente el significado y la medición de la Brecha Representacional.

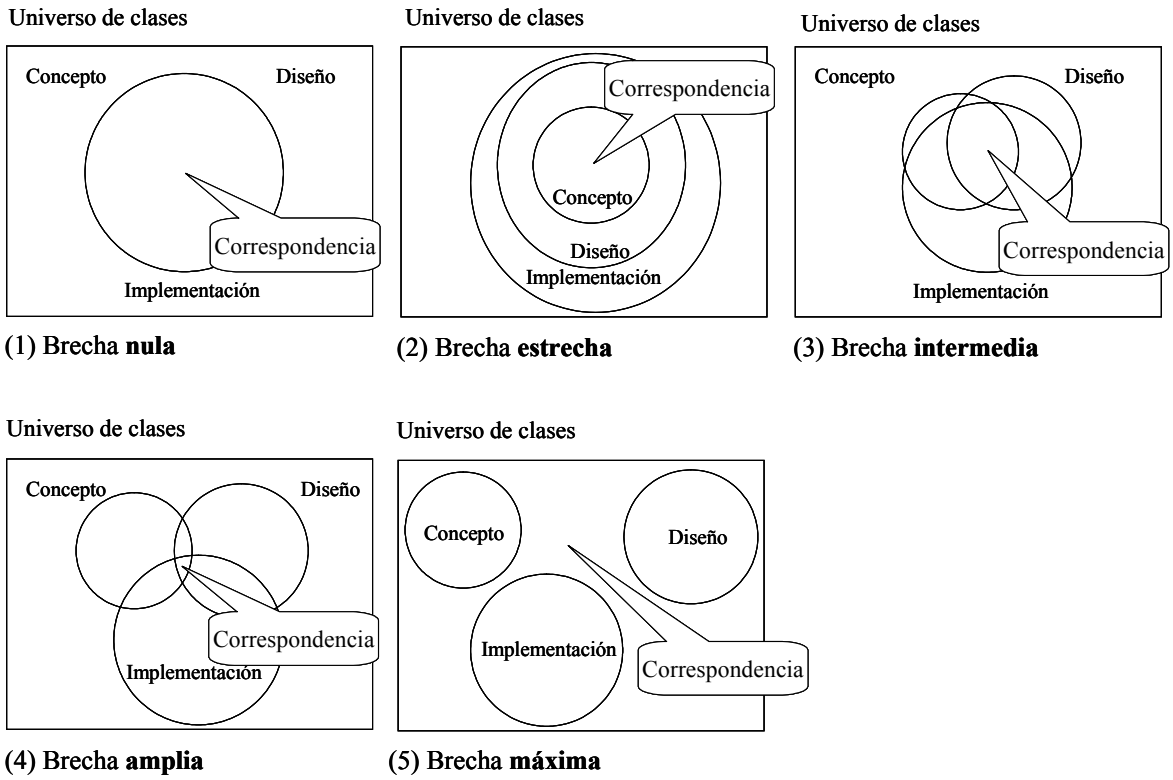


Figura 3.11. Brecha Representacional ilustrada con Diagramas de Venn

Aunque la Brecha Representacional se aplica a varios tipos de elementos, la explicación de esta sección se limita a las clases. En la Figura 3.11, los niveles de abstracción de las clases (e.g. Concepto, Diseño e Implementación) se representan con los círculos (i.e. conjuntos). Por lo que la intersección representa la correspondencia. En otras palabras, a mayor área de la intersección, mayor estrechez de la Brecha. El problema con esta representación es la forma de definir y medir estas intersecciones. Más adelante se explicará la interpretación de la Brecha mediante las Discrepancias Ontológicas (ver "Discrepancias Ontológicas" en la página 85).

Para que la Brecha sea nula (deseable para [Larman01]), se requiere que la modelación de los conceptos del mundo real corresponda directamente con la representación de las entidades de software –esto se aprecia en el inciso (1) de la Figura 3.11–. Existen muchas formas en las que la Brecha se podría ir ampliando, como sucede en los incisos (2), (3) y (4). El caso de subconjuntos se muestra en el inciso (2), donde las Clases Conceptuales son un subconjunto de las Clases de Diseño, que a su vez lo son de las Clases de

Implementación. La transición de una Brecha mínima a una máxima, se ilustra con la secuencia del (1) al (5).

3.4 Métricas Intradisciplinarias, Interdisciplinarias y Multidisciplinarias

A pesar de la gran cantidad de métricas en la literatura ([Abreu99] reporta más de 300 métricas y la selección de [Hogan97] es de 117 métricas), aún existen ciertas relaciones sin medir. Lo cuál, da pie a proponer una clasificación de métricas en Intradisciplinarias, Interdisciplinarias y Multidisciplinarias.Luego de comentar las características de estos tipos, se verán algunos "Ejemplos de clasificación de métricas".

Bajo la clasificación propuesta, el presente trabajo muestra métricas Interdisciplinarias de naturaleza similar para los atributos internos de los productos (i.e. Artefactos).

3.4.1 Métricas de naturaleza similar

Se llamará de "naturaleza similar" cuando se hable de métricas de un mismo tipo que midan el mismo atributo con una misma escala de medición. Se entiende por tipo, si es de productos (tamaño, acoplamiento, confiabilidad, etc.), procesos (tiempo, defectos, estabilidad, etc.) o recursos (costo, velocidad, productividad, etc.); por atributo: interno (tamaño, acoplamiento, tiempo, defectos, costo, velocidad, etc.) o externo (confiabilidad, estabilidad, productividad, etc.); y por escala de medición (nominal, intervalo, absoluto, etc.). Para mayor detalle, se puede consultar el "Apéndice A. Glosario".

El caso contrario se considera como de "naturaleza diferente". Por ejemplo, LOC/hora es de naturaleza diferente, ya que LOC es una medida de producto, del atributo interno de tamaño y de escala de razón[26]; mientras que las horas son métricas del proceso, del atributo interno de tiempo y de escala de intervalo

Se cree conveniente emplear el término "similar" en vez de "igual", pues pueden ser entidades del mismo modelado pero para distinto propósito; por ejemplo, las Clases Conceptuales y las Clases de Diseño, siguen las mismas reglas de modelado, pero tienen diferente aplicación.

3.4.2 Métricas Intradisciplinarias

De particular importancia, se percibe que las métricas miden algún atributo en una sola Disciplina, o son cálculos con métricas de elementos de distinta naturaleza. Se propone llamarlas "métricas Intradisciplinarias" para el contexto del Proceso Unificado y "métricas intra-etapa" para otras metodologías. Un ejemplo de métrica Intradisciplinaria: número de Clases Conceptuales.

3.4.3 Métricas Interdisciplinarias

Por lo anterior, sería deseable poder medir entidades de diferentes Disciplinas pero de naturaleza similar. Cabe resaltar que se propone medir a las entidades, no a las métricas sobre estas entidades. De igual forma, a estas se propone llamarlas "métricas

[26] Es un error común asumir que LOC es una escala absoluta que mide la longitud porque se obtiene del conteo. No puede ser absoluta, debido a que hay muchas formas de medir la longitud (i.e. número de caracteres, número de palabras) [Fenton97] p. 53.

Interdisciplinarias" y "métricas inter-etapa". De hecho, no se encontraron métricas de este tipo. Un ejemplo de métrica Interdisciplinaria: métricas de la Brecha Representacional (presente trabajo).

3.4.4 Métricas Multidisciplinarias

Lo más cercano a las métricas Interdisciplinarias, son razones de una métrica en una Disciplina dividida entre otra métrica de otra Disciplina, pero sin considerar la estructura interna de esta correspondencia. Esto es, son métricas de una función de métricas, más no una métrica entre las dos Disciplinas. Se propone llamar a estas métricas "Multidisciplinarias" ("multi-etapa"), aunque se piensa que por lo general serán "bidisciplinarias" ("bi-etapa"). Un ejemplo de métrica Multidisciplinaria: número de Clases Conceptuales por Clase de Diseño.

3.4.5 Diferencias entre métricas Intradisciplinarias, Interdisciplinarias y Multidisciplinarias

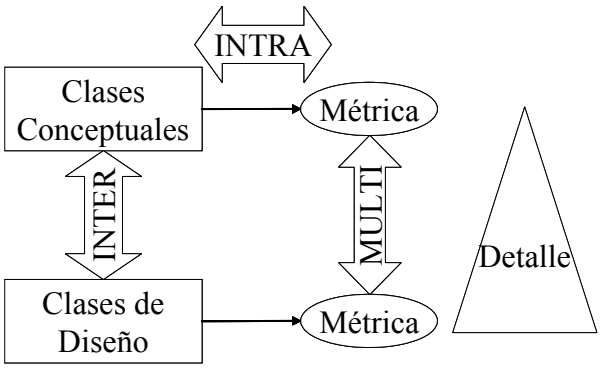


Figura 3.12. Diagrama de las métricas Interdisciplinarias, Intradisciplinarias y Multidisciplinarias

La Figura 3.12 ilustra la clasificación propuesta. La diferencia básica radica que la métrica Interdisciplinaria considera las entidades individualmente; mientras que la Multidisciplinaria, es una relación entre métricas de las entidades en conjunto. Además, lo que distingue una métrica de una Disciplina a otra, es el detalle de la abstracción.

3.4.6 Ejemplos de clasificación de métricas

Tabla 3.5. Naturaleza de las métricas. Horizontal: misma Disciplina, naturaleza distinta. Vertical: Disciplina distinta, naturaleza similar

Tipo de métrica	Productos		Procesos		Recursos	
Atributos internos	Tamaño		Tiempo		Costo	
Disciplina	**Directa**	**Indirecta**	**Directa**	**Indirecta**	**Directa**	**Indirecta**
Modelado del Dominio (Clases Conceptuales)	Número de Clases	Número de Clases /Paquete	Horas-hombre	Horas-hombre /Clase	Pesos	Pesos /Clase
Diseño (Clases de Diseño)	Número de Clases	Número de Clases /Paquete	Horas-hombre	Horas-hombre /Clase	Pesos	Pesos /Clase
Implementación (Clases de Implementación)	Número de Clases	Número de Clases /Paquete	Horas-hombre	Horas-hombre /Clase	Pesos	Pesos /Clase
Implementación	LOC	LOC /Clase	Horas-hombre	LOC /Hora-hombre	Pesos	Pesos /LOC

Se tomarán varios ejemplos de la Tabla 3.5:

- Intradisciplinaria: Número de Clases Conceptuales, Horas-diseñador
- Intradisciplinaria (de naturaleza similar): LOC/Clase de Implementación
- Intradisciplinaria de naturaleza diferente: Hora-diseñador/Clase de Diseño
- Multidisciplinaria: Razones entre métricas de distinta Disciplina:
  - De naturaleza similar: Número de Clases Conceptuales/Número de Clases de Diseño, Horas-analista/Horas-diseñador
  - De naturaleza diferente: Número de Clases Conceptuales/Horas-programador, LOC/Hora-analista.

Ejemplos de clasificación de métricas de la literatura

Varios autores: Chidamber, Kim, Hogan

Aunque, algunas métricas son muy reconocidas como el conjunto de [Chidamber94], fueron diseñadas para ser válidas en una sola etapa (i.e. Disciplina), es decir, son métricas Intradisciplinarias.

Si bien [Kim02], propone una herramienta que extrae automáticamente 27 métricas de los modelos de UML, son métricas Intradisciplinarias que en su mayoría, son conteos de elementos de modelado.

A pesar de que en [Hogan97] se mencionan 40 métricas de diagramas de clases, son métricas Intradisciplinarias.

Dumke

Tabla 3.6. Cálculo de Indicadores de [Dumke95]

Abreviación	Métrica (Indicador de:)	Numerador (Número de:)	Denominador (Número de:)
CDI	Definición de Clases	Nociones	Clases definidas
ADI	Definición de Atributos	Adjetivos o predicados	Atributos definidos
SDI	Definición de Servicios	Verbos o adverbios	Servicios definidos
CMI	Modificación de Clases	Clases organizacionales	Todas las clases diseñadas
AMI	Modificación de Atributos	Atributos organizacionales	Todas los atributos diseñados
SMI	Modificación de Servicios	Servicios organizacionales	Todas los servicios diseñados
CII	Implementación de Clases	Clases de nueva implementación	Clases diseñadas
AII	Implementación de Atributos	Atributos de nueva implementación	Atributos diseñados
SII	Implementación de Servicios	Servicios de nueva implementación	Servicios diseñados

En el trabajo de [Dumke95], se proponen algunos indicadores de diseño e implementación, que son razones entre el número de clases, atributos y métodos, divididos por el total de cada tipo. Estos indicadores se muestran en la Tabla 3.6. Sin embargo, los indicadores son mediciones sobre métricas de todo el modelo, es decir, no considera las relaciones entre cada entidad.

Tabla 3.7. Indicadores por elemento del proyecto [Dumke95]

\ Elemento Indicador	Clase	Valor	Atributo	Valor	Servicio	Valor
Definición	CDI	0.02	ADI	0.03	SDI	0.06
Modificación	CMI	0.33	AMI	0.48	SMI	0.21
Implementación	CII	0.31	AII	0.51	SII	0.22

En la Tabla 3.7, se muestran los valores del proyecto realizado por [Dumke95]. Se puede apreciar que muy pocos elementos gramaticales se "mutaron" a un elemento de diseño. Esto es, de cada cien "nociones" sólo dos llegaron a ser clases de definición.

Cabe mencionar que estas métricas no hacen distinción de la etapa (i.e. Disciplina). Y dado que miden los servicios (i.e. métodos), no se pueden referir al Modelo del Dominio. Las métricas relacionadas con la etapa de definición, al parecer relacionan las etapas de análisis y de diseño, por lo que se pueden considerar como métricas Multidisciplinarias.

Mientras que las demás, se consideran métricas Intradisciplinarias de las etapas de diseño e implementación[27].

Abreu

Las estructuras de proximidad de Poels que se mencionan en [Abreu99], son configuraciones de relaciones empíricas que describen el concepto de distancia conceptual. Sin embargo, las mediciones parten de otras métricas de software, por lo que se les considera como métricas Multidisciplinarias.

Si bien, la herramienta MOODKIT G2 presentada por [Abreu99] permite la extracción de métricas de código en Eiffel y Smalltalk, así como de lenguajes de modelado como OMT y UML, no trabaja con la interrelación entre varios niveles de abstracción (i.e. Modelado del Dominio y Diseño).

Aunque normalmente se pueden validar las métricas con algún tipo de correlación, aun resulta muy difícil relacionar las métricas de los paradigmas orientados a objetos y estructurado, con la predicción del esfuerzo del mantenimiento, según Zuse en [Abreu99].

Uno de los trabajos relacionados con el presente es el de Henderson presentado por [Abreu99]. Si bien, permite la colección de métricas en cuatro niveles de abstracción (método, clase, paquete, sistema), que en el presente contexto denominamos como elementos de modelado, no se plantean métricas entre el mismo "nivel de abstracción" pero para distinta etapa (i.e. Disciplina). Esto es, se obtienen métricas sobre las clases pero no la relación entre dos tipos de clases (i.e. Clases Conceptuales, Clases de Diseño) por lo que se consideran como métricas Intradisciplinarias.

La técnica propuesta con el fin de detectar y corregir fallas en el diseño por Sahraoui en [Abreu99] se basa en la correlación mediante una transformación, de las características de calidad (e.g. mantenimiento) y los atributos de calidad del software (e.g. métricas). Esta técnica se clasificaría como "Multidisciplinaria" pues mide a partir de métricas, y no directamente de las entidades.
A pesar de que, para 1999, existían más de trescientas métricas, en el taller ECOOP [Abreu99] se plantea la necesidad de tener métricas dinámicas (especialmente para la dinámica del modelado del diseño OO). Por esta razón, se pretende medir el cambio entre las métricas propuestas. También se menciona la necesidad de tener estándares de métricas del producto, aunque no sean "perfectas", pues la falta de validación aun constituye un obstáculo.

[27] Por motivos de distinción, las Disciplinas de UP se inician con mayúsculas y las etapas, se escriben en minúsculas

3.5 Ejemplo: personas de una facultad

Con fines didácticos, se empleará un ejemplo hipotético simplificado. El dominio se refiere a las personas que usualmente conformarían una facultad universitaria. Con el fin de ilustrar la Brecha Representacional, se introdujeron intencionalmente ciertas diferencias entre los modelos. Se pudiera pensar que el Modelo del Dominio se produjo por un analista, de forma un tanto independiente del modelo del diseñador.

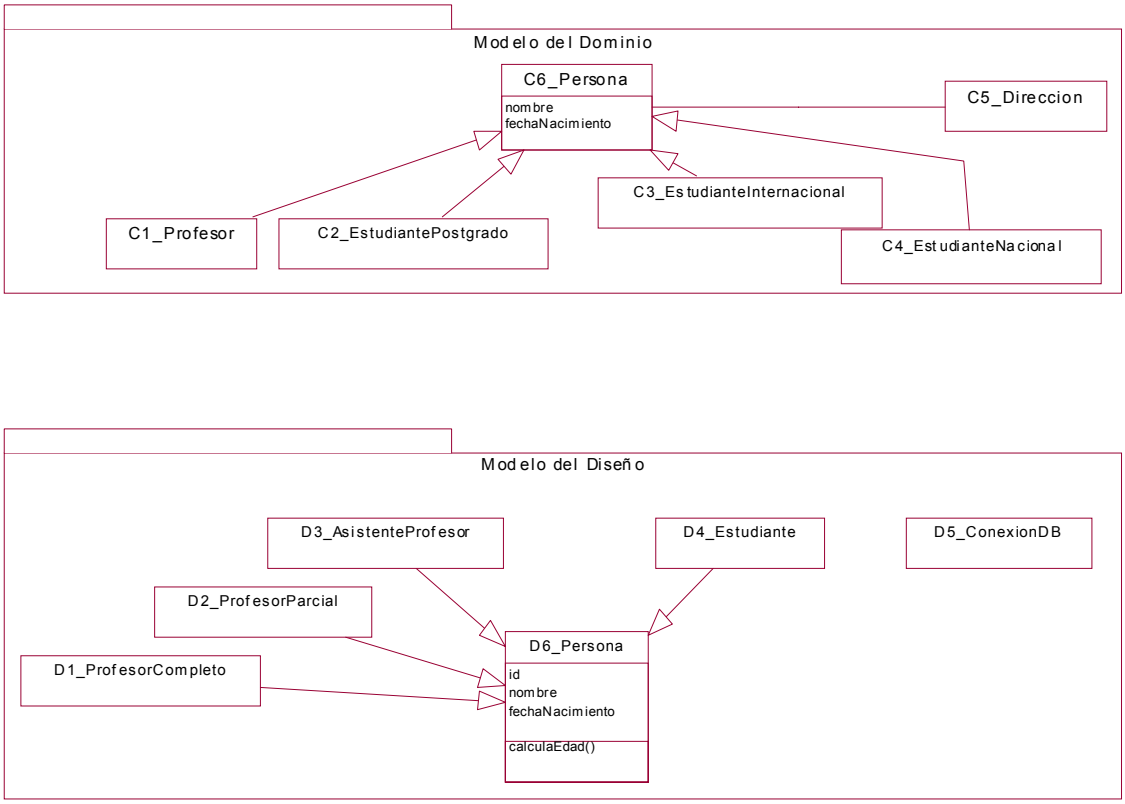


Figura 3.13. Modelo del Dominio y Modelo del Diseño de las personas de la facultad

Para distinguir las Clases Conceptuales de las de Diseño, se deberán colocar dentro del paquete del Modelo de Dominio y del Modelo de Diseño, respectivamente como se muestra en la Figura 3.13. Otra forma posible (y redundante) para esta distinción, es la agregación de estereotipos <<concept>> y <<design>>, a las clases del Modelo del Dominio y a las del Modelo del Diseño, respectivamente. En la Tabla 3.8, se explicarán algunos detalles de cada clase.

Tabla 3.8. Breve documentación de las clases del ejemplo de las personas de una facultad

Clases Conceptuales	**Documentación**
C1_Profesor	La distinción entre los tipos de profesores se puede establecer mediante atributos como horas semanales.
C2_EstudiantePostgrado	La concepción de un estudiante de postgrado se encuentra separada de un estudiante de licenciatura. En parte, por la poca cantidad de estudiantes de postgrado y la atención personalizada, en comparación con los estudiantes de licenciatura.
C3_EstudianteInternacional	Requiere información adicional sobre documentos migratorios
C4_ EstudianteNacional	Se puede disponer de esta información para conectarla con otras bases de datos gubernamentales
C5_Direccion	Se desea tener al menos una dirección de cada persona.
C6_Persona	Se cree superflua la creación de la clase “Student” pues heredaría completamente los atributos de las personas.
Clases de Diseño	**Documentación**
D1_ProfesorCompleto	Profesor de tiempo completo. Cubre 40 horas semanales
D2_ProfesorParcial	Profesor de tiempo parcial. A lo más, cubre 20 horas semanales
D3_AsistenteProfesor	Sólo a los estudiantes de postgrado se les permite impartir ayudantías. Dado que reciben un tipo de beca, no se les considera como empleados, por lo que son diferentes a los profesores de tiempo parcial.
D4_Estudiante	Estudiante de licenciatura. No incluye a los estudiantes de postgrado, pues los miembros de las clases son distintos.
D5_ConexionDB	Conexión a la base de datos. Clase de librería para la comunicación con la base de datos.
D6_Persona	El identificador (ID) de la persona es una llave primaria, usual en las tablas. Una Clase de Diseño puede tener métodos.

Capítulo 4. Localizando las Discrepancias

En nuestro contexto, se referirá a la determinación de los elementos (e.g. clases de un sistema en particular) de los modelos participantes (e.g. Modelo del Dominio y del Diseño) en la relación de correspondencia. Con especial énfasis en la determinación de las Clases Conceptuales y de Diseño.

4.1 El uso del refinamiento para la correspondencia

Para establecer la correspondencia entre los elementos en UML (paquetes y clases) se eligió el refinamiento con la dirección del Modelado de Dominio al Diseño (ver "Apéndice A. Glosario"). El refinamiento es una dependencia con estereotipo <<refine>> que es una relación de la categoría de abstracción.

[Rumbaugh99] define al refinamiento como:

> "Una relación entre dos versiones de un concepto en diferentes etapas de desarrollo o en diferentes niveles de abstracción. […] En principio, hay una correspondencia desde el concepto menos terminado hacia el concepto más terminado. Esto no significa que la traducción es automática. […] En principio, los cambios a un modelo pueden ser validados contra el otro, con las desviaciones marcadas. En la práctica, las herramientas de hoy no pueden hacer todo esto, aunque algunas correspondencias simples pueden reforzarse."

De esta forma, los refinamientos entre los elementos (paquetes, clases, etc.) nos permiten caracterizar la relación entre las dos Disciplinas.

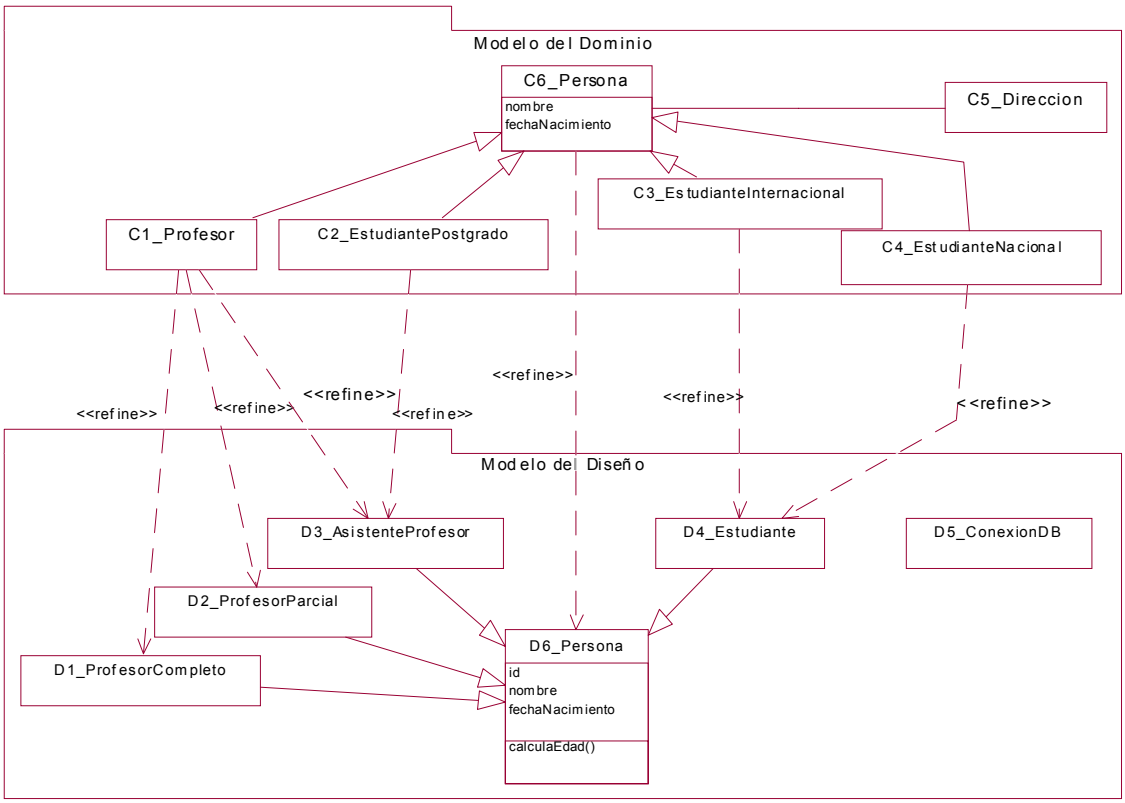
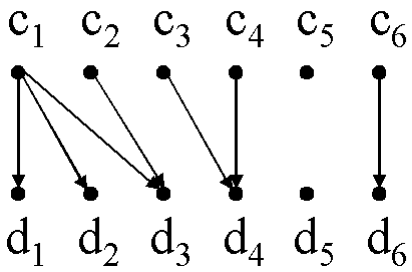


Figura 4.1. Establecimiento de correspondencias mediante los refinamientos entre los Modelos de Dominio y Diseño del ejemplo de la facultad

Si bien el refinamiento es una dependencia estándar, la propuesta radica en el establecimiento de las correspondencias entre los elementos de forma sistemática. Para cada conjunto de elementos que se desee caracterizar su Brecha, se deberá de trazar un refinamiento de todo elemento al elemento más pertinente de la Disciplina siguiente. Este establecimiento se debe realizar según el criterio del verificador (analista y/o diseñador) . El verificador se puede basar en los nombres de las clases, sus atributos y otros elementos que le permitan justificar el establecimiento del refinamiento. Esto se puede apreciar en el ejemplo de la facultad de la Figura 4.1, en donde se refinan los elementos del modelo general (i.e. Clases Conceptuales del Modelo del Dominio) a los elementos del modelo particular (i.e. Clases de Diseño del Modelo de Diseño).

4.2 La Brecha Representacional como un grafo



R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	1	0	0	0
c[2]	0	0	1	0	0	0
c[3]	0	0	0	1	0	0
c[4]	0	0	0	1	0	0
c[5]	0	0	0	0	0	0
c[6]	0	0	0	0	0	1

Figura 4.2. Localización de los refinamientos y los elementos de los Modelos del Dominio y del Diseño para el ejemplo de la facultad

Para solucionar el problema de la localización y la medición de la Brecha, se eligió la representación por un grafo específico, el digrafo bipartito. En este contexto, la localización[28] se referirá a la determinación de los elementos de los modelos participantes en el refinamiento (relación de correspondencia). El ejemplo de la facultad se puede visualizar en la Figura 4.2. La elección se basó en lo siguiente:

- Un grafo permite una definición formal del problema. La formalidad está respaldada por la Teoría de Grafos.
- Un digrafo permite simbolizar la dirección del refinamiento.
- Al ser un grafo bipartito, los refinamientos entre los elementos (e.g. clases, paquetes o atributos) de los modelos se pueden representar mediante una matriz $R \in \{0,1\}^{n_D \times n_C}$, donde

$$r_{i,j} = \begin{cases} 1, & c_i \text{ se refina en } d_j \\ 0, & \text{en cualquier otro caso} \end{cases}$$

  - Los vértices representan a los elementos de los modelos
    - Los vértices iniciales (fuente) constituyen los elementos (e.g. Clases Conceptuales) del primer modelo (e.g. Modelo del Dominio) y se representan con los renglones de la matriz.
    - Los vértices finales (destino) constituyen los elementos (e.g. Clases de Diseño) del segundo modelo (e.g. Modelo del Diseño) y se representan con las columnas de la matriz.
  - Las aristas constituyen los refinamientos entre el elemento del primer modelo con el elemento del segundo modelo. En el ejemplo de la facultad, la Clase Conceptual "C1_Profesor" se refina en la Clase de Diseño "D1_ProfesorCompleto".

Teniendo *m* Clases Conceptuales y *n* Clases de Diseño, la matriz de un grafo bipartito es más pequeña ($m \times n$) que la de una matriz de un grafo normal ($(m+n)\times(m+n)$). Esta

[28] Ver "Apéndice A. Glosario"

última representación se pudiera tener si se buscara la representación redundante entre elementos de un mismo modelo, lo cual no es parte del objetivo.

Capítulo 5. Identificando las Discrepancias

La Ontología es la rama de la Filosofía que trata con las teorías acerca de la naturaleza de las cosas en general, en oposición a las teorías sobre las cosas en particular. Por ello, la Teoría Ontológica se ajusta para modelar las construcciones que representan a los dominios concretos del problema [Opdahl02].

5.1 Discrepancias Ontológicas

En [Opdahl02] se menciona que en al menos ocho referencias se ha empleado el modelo de Bunge-Wand-Weber (BWW), para ver el detalle de las aplicaciones ver "Discrepancias Ontológicas" en la página 52. El importante trabajo sobre métricas del desarrollo Orientado a Objetos de [Chidamber94] del MIT también utiliza el modelo BWW. Este modelo es una adaptación de la ontología de Mario Bunge que se inspira en la teoría de sistemas.

5.1.1 Justificación de las Discrepancias para identificar la Brecha

El trabajo de [Opdahl02] consiste en corresponder los conceptos de la Ontología de BWW con las construcciones de UML.

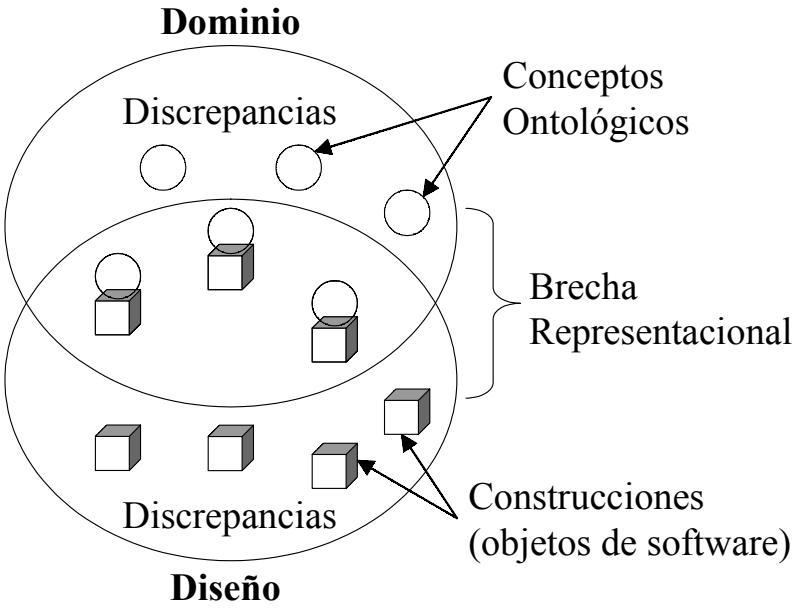


Figura 5.1. Las Discrepancias Ontológicas son útiles para la caracterización de la Brecha Representacional

Dado que las Discrepancias Ontológicas caracterizan las diferencias entre los Conceptos Ontológicos y las Construcciones [Opdahl02], se pueden emplear para caracterizar la Brecha Representacional pues es el intervalo entre el modelo del problema (i.e. conceptualización) y el modelo de la solución (i.e. objetos de software) [Larman01]. En

el Proceso Unificado, el problema se representa con el Modelo del Dominio y la solución con el Modelo del Diseño. Esto se puede ilustrar con la Figura 5.1.

5.1.2 Tipos de Discrepancias problemáticos

Los tipos de Discrepancias (algunos con subclases) que [Opdahl02] propone son:

- Redundancia
  - De subtipo
  - Genuina (problemática)
- Exceso
  - No orientado al dominio del problema
  - Genuino (problemático)
- Sobrecarga (problemática)
- Déficit (problemático)

Cabe mencionar que para saber diferenciar entre las subclases se requiere del modelador y/o verificador, pues se requiere de un criterio subjetivo. Las subclases que usualmente son problemáticas son las genuinas. Por lo que los tipos que no necesariamente son problemáticos son la Redundancia de subtipo y el Exceso no orientado al dominio del problema. En resumen, a menos que alguien justifique o abogue a favor de las Discrepancias, usualmente son problemas que hay que eliminar.

5.1.3 Mejoras propuestas al modelo BWW

Al analizar y experimentar con las Discrepancias para caracterizar la Brecha Representacional, se apreciaron algunas áreas de oportunidad en el modelo BWW, en las que se realizaron propuestas concretas:

- Falta de una formulación matemática.
  - Esta formulación debería permitir la localización, identificación y medición de las diferencias entre los modelos. En el pseudocódigo del algoritmo propuesto, se ofrece una definición más precisa para su caracterización.
- Similaridad entre los tipos de Discrepancias.
  - Se percibió la relación interna entre los tipos de Discrepancias, por lo que se propone una categorización, que se comentará en “Categorías de Discrepancias”.
- Unión de las Discrepancias.
  - Se percibió el fenómeno de que la Redundancia y la Sobrecarga (Discrepancias Representadas) se podían existir al mismo tiempo.
- Complementariedad de las Discrepancias.
  - Se percibió que al fijar el número de entidades modeladas y al provocar una Discrepancia de un tipo, surgía o se eliminaba otra Discrepancia.

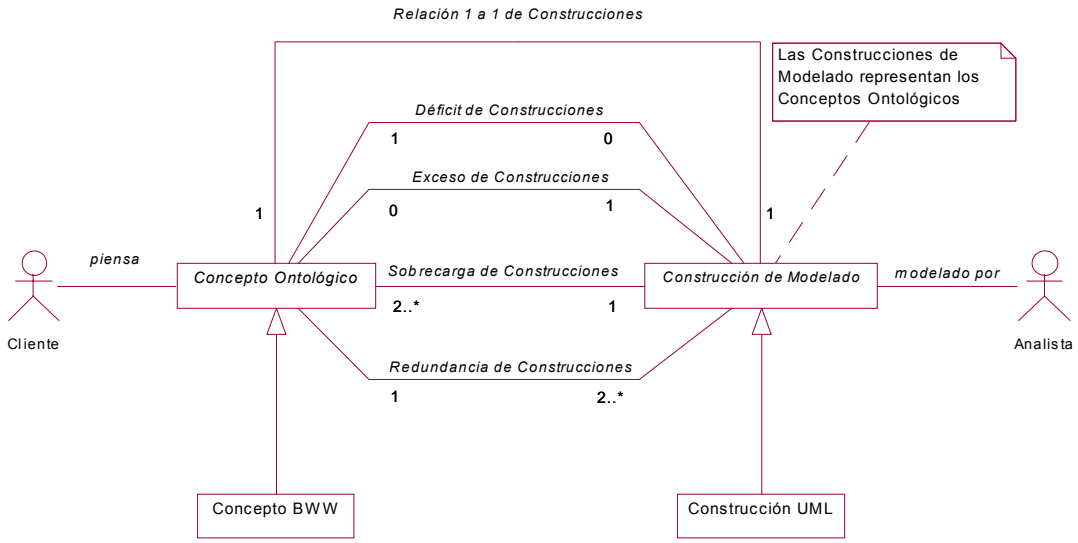


Figura 5.2. Apreciación personal de las Discrepancias Ontológicas del modelo BWW

En la Figura 5.2, se ilustran la apreciación de las Discrepancias Ontológicas del modelo BWW aplicadas por [Opdahl02] a la representación de los Conceptos BWW mediante las Construcciones de Modelado UML. El cliente tiene un conjunto de Conceptos Ontológicos, que el modelo BWW los organiza en 47 Conceptos Ontológicos y que [Opdahl02] los compara de forma sistemática con los 216 Construcciones del Modelado de UML. El analista modela el mundo que le presenta el cliente con las Construcciones de Modelado.

Cabe notar que los Conceptos Ontológicos y las Construcciones del Modelado, son clases abstractas; las metodologías son las que tienen las clases concretas (e.g. BWW y UML). En la transmisión de las representaciones pueden presentarse cinco situaciones: la relación uno a uno (“match”) y las cuatro Discrepancias del modelo BWW.

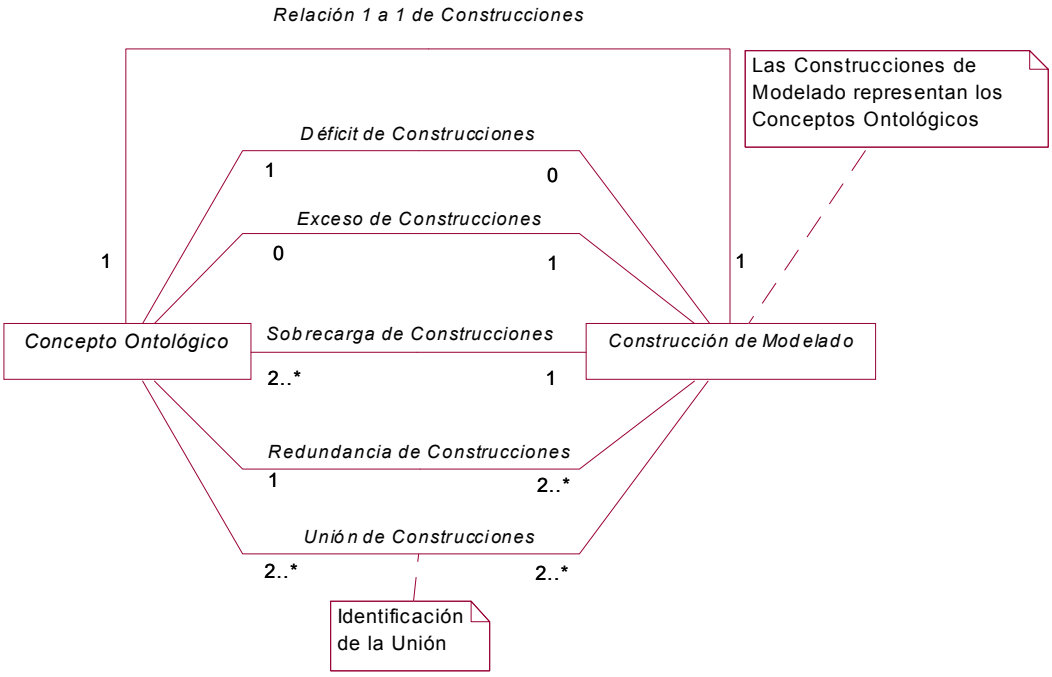


Figura 5.3. Detección de la Unión de la Redundancia y la Sobrecarga de Construcciones en el modelo BWW

La modificación al modelo de BWW es la inclusión de la "Unión de Construcciones" ("Construct Union") para resaltar el fenómeno de la presencia simultánea de las Discrepancias de Redundancia y Sobrecarga. Esta inclusión se muestra en la Figura 5.3. Entonces, se presentan cinco percepciones de las Discrepancias: Déficit, Exceso, Sobrecarga ("Overload"), Redundancia y Unión. Se les llamará "Discrepancias del Modelo" a los resultados de las Discrepancias en conjunto. Para recordarlas más fácilmente, nos referiremos a su acrónimo "DEORUM"[29] por sus siglas en inglés.

El marco conceptual de la propuesta del presente trabajo permite generalizar la caracterización de la Brecha Representacional entre dos modelos distintos pero con elementos de modelado similares.

[29] "Deorum" significa "dioses" en latín. Las Discrepancias detectadas por Wand y Weber [Opdahl02] serían "deor" (quitando la unión y el modelo) que derivó en "deer", "venado" en inglés; o "animal, bestia" en inglés antiguo. Se escogió este mneumónico para recordar las ecuaciones que se deben de cumplir para la asignación de pesos.

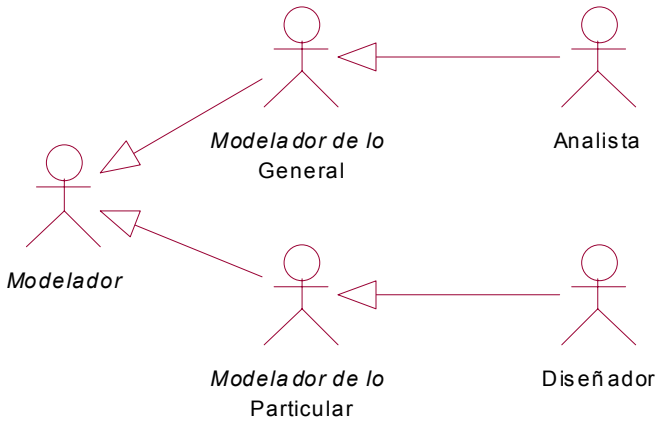


Figura 5.4. Relación entre los modeladores

El Modelador, el Modelador de lo General y el Modelador de lo Particular son actores abstractos (en itálicas en la Figura 5.4) hasta que se elija el modelo correspondiente. En este trabajo, se eligió que el Analista fuera el Modelador del Modelo General que modela el Modelo del Dominio. Haciendo una analogía con el modelo BWW (ver Figura 5.2), el Cliente sería el Modelador del Modelo General; el Analista, el Modelador del Modelo Particular; el Concepto Ontológico, el Elemento del Modelo General y la Construcción del Modelado sería el Elemento del Modelo Particular.

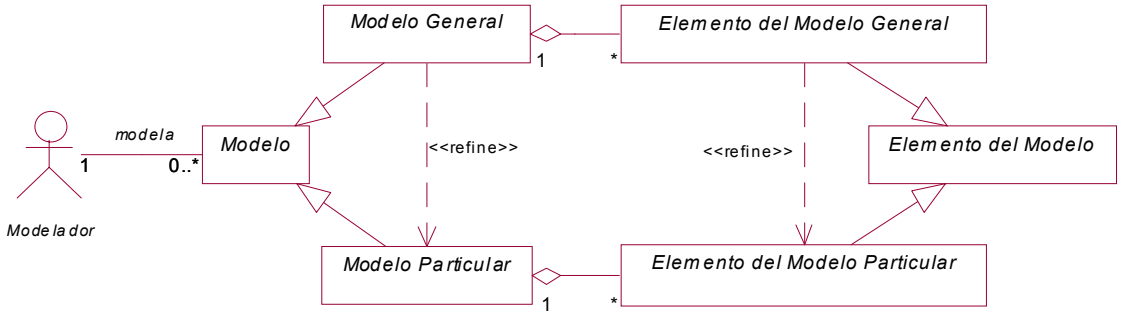


Figura 5.5. Marco conceptual (generalización) de la propuesta

En la Figura 5.5, se muestra la posibilidad de establecer las correspondencias de refinamiento entre los elementos de un modelo; de una entidad poco refinada (i.e. general) a una más refinada (i.e. particular). Cabe notar que todas las clases son abstractas pues es la generalización de los modelos seleccionados (i.e. Modelo del Dominio y Modelo del Diseño) que serían las clases concretas. Cabe recordar que el Analista modela el Modelo del Dominio, así como el Diseñador modela el Modelo del Diseño.

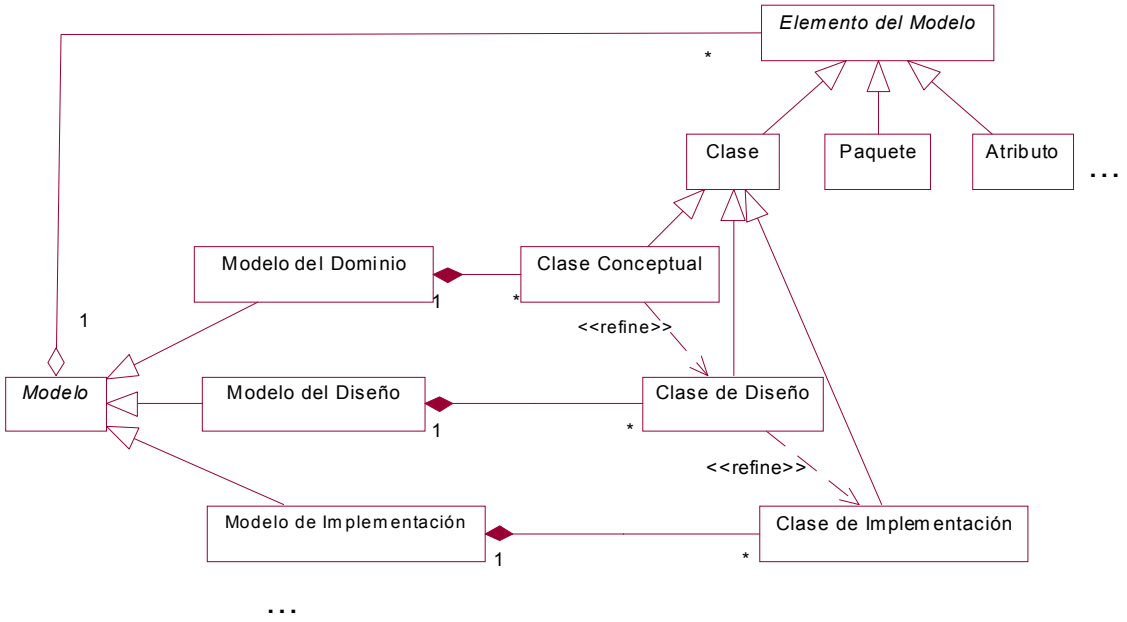


Figura 5.6. Ejemplificación del metamodelo de refinamiento para establecer las correspondencias

En la Figura 5.6, se muestra la composición de varios modelos, por sus elementos. Cabe destacar que es una agregación en el caso general, mientras que para las clases concretas se tiene una composición (agregación por valor). Por razones de legibilidad, sólo se ejemplificó el caso de las clases pertenecientes a varios modelos, pero el cruzamiento entre los modelos (de Dominio, Diseño e Implementación) aplica de forma similar para los paquetes y atributos.

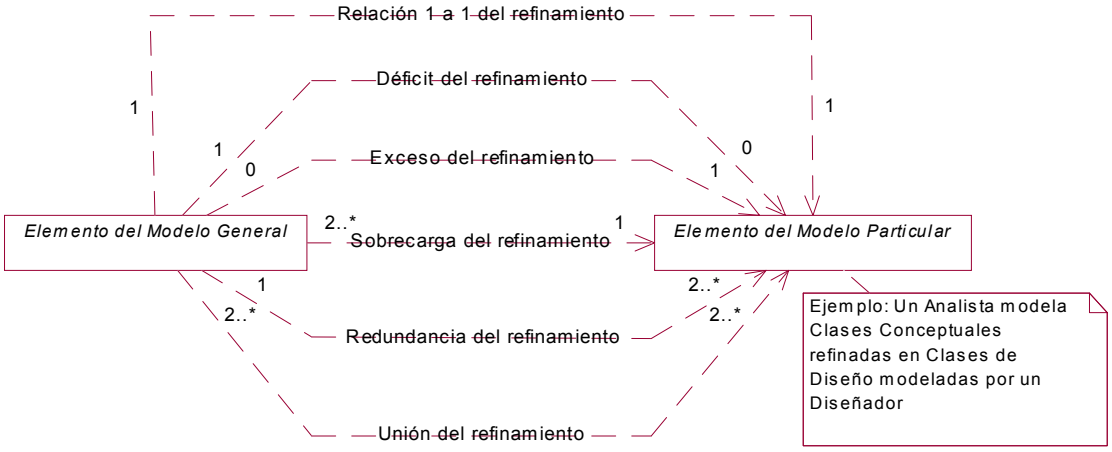


Figura 5.7. Metamodelo del refinamiento para la correspondencia

En particular, de la Figura 5.4, Figura 5.6 y Figura 5.7, se puede decir que un analista modela el problema con varias Clases Conceptuales; mientras que un diseñador plantea una solución con varias Clases de Diseño. La correspondencia entre las Clases Conceptuales y las Clases de Diseño puede establecerse mediante refinamientos uno a uno, o mediante las Discrepancias mencionadas anteriormente ("DEORUM"). La Discrepancia de Refinamiento es el tipo particular de la Discrepancia Ontológica, por lo que, en esencia, significan lo mismo.

Dicho de otra forma, se toma al Modelo del Dominio para ser refinado con el Modelo de Diseño. El Modelo del Dominio se compone de Clases Conceptuales (entre otros elementos de modelado como paquetes y atributos); mientras que el Modelo del Diseño, se forma de Clases de Diseño, que pueden representar mediante el refinamiento a las Clases Conceptuales.

Por lo tanto, la analogía del modelo BWW, así como la generalización al Modelo General y Particular, sientan las bases teóricas para poder caracterizar la correspondencia entre dos modelos de Disciplinas consecutivas.

5.1.4 Ejemplo: identificando las discrepancias con el modelo de la facultad

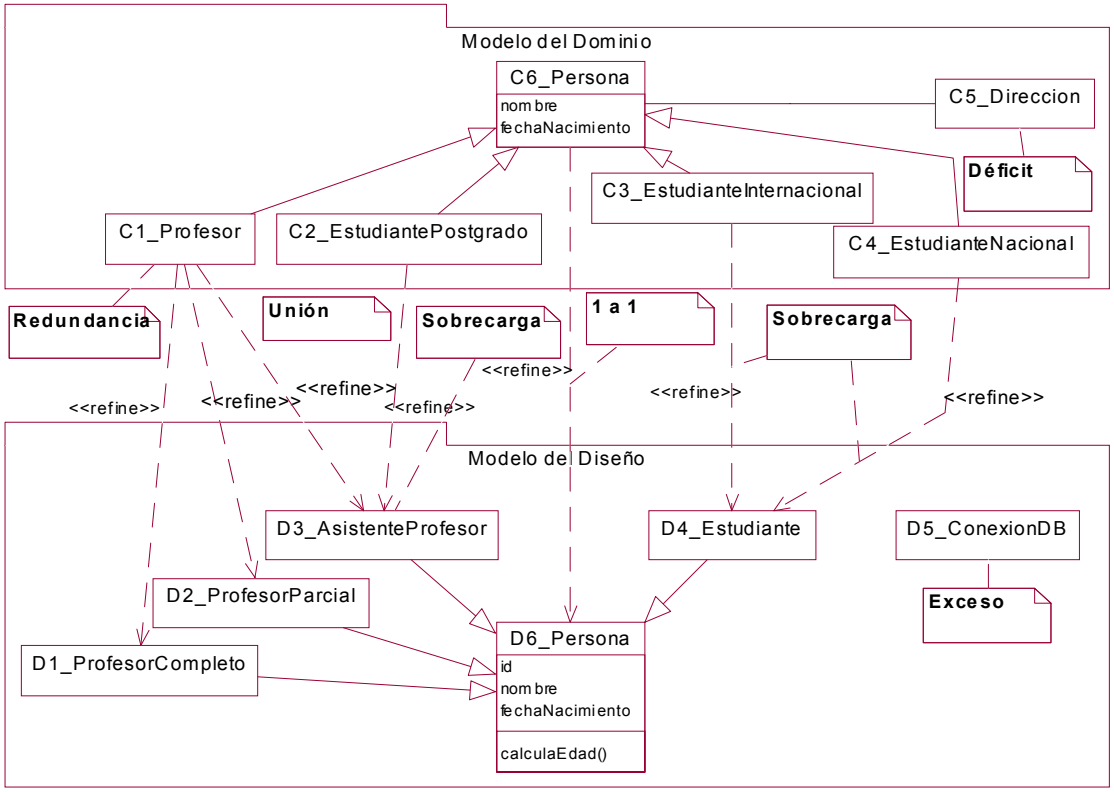


Figura 5.8. Identificación de Discrepancias entre el Modelo de Dominio y el Modelo de Diseño para el ejemplo de la facultad

En la Figura 5.8, se ilustran los tipos de Discrepancias entre el Modelo de Dominio y el Modelo de Diseño.

Para un modelo sencillo como el del ejemplo de la facultad, es fácil reconocer visualmente las Discrepancias. Sin embargo, conforme el sistema crezca, el número de clases y de refinamientos posiblemente se incrementarán, complicando la identificación de las Discrepancias. Es por ello que se requiere de un análisis sistemático de la Brecha.

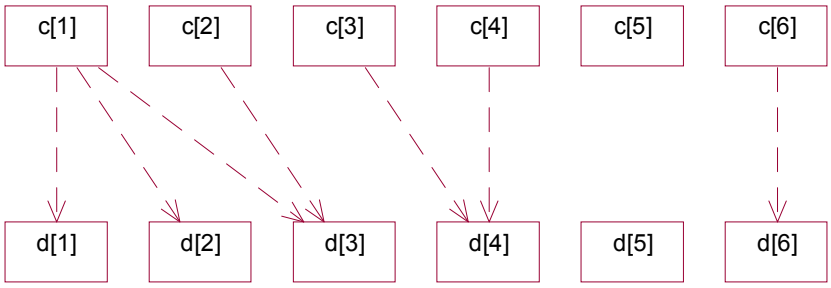


Figura 5.9. Diagrama de clases simplificado del ejemplo de la facultad

Para simplificar la Figura 5.8, se muestra el diagrama de clases de la Figura 5.9, en el que se omiten los nombres largos de las clases, los estereotipos de refinamiento y los paquetes (las clases *c[i]* pertenecen al Modelo del Dominio y las *d[j]*, al Modelo del Diseño).

Tabla 5.1. Grados de las Clases Conceptuales y de Diseño del ejemplo de la facultad. Entre paréntesis, se tiene el orden de la Discrepancia

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**Grado (Suma)**	**Red, Def**
c[1]	1	1	1	0	0	0	**3**	Red(3)
c[2]	0	0	1	0	0	0	**1**	Leaf
c[3]	0	0	0	1	0	0	**1**	Leaf
c[4]	0	0	0	1	0	0	**1**	Leaf
c[5]	0	0	0	0	0	0	**0**	Def
c[6]	0	0	0	0	0	1	**1**	Leaf
Grado (Suma)	**1**	**1**	**2**	**2**	**0**	**1**		
Over, Exc	Leaf	Leaf	Over(2)	Over(2)	Exc	Leaf		

En la Tabla 5.1, se localizan las Discrepancias mediante la representación matricial. Esto, con el fin de identificar su tipo y orden (ver "joinStrDisc" en "Algoritmo (pseudocódigo)", página 110). Para ello, se calculan los grados de los elementos (vértices) del Modelo del Dominio y el Modelo del Diseño del ejemplo de la facultad, mediante la suma de los renglones y las columnas. Los elementos seleccionados fueron las clases. Se pueden presentar varias situaciones:

- Se tiene un par (relación uno a uno, "1to1"), si el grado de una Clase Conceptual (vértice o elemento del Dominio) y el grado de una Clase de Dominio son iguales a uno.
- Se tiene una Redundancia ("Red"), si el grado de una Clase Conceptual es mayor a uno.
- Se tiene una Sobrecarga ("Over"), si el grado de una Clase de Diseño es mayor a uno.

- El orden de la Redundancia y la Sobrecarga es el grado de la clase correspondiente, es decir, el número de arcos adyacentes. Se pueden presentar una Redundancia y una Sobrecarga en una clase, en cuyo caso, se tiene un orden para cada Discrepancia. Dado que sólo hay un grado por vértice, se vio la necesidad de otro nombre, optándose por "orden de la Discrepancia" para que pudiera haber dos órdenes por arco.
- Se tiene un Déficit ("Def"), si el grado de una Clase de Dominio es menor a uno.
- Se tiene un Exceso ("Exc"), si el grado de una Clase Conceptual es menor a uno.
- El grado igual a uno, no contribuye a determinar el tipo de Discrepancia y se optó por llamarlo "Hoja" ("Leaf").

Para la colocación de etiquetas, se concatena el tipo y el orden de cada renglón y cada columna donde haya un arco. Para los Déficit y Excesos se añade un renglón y columna a la matriz de etiquetas para localizar la clase con estas Discrepancias. Como resultado, se obtiene la matriz de etiquetas de la Tabla 5.2.

Tabla 5.2. Identificación de las Discrepancias[30], matriz de etiquetas del ejemplo de la facultad

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**Def**
c[1]	R3	R3	O2R3	-	-	-	-
c[2]	-	-	O2	-	-	-	-
c[3]	-	-	-	O2	-	-	-
c[4]	-	-	-	O2	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
Exc	-	-	-	-	E	-	

[30] Se tomaron las iniciales de las Discrepancias y se retiraron los paréntesis para mayor claridad en la matriz.

5.1.5 Categorías propuestas de Discrepancias

Se proponen cuatro categorías complementarias entre sí, que agrupan a los tipos de Discrepancias:

- Discrepancia Representada
  - Redundancia y Sobrecarga.
  - En el caso en el que se presentan ambas en uno o varios elementos (i.e. un elemento tiene Redundancia y Sobrecarga), se tiene una "Unión" de Discrepancias y a los elementos que tienen ambas, se les denomina "Cruces". El conjunto de todos los arcos que participan en estas Discrepancias es lo que forma una unión.
- Discrepancia No Representada
  - Déficit y Exceso.
- Discrepancia Abundante en lo General (o en el Dominio)
  - Sobrecarga y Déficit.
  - Usualmente problemática.
- Discrepancia Abundante en lo Particular (o en el Diseño)
  - Redundancia y Exceso.
  - De problemática condicionada. Esto es, usualmente hay Discrepancias que tienen una justificación.

Los términos de las categorías: "Abundante en lo General" y "Abundante en lo Particular", son las generalizaciones del presente caso. Si se desea hacer referencia al nombre del modelo en turno, en el presente caso se puede usar: "Abundante en el Dominio" y "Abundante en el Diseño". Se eligieron los términos más genéricos pues puede presentarse la Brecha entre otros modelos (e.g. "Abundante en el Diseño" y "Abundante en la Implementación"). Otros sinónimos independientes del modelo podrían ser "Abundante en el Modelo por Refinar" y "Abundante en el Modelo Refinado". Lógicamente, el encargado del modelo general, es el modelador de lo general, de igual forma aplica para el modelo particular. En la Tabla 5.3 se resumen las categorías propuestas de las Discrepancias.

Tabla 5.3. Resumen de los tipos y categorías de las Discrepancias

Discrepancias Ontológicas		**¿Hay representación en la contraparte del Modelo?**	
		Sí	No
Más elementos en el Modelo del:	**"Categoría" (Problemática)**	"Representada"	"No representada"
Dominio	"Abundante en lo General" (Usualmente problemática)	Sobrecarga	Déficit
Diseño	"Abundante en lo Particular" (Problemática condicionada)	Redundancia	Exceso

Cabe aclarar que los tipos son: Redundancia, Sobrecarga, Déficit y Exceso; mientras que las categorías son: Representada, No Representada, Abundante en el Dominio y Abundante en el Diseño.

5.1.6 Discrepancias Complementarias

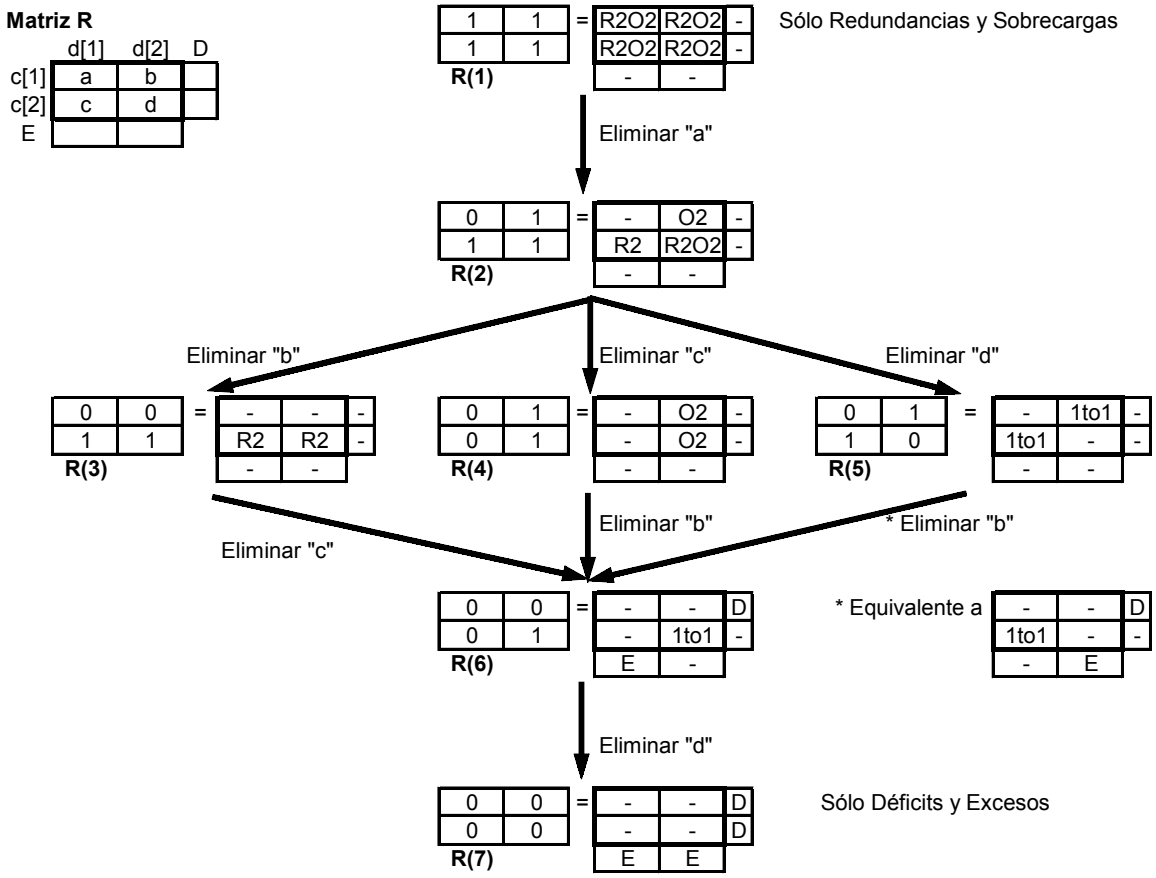


Figura 5.10. Discrepancias complementarias en el caso 2x2.

Se apreció el fenómeno de que las Discrepancias son complementarias. Esto no es mencionado en el trabajo de [Opdahl02]. Para ilustrarlo, se puede tener la siguiente situación extrema. Un modelo repleto de Discrepancias sólo puede tener Discrepancias Representadas o No Representadas. Dicho de otra manera, si se tienen todas las Redundancias posibles de mayor orden (un grafo bipartito completo o una matriz de unos), se tienen todas las Sobrecargas posibles de mayor orden (ver Figura 5.10, matriz R(1)), pero ningún Déficit o Exceso; y viceversa (ver Figura 5.10, matriz R(7)).

Ahora bien, si se fija el número de vértices de ambos conjuntos (elementos de ambos modelos), al eliminar un arco, se pueden eliminar o provocar una o varias Discrepancias. En la Figura 5.10, se puede apreciar la complementariedad de las Discrepancias para el caso 2x2. Se puede ver que al eliminar el arco "a" (en la transición de la matriz R(1) a la matriz R(2)), se elimina una Redundancia y una Sobrecarga, lo cual parece estrechar la

Brecha. En el caso contrario se tiene que al eliminar “c” (de R(3) a R(6)), se elimina una Redundancia, formándose un Déficit y un Exceso, lo cual parece ampliar la Brecha.

Debido a que las Discrepancias son complementarias, los pesos normalizados se necesitan ajustar multiplicándolos por dos, para que el rango de la métrica quede normalizado.

5.2 Problemas y soluciones propuestas por elemento

5.2.1 Paquetes

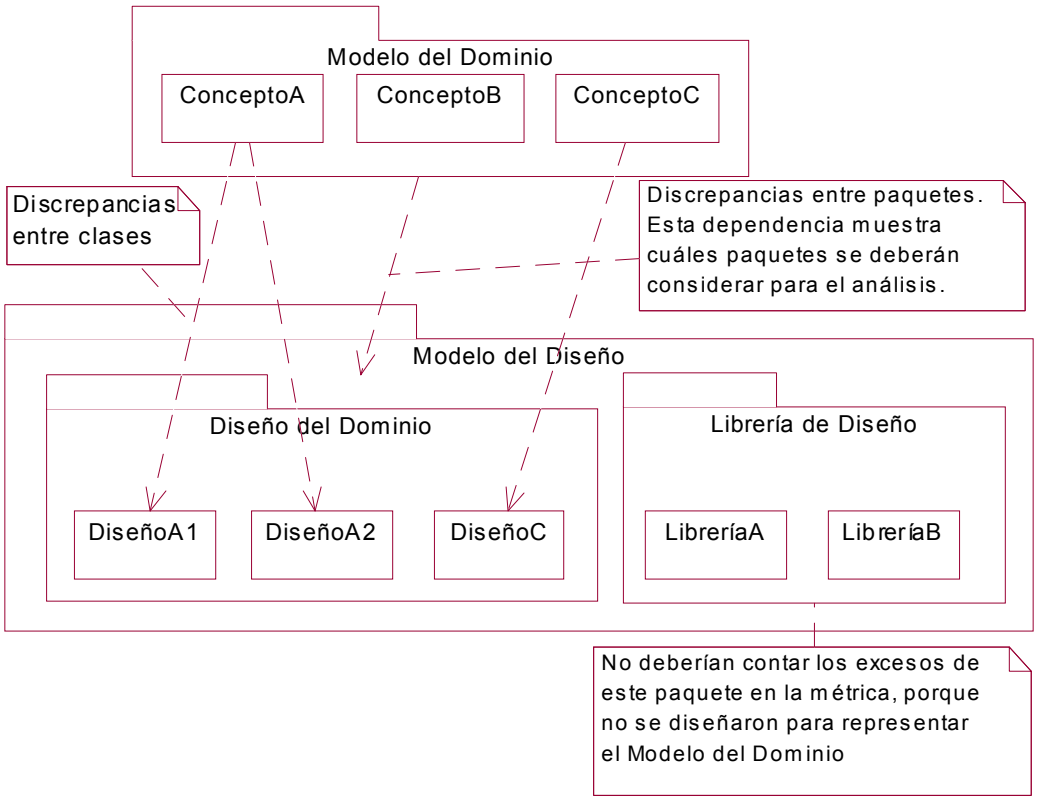


Figura 5.11. Los refinamientos entre paquetes pueden ayudar a reducir los Excesos

Si se agrupan las clases por paquetes, se pueden reducir los Excesos (e.g. de las librerías), como se aprecia en la Figura 5.11.

Si bien, aislar conjuntos de Discrepancias por paquetes ayuda a tener una localización más precisa, puede resultar confusa cuando se quiera establecer correspondencias entre paquetes de distinto nivel. Aunque, intuitivamente se piensa que los anidamientos de paquetes pueden ser infrecuentes, se sugieren las siguientes reglas:

- Establecer correspondencias entre los paquetes más internos con al menos una clase[31].
- Unir todos los paquetes anidados con al menos una clase

[31] No se emplea el término “no vacíos” pues se pudiera tener otro paquete sin clases.

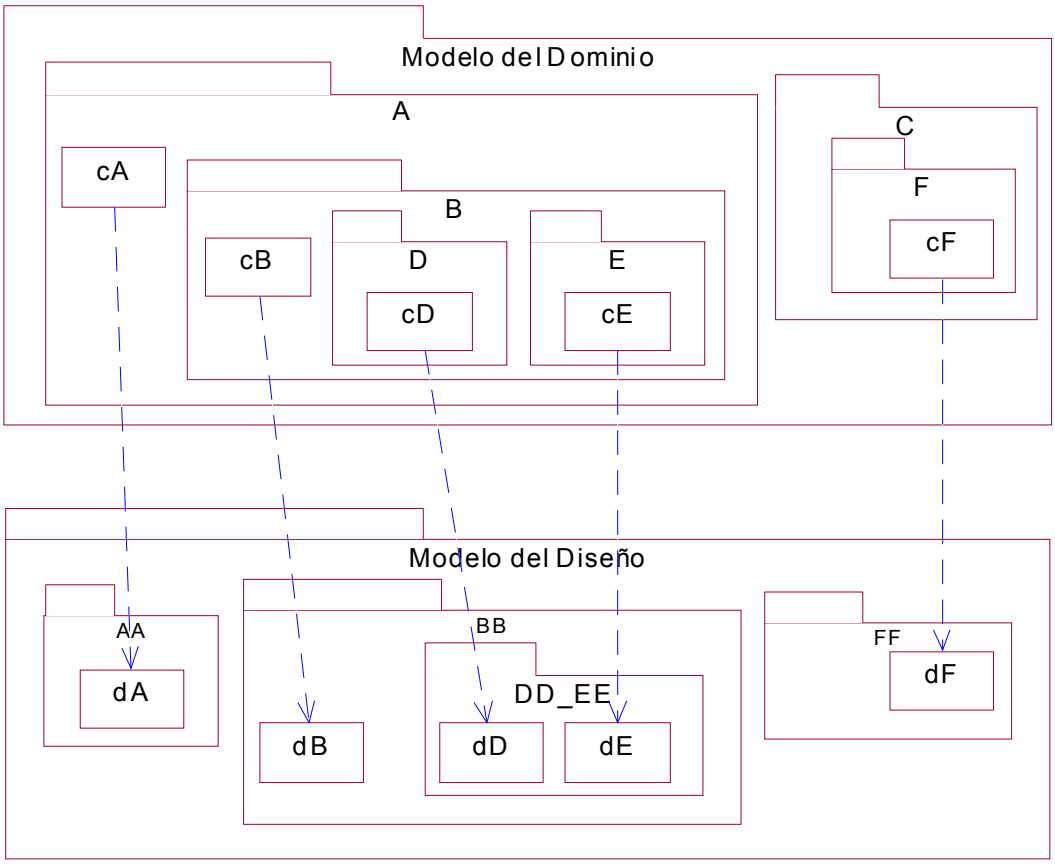


Figura 5.12. Refinamientos entre clases en paquetes anidados

En la Figura 5.12, se establecen las correspondencias entre las clases. Como se aprecia, sólo hay relaciones uno a uno entre las Clases Conceptuales y las Clases de Diseño pues el propósito es mostrar la correspondencia anidada entre los paquetes.

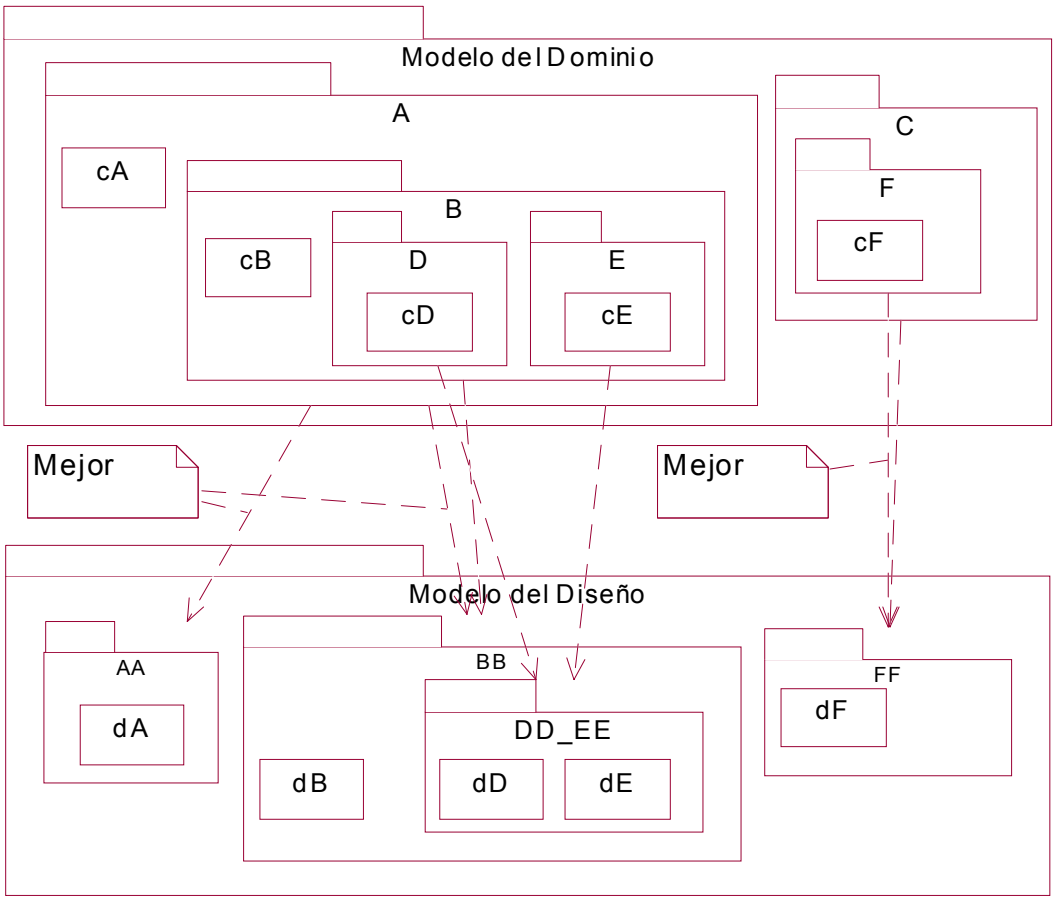


Figura 5.13. Refinamientos entre paquetes anidados

En la Figura 5.13, se establecen algunas correspondencias lógicas y posibles entre los paquetes anidados. Las tuplas entre paréntesis se refieren a los refinamientos entre el primer elemento (que pertenece al Modelo General o Modelo del Dominio), y el segundo (del Modelo Particular o Modelo del Diseño). Aplicando un poco de sentido común y las dos reglas mencionadas, se puede decir que:

- El refinamiento (A, AA) parece lógico y sería directo, si no tuviera el paquete B.
- El refinamiento (B, BB) parece lógico por tener (cB, dB).
- Los refinamientos (D, DD_EE) y (E, DD_EE) parecen lógicos por tener (cD, dD) y (cE, cE).
- El refinamiento (A, BB) es mejor que (B, BB), (D, DD_EE) y (E, DD_EE) porque une los paquetes anidados.
- El refinamiento (C, FF) parece lógico pues son los paquetes más externos dentro de sus modelos y por tener (cF, dF).
- El refinamiento (F, FF) es mejor, pues además de tener (cF, dF), es el paquete más interno con al menos una clase.

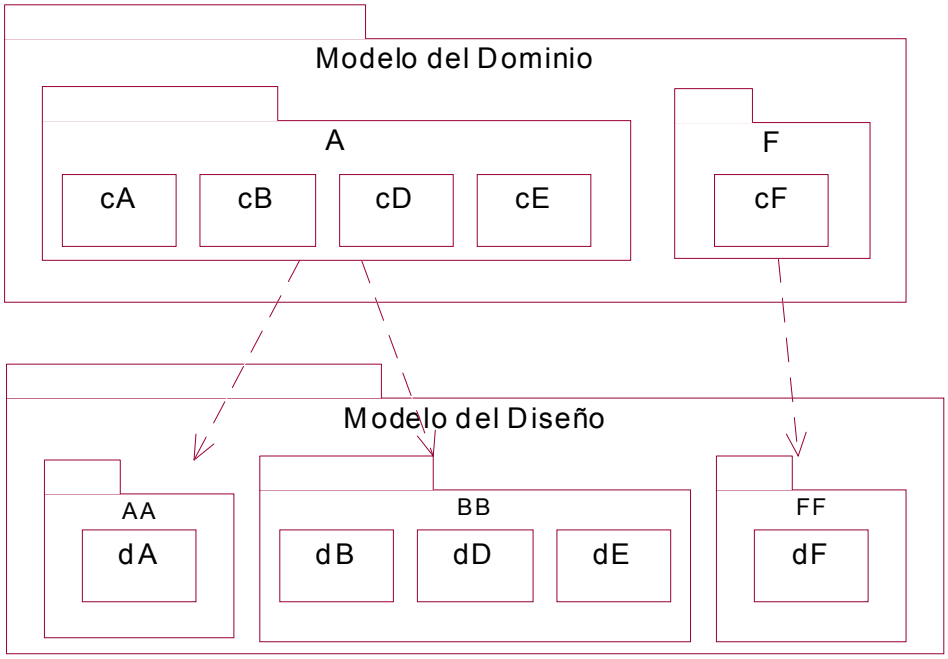


Figura 5.14. Equivalencia de los paquetes anidados

Por lo tanto, se recomienda dejar los refinamientos entre paquetes (A, AA), (A, BB) y (F, FF). Un problema similar se puede encontrar en el caso de matrices ("Matrix2D" refinado en "abstractmatrix" y "matrixdouble"). En la Figura 5.14, se muestra el diagrama de clases equivalente.

5.2.2 Atributos

Atributo por omisión: ObjectId

Para establecer las correspondencias al nivel de atributos, se vio la necesidad de agregar el atributo por omisión "(ObjectId)" para resolver los siguientes problemas:

- Cuando se quería establecer la correspondencia con una clase sin atributos.
  - Por ejemplo, las clases de librerías de funciones no tienen atributos.
  - Ver caso de matrices (Clases Conceptuales de "Function2D" con algunas Clases de Diseño heredadas de "MatrixDouble" como "Arithmetic" con "BandedMatrix", página 137) y caso de escolar (Clase Conceptual Profesor sin atributos, página 153).
- Cuando los atributos de las dos clases se refieren a aspectos diferentes.
  - En el caso de matrices (ver página 137), el atributo "isInferior" (debería ser "isLowerMatrix") representaba en realidad a dos subclases: "LowerMatrix" y "UpperMatrix". Al sustituirlo por las clases se puede eliminar la lógica tipo "switch-case" y simplificar el código.
  - En el caso de escolar (ver página 153), se tiene la Clase Conceptual Persona con los atributos de fecha de nacimiento y sexo, así como la Clase de Diseño Persona con estatura y peso. Aunque la edad puede estar relacionada con el peso, la fecha de nacimiento no se puede representar

con el peso. Sin embargo, las dos clases representan a una persona, a pesar de tener varias Discrepancias.

- Cuando hay una Discrepancia en la herencia que se quiere representar
  - Por ejemplo, se tienen las Clases Conceptuales CX, CY y CZ (donde CY y CZ heredan de CX), así como las Clases de Diseño DY y DZ, todas sin atributos. Se establecen las correspondencias más obvias (CY, DY), (CZ, DZ) y las heredadas serían (CX, DY), (CX, DZ).
  - En el caso de matrices (ver página 137), la Clase de Diseño sin herencia "DiagonalPos" tenía "index", "value" y "size". Estos atributos se eliminaron de la clase al ser heredados de "MatrixDouble".

5.2.3 Relaciones

Es posible que algunas relaciones no se puedan implementar para ciertas tecnologías. Como en el caso escolar, donde la agregación no tiene un sinónimo directo para ciertos manejadores de bases de datos (aunque se pudiera simular con una tabla intermedia).

Herencia

Como se menciona en la justificación del atributo "ObjectId", se debe prestar atención a las herencias que no son similares en ambos modelos, sobretodo en los casos donde haya herencia múltiple. Una forma de solucionarlo sería tomar en cuenta a los atributos de las clases padres en las clases hijos. Esto es, cambiar la reutilización de la herencia por una redundancia de atributos.

5.2.4 Ventajas y desventajas de los refinamientos entre los elementos

En la Tabla 5.4, se resumen algunas observaciones sobre la Brecha para cada tipo de elemento. La información de esta tabla se fue recopilando mientras se modelaba el "Caso de estudio escolar: un sistema de administración escolar", página 153.

Tabla 5.4. Ventajas y desventajas de los refinamientos entre los elementos

Elementos	Utilidad	Matriz	Ventajas	Desventajas
Clases	Muy alta	Usar dependencias de refinamiento de UML	Si no hubiera paquetes, atributos y asociaciones, se tendría una buena idea de la correspondencia entre los modelos. Con menos trabajo que con los atributos, se tendría una buena idea. Representa a los conceptos y los objetos de software Elemento independiente.	Resolución media. No se detectaría que el número y contenido de los atributos fueran muy distintos entre las Clases Conceptuales y las de Diseño. Esto no necesariamente es malo, pues se esperan pocos atributos en el Modelo del Dominio (Ver ejemplo de [Larman01])
Paquetes	Alta	Usar dependencias de refinamiento de UML	Sencilla, si la agrupación de los Modelos de Dominio y Diseño es similar. Elemento independiente.	Poca resolución
Atributos	Media	Matriz externa a UML	Ayuda al verificador a localizar posibles correspondencia de clases	Si el Modelo del Dominio tiene muy pocos atributos con respecto al Modelo del Diseño, las métricas pueden ser confesas.
Relaciones	Baja	Matriz externa a UML	Las diferencias de los adornos deben ser consideradas pues pueden ser problemáticas.	Localizar las diferencias puede ser muy complejo y difícil de mantener por el número de combinaciones de los adornos (multiplicidad, agregación, navegabilidad, etc.) Elemento dependiente
Herencia	Baja	Matriz externa a UML	Consideración de conceptos y atributos heredados	Elemento dependiente

Capítulo 6. Midiendo las Discrepancias

6.1 Método de medición seleccionado: Conteo Ponderado

Si bien es cierto que medir directamente la transferencia del conocimiento entre el pensamiento del modelador del dominio y el del modelador de la solución es prácticamente imposible [Krueger92]; es posible un enfoque para controlar y medir, en términos concretos, a los refinamientos propuestos entre los elementos de los productos de los modeladores.

Después de probar varios métodos para medir la Brecha Representacional, se eligió el Conteo Ponderado por ser el más intuitivo, sencillo, flexible, que permitía medir la Brecha para cada tipo de Discrepancia y respetaba la secuencia esperada. Se le denominó "Conteo Ponderado" en vez de "promedio ponderado" porque la base es el conteo de las Discrepancias y porque el cálculo es ligeramente distinto al promedio (i.e. se realizan ajustes debidos el fenómeno de complementariedad).

Es posible que en un futuro se encuentren métodos más sofisticados para medir la Brecha. Sin embargo, se eligió un enfoque de sencillez[32] tanto en la comprensión como en el cálculo, para que tuviera mejor aceptación. Por ejemplo, la métrica de complejidad más ampliamente usada (y debatida) es muy simple: $V(G) = E - N + 2$, donde $V(G)$ es la complejidad ciclomática para un grafo de flujo G, con E aristas y N vértices. Esta métrica originalmente fue desarrollada por Thomas McCabe [Pressman01].

De forma resumida, el método del Conteo Ponderado, obtiene la cantidad de Discrepancias por tipo y orden; divide estas cantidades entre el máximo de las Discrepancias posibles; y pondera la importancia de cada tipo de Discrepancia. con un vector de pesos dado (que debe cumplir con ciertas restricciones).

Se recomienda recabar las métricas de la Brecha Representacional, antes de terminar el Diseño de cada iteración. En especial, en las iteraciones de las Fases de Elaboración y Construcción (ver Tabla 2.3), pues es donde se dedica mayor esfuerzo al Modelo del Dominio y al Modelo del Diseño [Larman01]. Esto, con el fin de corregir los defectos antes de su propagación a las Disciplinas e iteraciones siguientes, y con ello, reducir el trabajo y aumentar la calidad del software.

Cabe remarcar la suposición de que los requerimientos no cambian (i.e. son estables) dentro de una iteración. Para manejar estos cambios que son muy comunes, se supone que los cambios se registran de una iteración a otra para mantener la consistencia de los documentos del desarrollo del sistema.

[32] Haciendo referencia al conocido "Keep It Simple" ("KIS")

6.1.1 Casos intuitivos sobre la métrica de la Brecha Representacional

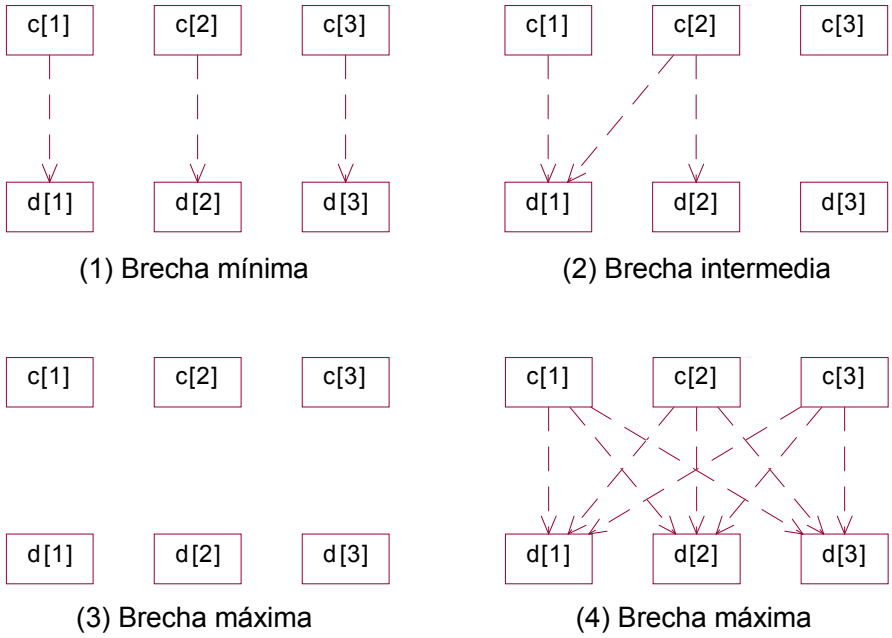


Figura 6.1. Casos intuitivos sobre la métrica de la Brecha Representacional

En la Figura, se muestran cuatro casos para ilustrar los posibles valores de la métrica. Dado que la Brecha es mínima en (1), se esperaría una métrica (normalizada) del modelo con valor cero. Mientras que para las Brechas máximas de (3) y (4), se esperaría una métrica con valor de uno. Estos serían los casos extremos, sin embargo, el problema radica en determinar el valor para las Brechas intermedias como en (2), pues pudiera ser cualquier valor entre cero y uno. El Conteo Ponderado cumple con los casos (1), (3) y (4) y calcula una medida para una Brecha intermedia

6.1.2 Algoritmo (pseudocódigo)

Significado de las variables

Tabla 6.1. Descripción de las variables

Variable	Descripción
C	Conjunto de los elementos del Modelo del Dominio (conceptos)
D	Conjunto de los elementos del Modelo del Diseño
R	Matriz de relaciones de refinamiento entre elementos de similar naturaleza entre el Modelo del Dominio y el Modelo del Diseño
Arc	Arcos del grafo. Refinamientos entre elementos de UML.
X	Matriz de localización de los cruces
M	Matriz de relaciones uno a uno
OD	Conjunto de las Discrepancias Ontológicas
def	Déficit. Discrepancia No Representada.
exc	Exceso. Discrepancia No Representada.
$over$	Sobrecarga (“overload”). Discrepancia Representada.
red	Redundancia. Discrepancia Representada.
uni	Unión de las Discrepancias Representadas
Mod	Discrepancias del modelo, en conjunto
max_b	Máximo número de la Discrepancia b
e_k	Vector de la existencia de la Discrepancia k
o_k	Vector del orden de la Discrepancia k
AW	Vector con los pesos absolutos (“absolute weights”)
W	Vector con los pesos relativos ajustados (“weights”)
$ratios$	Vector con las razones de las Discrepancias
G	Vector con el peso global de las Discrepancias
g_{Mod}	Conteo Ponderado de las Discrepancias del modelo
$cont_b$	Contribución relativa de cada tipo de Discrepancia
S	Matriz de cadenas que identifican las Discrepancias (representación textual)

Declaración de las variables

$C = \{c_i\}, \qquad i = \{1,...,n_C\}$

$D = \{d_j\}, \qquad j = \{1,...,n_D\}$

$R \in \{0,1\}^{n_D \times n_C}$, donde

$$r_{i,j} = \begin{cases} 1, & c_i \text{ se refina en } d_j \\ 0, & \text{en cualquier otro caso} \end{cases}$$

$n_{Arc} = |Arc| \in \{0,...,n_C \cdot n_D\}$

$X \in \{0,1\}^{n_D \times n_C}$

$M \in \{0,1\}^{n_D \times n_C}$

$OD = \{def, exc, over, red, uni\}$

$k \in \{def, exc, over, red\} \subset OD \equiv \{1,...,4\}$ (para cada Discrepancia)

$b, t \in OD \equiv \{1,...,5\}$ (índice)

$v \in \{1,...,max_k\} \equiv \{1,...,n_C\} \vee \{1,...,n_D\}$ (índice)

$e_k \in \{0,1\}^{n_k}$

$o_a \in \{0,2,...,n_a\}^{n_a}$

$a \in \{over, red\} \equiv \{1, 2\}$ (para cada Discrepancia Representada)

$n_C = |C| = |e_{def}| = |e_{red}| = |o_{red}| = max_{def} = max_{red}$

$n_D = |D| = |e_{exc}| = |e_{over}| = |o_{over}| = max_{exc} = max_{over}$

$max_{uni} = n_C \cdot n_D$

$max_{Mod} = n_C + n_D$

$aw \in [0, N]^5$, con una N arbitraria, donde

$aw_1 = aw_{def}$

$aw_2 = aw_{exc}$

$aw_3 = aw_{over}$

$aw_4 = aw_{red}$

$aw_5 = aw_{uni}$

$w \in [0, 1]^5$, (fenómeno de complementos) mismo orden que a aw

$ratios \in [0, 1]^5$, mismo orden que a aw

$g \in [0, 1]^5$, mismo orden que a aw

$cont \in [0, 1]^5$, mismo orden que a aw

$S = \{s_{i,j}\}^{(n_C+1)\times(n_D+1)}$, donde $s_{i,j}$ es una cadena de caracteres

Definición de las funciones

Como convención rotacional, la cardinalidad de un conjunto se representa por $n_A = |A|$, o como la suma de los elementos de una matriz o vector

$$n_B = \sum_{i=1}^{max_B} b_i$$

locDisc

(entradas: R; salidas: $e_{def}, e_{exc}, e_{over}, e_{red}, o_{over}, o_{red}$)

Localización de las Discrepancias

1. Sumar por renglones para encontrar el grado de los vértices

$$r_{i\bullet} = \sum_{j=1}^{n_D} r_{i,j}$$

2. Sumar por columnas para encontrar el grado de los vértices

$$r_{\bullet j} = \sum_{i=1}^{n_C} r_{i,j}$$

3. Marcar los elementos participantes en la Discrepancia

3.1. $e_{def,i} = \begin{cases} 1, & r_{i\bullet} < 1 \quad (\equiv r_{i\bullet} = 0) \\ 0, & \text{en cualquier otro caso} \end{cases}$

3.2. $e_{exc,j} = \begin{cases} 1, & r_{\bullet j} < 1 \quad (\equiv r_{\bullet j} = 0) \\ 0, & \text{en cualquier otro caso} \end{cases}$

3.3. $e_{over,j} = \begin{cases} 1, & r_{\bullet j} > 1 \quad (\equiv r_{\bullet j} \geq 2) \\ 0, & \text{en cualquier otro caso} \end{cases}$

3.4. $e_{red,i} = \begin{cases} 1, & r_{i\bullet} > 1 \quad (\equiv r_{i\bullet} \geq 2) \\ 0, & \text{en cualquier otro caso} \end{cases}$

4. Registrar el orden de las Discrepancias Representadas

4.1. $o_{over,j} = \begin{cases} r_{\bullet j}, & r_{\bullet j} > 1 \quad (\equiv r_{\bullet j} \geq 2) \\ 0, & \text{en cualquier otro caso} \end{cases}$

4.2. $o_{red,i} = \begin{cases} r_{i\bullet}, & r_{i\bullet} > 1 \quad (\equiv r_{i\bullet} \geq 2) \\ 0, & \text{en cualquier otro caso} \end{cases}$

counDisc

(entradas: R; salidas: $n_{def}, n_{exc}, n_{over}, n_{red}, n_{Mod}$)

Contabilización de las Discrepancias

1. locDisc(R; $e_{def}, e_{exc}, e_{over}, e_{red}, o_{over}, o_{red}$)
2. Calcular la suma de los órdenes de cada Discrepancia Representada

2.1. $o_{over} = \sum_{j=1}^{n_D} o_{over,j}$

2.2. $o_{red} = \sum_{i=1}^{n_C} o_{red,i}$

3. Contar las Discrepancias por tipo

$$n_k = \sum_{v=1}^{max_k} e_{k,v} = e_k \cdot e_k$$

Por ejemplo, para $k = red$, $\quad n_{red} = \sum_{v=1}^{n_C} e_{red,v} = e_{red} \cdot e_{red}$

4. Contar las Discrepancias

$$n_{Mod} = \sum_k n_k = n_{def} + n_{exc} + n_{over} + n_{red}$$

ratDisc

(entradas: R; salidas: $ratio_{def}, ratio_{exc}, ratio_{over}, ratio_{red}, ratio_{Mod}$)

Razones de las Discrepancias

1. counDisc(R; $n_{def}, n_{exc}, n_{over}, n_{red}, n_{Mod}$)
2. Calcular razones por Discrepancias por tipo

$$ratio_k = \frac{n_k}{max_k}$$

Por ejemplo, para $k = over$, $\quad ratio_{over} = \frac{n_{over}}{n_D} = \frac{n_{over}}{max_{over}}$

3. Se ajusta (fenómeno de Discrepancias complementarias) la razón del modelo

$$ratio_{Mod} = \frac{1}{2}\sum_k ratio_k = \frac{1}{2}\left(ratio_{def} + ratio_{exc} + ratio_{over} + ratio_{red}\right)$$

uniDiscSimp

(entradas: R; salidas: $n_{uni}, ratio_{uni}$ (opcional: $X, n_X, M, n_M, r_{\bullet\bullet}$))

Localización, conteo y razón de la unión de las Discrepancias Representadas

1. Número de cruces

$$n_X = \sum_{i=1}^{n_C}\sum_{j=1}^{n_D} x_{i,j}$$

donde, $x_{i,j} = \begin{cases} 1, & e_{red,i} = e_{over,j} = r_{i,j} = 1 \\ 0, & \text{en cualquier otro caso} \end{cases}$

2.

3. Número de relaciones uno a uno

$$n_M = \sum_{i=1}^{n_C}\sum_{j=1}^{n_D} m_{i,j}$$

donde, $m_{i,j} = \begin{cases} 1, & e_{red,i} = e_{over,j} = 0, r_{i,j} = 1 \\ 0, & \text{en cualquier otro caso} \end{cases}$

4. Número de relaciones de las Discrepancias Representadas

$$n_{uni} = r_{\bullet\bullet} - n_M$$

donde, $r_{\bullet\bullet} = \sum_{i=1}^{n_C}\sum_{j=1}^{n_D} r_{i,j}$

5. Razón de la unión de las Discrepancias Representadas

$$ratio_{uni} = \frac{n_{uni}}{max_{uni}}$$

weigDisc

(entradas: R, aw; salidas: g, g_{Mod}, (opcional: w_k))

Peso relativo y global de cada tipo de Discrepancias

1. ratDisc(R; $ratio_{def}, ratio_{exc}, ratio_{over}, ratio_{red}, ratio_{Mod}$)
2. uniDiscSimp(R; $n_{uni}, ratio_{uni}$)
3. Normalizar y ajustar los pesos absolutos (el factor 2 refleja el ajuste por la complementariedad de las Discrepancias)

$$w_b = 2 \cdot \frac{aw_b}{\sum_{t=1}^{5} aw_t}$$

4. Métrica por Discrepancia (peso global)

$$ratios = (ratio_{def}, ratio_{exc}, ratio_{over}, ratio_{red}, ratio_{uni})$$

$$g_b = ratios_b \cdot w_b$$

5. Métrica de las Discrepancias (de todo el modelo)

$$g_{Mod} = \sum_{t=1}^{5} g_t$$

6. Contribución relativa por cada tipo de métrica

$$cont_b = \frac{1}{g_{Mod}} g_b$$

joinStrDisc

(entradas: R; salidas: S)

Identificación de las Discrepancias en representación textual

1. locDisc(R; $e_{def}, e_{exc}, e_{over}, e_{red}, o_{over}, o_{red}$)
2. Para $i = \{1,...,n_C\}$, $j = \{1,...,n_D\}$ (Sobrecarga, Redundancia, Uno a uno)

$$s_{i,j} = \begin{cases} (\text{"O"}+o_{over,j}) + (\text{"R"}+o_{red,i}), & e_{red,i} = 1 \vee e_{over,j} = 1, r_{i,j} = 1 \\ \text{"1tol"}, & e_{red,i} = e_{over,j} = 0, r_{i,j} = 1 \\ \text{"}\varnothing\text{"}, & \text{en cualquier otro caso} \end{cases}$$

3. Para $i = \{1,...,n_C\}$ (Déficit)

$$s_{i,n_D+1} = \begin{cases} \text{"D"}, & e_{def} = 1 \\ \text{"}\varnothing\text{"}, & \text{en cualquier otro caso} \end{cases}$$

4. Para $j = \{1,...,n_D\}$ (Exceso)

$$s_{n_C+1,j} = \begin{cases} \text{"E"}, & e_{exc} = 1 \\ \text{"}\varnothing\text{"}, & \text{en cualquier otro caso} \end{cases}$$

weigMetrSummDisc

(entradas: R, $aw = (20, 10, 18, 9, 3)^T$; salidas: S)

Resume las métricas ponderadas formateándolas en una tabla y agrega los encabezados. Se recomienda obtener estadísticos básicos (media, mínimo y máximo) sobre los órdenes de las Discrepancias Representadas como información para el verificador.

Realiza llamadas a las siguientes funciones:

- ratDisc(R; $ratio_{def}, ratio_{exc}, ratio_{over}, ratio_{red}, ratio_{Mod}$)
- uniDiscSimp(R; $n_{uni}, ratio_{uni}$)
- weigDisc(R, aw; g, g_{Mod}, w_b)

6.1.3 Guías para la asignación de los pesos

Si bien, los pesos permiten flexibilidad para resaltar algún tipo de Discrepancia, plantean la interrogante sobre la asignación de los pesos. Cabe señalar que los pesos pueden variar dependiendo de la aplicación, dominio y experiencia del verificador. Además, el método de Conteo Simple es un caso particular donde los pesos de las Discrepancias son iguales y se excluye a la Unión, es decir,

$$AW = (1, 1, 1, 1, 0)^T \equiv W = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0\right)^T$$

Para mantener la complementariedad de las Discrepancias, se debe cumplir que

$$w_{def} + w_{exc} = w_{over} + w_{red} + w_{uni} = 1 \qquad [1]$$

para preservar el rango de las métricas ponderadas del modelo entre [0, 1].

Aunque la responsabilidad de la asignación de los pesos recae en el verificador, a continuación se enlistan algunas guías, que fueron útiles para los casos de estudios del presente trabajo.

1. Si se quiere resaltar que las Discrepancias Abundantes en lo General (Sobrecarga y Déficit) son más problemáticas que las Abundantes en lo Particular, como lo afirma [Opdahl02], se debería cumplir que

   $$w_{def} > w_{exc}$$

   $$w_{over} > w_{red}$$

   En particular, se sugiere que

   $$w_{def} = \alpha \cdot w_{exc} \qquad [2]$$

   $$w_{over} = \alpha \cdot w_{red} \qquad [3]$$

   con un valor sugerido de $\alpha = 2$, o al menos, que las proporciones

   $$\frac{w_{def}}{w_{exc}} \approx \frac{w_{over}}{w_{red}}$$

   sean similares
2. Si se quiere que la Unión sea un discriminante que penalice (incremente) la métrica debido a las uniones y a los órdenes altos, se recomienda que sea menor a los demás pesos

   $$w_{uni} < \min(w_k), \qquad k \in \{def, exc, over, red\} \qquad [4]$$

   En particular, se sugiere que represente un bajo porcentaje (por ejemplo, 10%) en el término de la derecha (Discrepancias Representadas) en [1]

   $$\frac{w_{uni}}{w_{over} + w_{red} + w_{uni}} = p_{uni} \approx 0.1 \qquad [5]$$

Una solución propuesta para el vector de pesos absolutos y de pesos relativos se muestra en la Tabla 6.2. Cabe recordar que la suma de los elementos de *W* es dos por el ajuste y no la unidad (ver "weigDisc" en la página 110). Esta sugerencia, cumple con [1] y sigue las sugerencias [2], [3], [4] y [5] (el porcentaje de p_{uni} en [5] fue del 10%),

Tabla 6.2. Vectores sugeridos de pesos absolutos y relativos

Discrepancia	*AW*	*W*
Déficit	20	0.67
Exceso	10	0.33
Sobrecarga	18	0.60
Redundancia	9	0.30
Unión	3	0.10
Suma	**60**	**2.00**

6.1.4 Interpretación de la métrica

Uno de los objetivos de las métricas propuestas es confirmar la apreciación de los modeladores sobre el sistema. Es decir, si el analista piensa que la Brecha es estrecha y la

métrica muestra lo contrario, debería localizar las Discrepancias para conocer realmente lo que sucede y realizar los cambios necesarios.

Para dar una idea más concreta, en la Tabla 6.3, se explican las columnas de la salida del programa[33] (ver Figura 6.2) para el ejemplo de la facultad ("Ejem4.dat").

Figura 6.2. Entrada y salida del ejemplo de la facultad[34]

> # Entrada

```
> mR = read.table("D:/Posada/TesisMetricas/AvancesMetricas/MetricInR/DataEjem/Ejem4.dat",
header = TRUE)
> mR
   d1 d2 d3 d4 d5 d6
c1  1  1  1  0  0  0
c2  0  0  1  0  0  0
c3  0  0  0  1  0  0
c4  0  0  0  1  0  0
c5  0  0  0  0  0  0
c6  0  0  0  0  0  1
>
```

> # Salida. Resumen de métricas

```
> mWM = weigMetrSummDisc(mR)
> mWM
     MeanOrder MinOrder MaxOrder Count Max   Ratio AdjRatio
Def          0        0        0     0   7 0.00000  0.00000
Exc          0        0        0     1   8 0.12500  0.06250
Over         2        2        2     1   8 0.12500  0.06250
Red          2        2        2     1   7 0.14286  0.07143
Uni          0        0        0     4  56 0.07143  0.03571
Mod          0        0        0     3   0 0.00000  0.19643
     NormWeight AdjWeight WeightDisc Contribution
Def      0.3333    0.6667   0.000000      0.00000
Exc      0.1667    0.3333   0.041667      0.25000
Over     0.3000    0.6000   0.075000      0.45000
Red      0.1500    0.3000   0.042857      0.25714
Uni      0.0500    0.1000   0.007143      0.04286
Mod      1.0000    2.0000   0.166667      1.00000
>
```

> # Salida. Representación textual

```
> msDisc = joinStrDisc(mR)
> msDisc
    d1   d2   d3     d4   d5  d6      Def
c1  "R3" "R3" "O2R3" "-"  "-" "-"     "-"
c2  "-"  "-"  "O2"   "-"  "-" "-"     "-"
c3  "-"  "-"  "-"    "O2" "-" "-"     "-"
c4  "-"  "-"  "-"    "O2" "-" "-"     "-"
c5  "-"  "-"  "-"    "-"  "-" "-"     "D"
c6  "-"  "-"  "-"    "-"  "-" "1To1" "-"
Exc "-"  "-"  "-"    "-"  "E" "-"     ""
```

[33] Se tradujeron los nombres de los renglones y columnas pero se preservó el orden de la salida del programa.

[34] En el Resumen de las métricas, los ceros significan que no aplica.

Tabla 6.3. Explicación de las columnas del resumen de métricas

Columna	Descripción	Modelo
Tipo	Tipo de la discrepancia.	Se empleó en vez del total de las sumas, por las excepciones.
Orden medio	Promedio de los órdenes de cada tipo de Discrepancia Representada.	No aplica
Orden mínimo	Mínimo de los órdenes de cada tipo de Discrepancia Representada.	No aplica
Orden máximo	Máximo de los órdenes de cada tipo de Discrepancia Representada.	No aplica
Cantidad	Número de Discrepancias en el modelo.	* La cantidad de la Unión se excluye de la suma, porque es un fenómeno derivado de las Discrepancias Representadas.
Máximo	Número posible de Discrepancias fijando el número de elementos de cada modelo (número de renglones y columnas). Hay dos igualdades: entre el Déficit y la Redundancia, y entre el Exceso y la Sobrecarga.	No aplica
Razón	Fracción de la división de la cantidad de Discrepancias entre el máximo.	No aplica (pues el rango sería de [0, 2] por la complementariedad)
Razón ajustada	Dividido entre dos para ajustarlo al rango [0, 1] por la complementariedad. Columna informativa.	* Dividido entre dos para ajustarlo al rango [0, 1] por la complementariedad.
Peso normalizado	Ponderación relativa del vector dado de pesos absolutos. Ver Tabla 6.2, para el vector de pesos por omisión. Columna informativa.	Debe ser 1.
Peso ajustado	Peso normalizado multiplicado por dos por la complementariedad. Columna informativa (parámetro).	* Debe ser 2 por la complementariedad.
Peso de la Discrepancia	Razón multiplicada por el peso ajustado. El ajuste de la métrica es para que el rango sea de [0, 1].	
Contribución	Aportación por tipo de la Discrepancia a la métrica del modelo.	Debe ser 1.

Para ilustrar el significado de las métricas propuestas, se emplearán los resultados del problema en la Tabla 6.4.

Tabla 6.4. Métricas del ejemplo de la facultad

Tipo	Orden medio	Orden mínimo	Orden máximo	Cantidad	Máximo
Déficit				**1**	6
Exceso				**1**	6
Sobrecarga	2	2	2	**2**	6
Redundancia	3	3	3	**1**	6
Unión				**6**	36
Modelo				**5***	

Tipo	Razón	Razón ajustada	Peso normalizado	Peso ajustado	Peso de la Discrepancia	Contribución
Déficit	**17%**	8%	33%	67%	**11%**	**26%**
Exceso	**17%**	8%	17%	33%	**6%**	**13%**
Sobrecarga	**33%**	17%	30%	60%	**20%**	**46%**
Redundancia	**17%**	8%	15%	30%	**5%**	**12%**
Unión	**17%**	8%	5%	10%	**2%**	**4%**
Modelo		**42%***	100%	200%*	**43%**	100%

* Ver Tabla 6.3

Nota: las métricas más útiles están en **negritas**.

En la Tabla 6.5, se muestra el tipo al que pertenecen las métricas que se consideran más útiles.

Tabla 6.5. Tipo de métricas propuestas (individual o por cada tipo de Discrepancia)

Métrica	**Rango de la medición**	**Tipo de escala de medición**[35]	**Medición directa o indirecta**[36]	**Aplicación a las Discrepancias: individual, modelo**
Tipo	$\{def, exc, over, red, uni, Mod\}$	Nominal	Indirecta	Individual, modelo
Orden medio	Real no negativo $[2, max_k]$ (0, si no hay Discrepancias Representadas)	Intervalo	Indirecta	Individual
Orden mínimo	Entero no negativo $\{2, 3, \ldots, max_k\}$ (0, si no hay Discrepancias Representadas)	Intervalo	Directa	Individual
Orden máximo	Entero no negativo $\{2, 3, \ldots, max_k\}$ (0, si no hay Discrepancias Representadas)	Intervalo	Directa	Individual
Cantidad	Entero no negativo $\{0, 1, \ldots, max_{over} + max_{red}\}$	Absoluto	Directa	Individual, modelo
Razón	Real [0, 1]	Razón	Indirecta	Individual
Razón ajustada	Real [0, 1]	Razón	Indirecta	Modelo
Peso de la Discrepancia	Real [0, 1]	Razón	Indirecta	Individual, modelo
Contribución	Real [0, 1]	Razón	Indirecta	Individual

Interpretación de las métricas de cada tipo de Discrepancia y del modelo

- Orden medio, orden mínimo, orden máximo
  - Individual
    - Los estadísticos de los órdenes de las Discrepancias Representadas, puedan dar una idea sobre el comportamiento de cada tipo sobre el establecimiento de las prioridades del análisis.
    - Se sugiere empezar por las Sobrecargas de mayor orden (usualmente son más problemáticas que la Redundancia).
    - Si el rango (variabilidad) es muy bajo y el orden medio es positivo (i.e. existen Discrepancias Representadas), se pudiera pensar en un comportamiento repetido de los modeladores. Esto pudiera sugerir

[35] [Fenton97] p. 46-53. Ver Glosario.
[36] [Fenton97] p. 39-40. Ver Glosario.

cierta relación entre algunos elementos agrupadores[37] (e.g. entre paquetes con un alto requerimiento de desempeño, modularidad, etc.)

- Por ejemplo, en un Dominio de una base de datos multimedia, se pudieran tener muchas Redundancias con orden medio de dos y un rango corto. Lo cual, pudiera sugerir que son Redundancias de subtipo (i.e. casi cada Clase Conceptual se representa con dos Clases de Diseño). Esto no sería problemático, sino útil para separar los grandes archivos de multimedia de la información de búsqueda sobre ellos (tipo de formato, tamaño, palabras clave, fechas, etc.).

  - Si se desea más detalle, se pueden tomar otros estadísticos como la desviación estándar, curtosis, sesgo, etc.

- Cantidad
  - Análogamente a las líneas de código, puede ser empleada para dar una idea sobre la estimación del esfuerzo del análisis de las Discrepancias.
  - Individual
    - Se debe analizar cada Discrepancia para elegir entre eliminarla o justificarla.
    - Dependiendo del dominio y los requerimientos, algunos tipos de Discrepancias pueden ser más problemáticos que otros, por lo que se les deberá de prestar mayor atención en el análisis.
  - Modelo
    - Si se puede pensar que un sistema de 100 KLOC requiere de más recursos que uno de 10 KLOC (para aplicaciones similares); un modelo de 100 Discrepancias pudiera necesitar más tiempo de análisis que uno de 10 Discrepancias.
    - No es determinante para la toma de decisiones para el manejo de las Discrepancias, aunque puede dar una idea sobre la magnitud del análisis.
- Razón
  - Individual
    - Establecer estándares de comparación ("benchmarks") para el control de las Discrepancias.
    - Comparación entre elementos de distinto tamaño (e.g. varios paquetes con distinto número de clases).
    - Analizar el comportamiento del modelado.
- Razón ajustada
  - Modelo
    - Puede ser útil para comparar la Brecha de dos sistemas de un mismo Dominio independientemente del tamaño de los modelos (e.g. número de elementos).

[37] El elemento agrupador de los atributos es la clase; el de las clases es el paquete; el del paquete es otro paquete de nivel más alto.

- Peso de la Discrepancia
  - Individual
    - Establecer prioridades para analizar primero las Discrepancias de mayor importancia.
  - Modelo
    - Aunque no es determinante para la toma de decisiones sobre el manejo de las Discrepancias, puede dar una idea sobre la magnitud del análisis.
- Contribución
  - Individual
    - Establecer prioridades para analizar primero las Discrepancias de mayor importancia.
    - Analizar el comportamiento del modelado.

Adjetivos para la Brecha Representacional

Se propone en la Tabla 6.6, un breve argot sobre la Brecha Representacional con adjetivos de base cuantificable.

Tabla 6.6. Adjetivos de la Brecha Representacional

Adjetivo	**Significado**	**Notación**
"larga"	Muchas clases	$n_C > N, n_D > N, N >> 1$
"corta"	Pocas clases	$n_C < N, n_D < N, N >> 1$
"amplia"	Representación dispersa	$g_{Mod} \approx 1,\ g_{Mod} \in (0.5, 1)$
"media"	Representación entre dispersa y emparejada.	$g_{Mod} \approx 0.5,\ g_{Mod} \in [0.5 - \Delta, 0.5 + \Delta]$
"estrecha"	Representación uno a uno frecuente (muy emparejada)	$g_{Mod} \approx 0,\ g_{Mod} \in (0, 0.5)$
"nula", "mínima", "inexistente"	Sólo representación uno a uno, correspondencia máxima del grafo	$g_{Mod} = 0 \Rightarrow n_C = n_D = n_{Arc}$
"máxima"	Sin representación o con representación entre todos los elementos (grafo disjunto o completo, respectivamente)	$g_{Mod} = 1 \Leftrightarrow n_{Arc} = 0 \vee n_{Arc} = n_C \cdot n_D$

6.2 Otros métodos para medir la Brecha Representacional

Para la selección del método se experimentó con varios métodos, por orden de importancia:

- Prueba de Independencia Chi Cuadrada
- Conteo Simple
- Flujo
- Heurísticas

A continuación se menciona una breve reseña de cada uno.

6.2.1 Prueba de Independencia Chi Cuadrada

Dado que el refinamiento es un tipo de dependencia, se hicieron diversas pruebas con la Prueba de Independencia Chi Cuadrada. Aunque el cálculo y los grados de libertad son los mismos que para la Prueba de Homogeneidad [Rice95], cabe recordar que para probar independencia solamente el total es fijo (los totales de las columnas y renglones puede cambiar).

Cabe resaltar que por la naturaleza del problema, la Prueba Exacta de Fisher es más apropiada que la Prueba de Independencia Chi Cuadrada. Básicamente, porque [Cochran52] y [Sheskin03] ofrecen la alternativa de la Prueba Exacta de Fisher para los problemas donde no se cumpla el supuesto de que cada celda tenga un mínimo de cinco elementos.

La métrica del modelo consiste en el complemento del p-valor $(1-p)$. Este complemento es para ser consistentes en que cuando la Brecha sea estrecha, la métrica esté cercana a cero. A pesar de ofrecer muy buenos resultados, se encontraron algunos contraejemplos con las siguientes observaciones (i.e. desventajas) que se aprecian en la Tabla 6.7 a la Tabla 6.10:

1. Se contradice la secuencia esperada de los ejemplos de prueba (ver la Tabla 6.7 a la Tabla 6.10)
2. La métrica mínima del modelo no era el caso de máxima correspondencia -sólo relaciones uno a uno- (ver la Tabla 6.7).
3. La métrica máxima del modelo no era sólo para la matriz de unos -grafo bipartito completo- o de ceros -grafo bipartito totalmente desconectado- (ver la Tabla 6.8)
4. No eran válidos y necesitaban algún ajuste empírico.

Tabla 6.7. Contraejemplo de la secuencia esperada y la métrica mínima en la Prueba de Independencia

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	O2R2	O2R2	-	-	-	-	-
c[2]	O2R2	O2R2	-	-	-	-	-
c[3]	-	-	1to1	-	-	-	-
c[4]	-	-	-	1to1	-	-	-
c[5]	-	-	-	-	1to1	-	-
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	-	-	

En la Tabla 6.7, con una métrica del modelo del 16%, se presenta la primera y segunda observación. La métrica es menor al mínimo (22%) que pertenece a la máxima correspondencia, con lo cual se rompe la secuencia esperada. Anterior a los contraejemplos, se realizaba un ajuste lineal de forma que el mínimo y el máximo correspondieran respectivamente al cero y al uno.

Tabla 6.8. Contraejemplo de la métrica máxima en la Prueba de Independencia

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	O3R3	O3R3	O3R3	-	-	-	-
c[2]	O3R3	O3R3	O3R3	-	-	-	-
c[3]	O3R3	O3R3	O3R3	-	-	-	-
c[4]	-	-	-	-	-	-	D
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	-	D
E	-	-	-	E	E	E	

En la Tabla 6.8, se tiene la tercera observación, donde la métrica del modelo era del 100%.

La cuarta observación se presenta cuando $r_{i,j} = 0$, pues no es válido para la Prueba de Independencia. Por lo que se intentó un ajuste con el Conteo con Prueba de Independencia Chi Cuadrada.

Conteo con Prueba de Independencia Chi Cuadrada

Una posibilidad de ajuste cuando se tenía $r_{i,j} = 0$, era un método híbrido. Se empleaba la Prueba de Independencia sólo en los renglones y columnas que participaran en una Discrepancia Representada (Sobrecarga y Redundancia). Mientras que para las No Representadas (Déficit y Exceso) sólo se contabilizaban y se calculaban las razones. Sin embargo, no se lograba el orden esperado.

Tabla 6.9. Discrepancias unidas

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	D
c[1]	R3	R3	O2R3	-	-	-	-
c[2]	-	-	O2	-	-	-	-
c[3]	-	-	-	O2	-	-	-
c[4]	-	-	-	O2	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	E	-	

Tabla 6.10. Discrepancias separadas

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	D
c[1]	R3	R3	R3	-	-	-	-
c[2]	-	-	-	O3	-	-	-
c[3]	-	-	-	O3	-	-	-
c[4]	-	-	-	O3	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	E	-	

Además, se invierte la secuencia del ejemplo de las Discrepancias unidas (Tabla 6.9) del ejemplo de las Discrepancias separadas (Tabla 6.10), pues las métricas respectivas son 90% y 96%, lo cual pudiera ser aceptable, pero no es deseado.

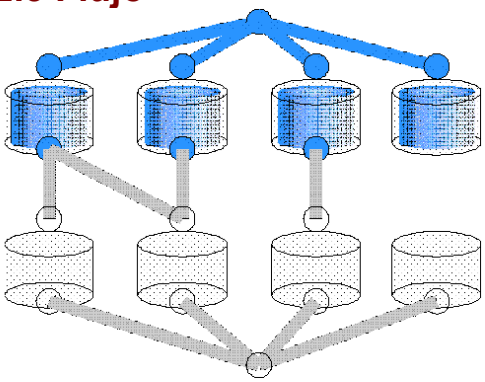
6.2.2 Conteo Simple

El Conteo Simple es un caso especial del Conteo Ponderado (ver "Guías para la asignación de los pesos" en la página 111). Sin embargo, tiene algunas desventajas respecto al Conteo Ponderado:

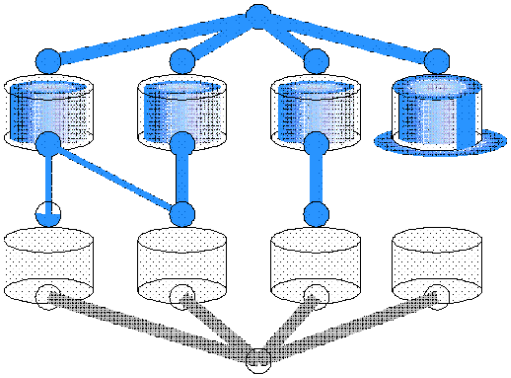
(a) No considera el orden de las Discrepancias
(b) No considera la Unión de las Discrepancias
(c) Supone que todos los tipos de Discrepancias tiene el mismo impacto

Debido a estas desventajas hay casos donde la métrica es igual y que se pudiera esperar que fueran distintas. Además, se tiene la tercera observación de la Prueba de Independencia de la Tabla 6.8.

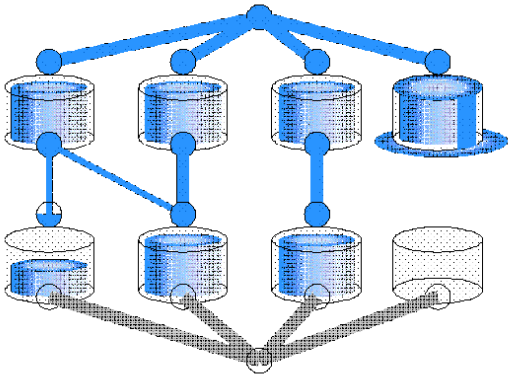
6.2.3 Flujo



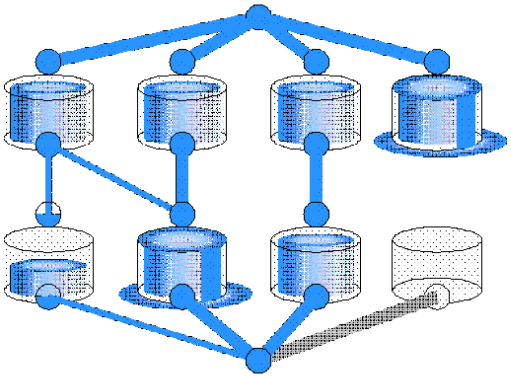
(1)
El flujo unitario inicia en la fuente



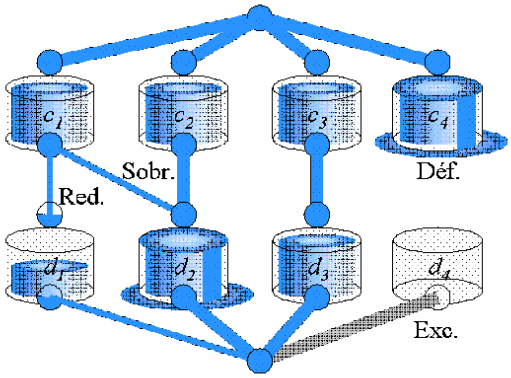
(2)
Se tiene un Déficit de capacidad en la Clase Conceptual y algunas Discrepancias en las tuberías



(3)
Se empieza el "llenado" de las Clases de Diseño



(4)
Se tienen Discrepancias en las tuberías conectadas al destino



(5)
Se localizan las Discrepancias de capacidad de representación de las clases

Figura 6.3. Apreciación de la Brecha mediante el flujo de la representación

La idea en este método es simular la representación ontológica mediante un flujo que pasa a través de los elementos. Se supone que el flujo inicia en un nodo fuente (e.g. como si fuera un manantial) que se conecta a cada elemento del Modelo General (e.g. Clase Conceptual) con una "tubería" de flujo unitario (Figura 6.3 (1)). Luego, se considera cada refinamiento como una "tubería" de capacidad unitaria que parte de los elementos del Modelo General a los del Modelo Particular (e.g. de las Clases Conceptuales a las Clases de Diseño), esto se aprecia en la Figura 6.3, incisos (2) y (3). El flujo se concentra de estos últimos elementos a un nodo destino (Figura 6.3, incisos (4) y (5)).

Para medir la Brecha se calcula alguna forma de error (e.g. error cuadrático) en los casos donde el flujo era distinto del unitario. El error se puede pensar como el desborde o la escasez en los nodos y en las "tuberías". Las Discrepancias se pueden ver como diferencias de la capacidad de los contenedores (nodos y "tuberías"). Por ejemplo, un Déficit de capacidad en el contenedor. Para ilustrar esta apreciación se tiene la secuencia del flujo en la Figura 6.3, donde los nodos son Clases Conceptuales (c_i) y Clases de Diseño (d_j), y las "tuberías" son los refinamientos.

Las desventajas con este método son:

- La matriz de unos (100%) tenía una métrica distinta a la matriz de ceros (40%).
- La curva de la métrica parecía cóncava hacia arriba mientras que los demás métodos parecían lineales. Es decir, se necesitaban muchas Discrepancias para que la métrica fuera mayor al 50%. Generalmente, la métrica de los ejemplos era menor al 40%.

6.2.4 Heurística

Más que un método en sí fueron intentos empíricos para dar una primera idea para medir la Brecha. De alguna forma, se trataba de medir el error cuadrático con las diferencias esperadas de la suma de los renglones y las columnas. Las mejores fórmulas tenían un cierto parecido con la Prueba de Independencia. El objetivo era penalizar los cruces de las Discrepancias Representadas y los órdenes altos (e.g. una Redundancia de orden dos, es más estrecha que una de orden tres).

6.2.5 Comparación de los métodos de medición

En la Tabla 6.11, se diferencian los métodos mediante los ejemplos más significativos. Las desventajas enlistadas anteriormente se pueden identificar con el orden ascendente esperado de la métrica. Cuando se tiene una misma letra, se pudiera considerar con una métrica "cercana".

Tabla 6.11. Comparación de los métodos de medición mediante ejemplos significativos

Orden esperado	Id	Ejemplo	Conteo ponderado	Conteo Simple	Prueba de Independencia	Flujo
A	1	Diagonal	0%	0%	**22%**	0%
B	7	Complementarias	30%	**33%**	**16%**	7%
C	5	Separadas	32%	**33%**	***96%***	***27%***
C	4	Unidas	42%	42%	***90%***	***22%***
D	6	Bloque	96%	**100%**	**100%**	40%
E	3	Unos	100%	100%	100%	100%
E	2	Ceros	100%	100%	100%	**40%**

Leyenda

123%	Se esperaba menor valor
123%	Se esperaba mayor valor
123%	Diferencia aceptable

Los ejemplos más significativos se muestran desde la Tabla 6.12 hasta la Tabla 6.18. El identificador es para referenciar el ejemplo con la Tabla 6.11, y se les nombró para que fuera más fácil distinguirlos.

Tabla 6.12. Ejemplo: 1. Diagonal (máxima correspondencia del grafo)

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	0	0	0	0	0
c[2]	0	1	0	0	0	0
c[3]	0	0	1	0	0	0
c[4]	0	0	0	1	0	0
c[5]	0	0	0	0	1	0
c[6]	0	0	0	0	0	1

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	D
c[1]	1to1	-	-	-	-	-	-
c[2]	-	1to1	-	-	-	-	-
c[3]	-	-	1to1	-	-	-	-
c[4]	-	-	-	1to1	-	-	-
c[5]	-	-	-	-	1to1	-	-
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	-	-	

Tabla 6.13. Ejemplo: 2. Ceros (conjuntos disjuntos)

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	0	0	0	0	0	0
c[2]	0	0	0	0	0	0
c[3]	0	0	0	0	0	0
c[4]	0	0	0	0	0	0
c[5]	0	0	0	0	0	0
c[6]	0	0	0	0	0	0

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	-	-	-	-	-	-	D
c[2]	-	-	-	-	-	-	D
c[3]	-	-	-	-	-	-	D
c[4]	-	-	-	-	-	-	D
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	-	D
E	E	E	E	E	E	E	

Tabla 6.14. Ejemplo: 3. Unos (grafo bipartito completo)

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	1	1	1	1
c[2]	1	1	1	1	1	1
c[3]	1	1	1	1	1	1
c[4]	1	1	1	1	1	1
c[5]	1	1	1	1	1	1
c[6]	1	1	1	1	1	1

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
c[2]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
c[3]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
c[4]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
c[5]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
c[6]	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	O6 R6	-
E	-	-	-	-	-	-	

Tabla 6.15. Ejemplo: 4. Unidas

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	1	0	0	0
c[2]	0	0	1	0	0	0
c[3]	0	0	0	1	0	0
c[4]	0	0	0	1	0	0
c[5]	0	0	0	0	0	0
c[6]	0	0	0	0	0	1

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	R3	R3	O2R3	-	-	-	-
c[2]	-	-	O2	-	-	-	-
c[3]	-	-	-	O2	-	-	-
c[4]	-	-	-	O2	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	E	-	

Tabla 6.16. Ejemplo: 5. Separadas

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	1	0	0	0
c[2]	0	0	0	1	0	0
c[3]	0	0	0	1	0	0
c[4]	0	0	0	1	0	0
c[5]	0	0	0	0	0	0
c[6]	0	0	0	0	0	1

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	R3	R3	R3	-	-	-	-
c[2]	-	-	-	O3	-	-	-
c[3]	-	-	-	O3	-	-	-
c[4]	-	-	-	O3	-	-	-
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	E	-	

Tabla 6.17. Ejemplo: 6. Bloque

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	1	0	0	0
c[2]	1	1	1	0	0	0
c[3]	1	1	1	0	0	0
c[4]	0	0	0	0	0	0
c[5]	0	0	0	0	0	0
c[6]	0	0	0	0	0	0

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	O3 R3	O3 R3	O3 R3	-	-	-	-
c[2]	O3 R3	O3 R3	O3 R3	-	-	-	-
c[3]	O3 R3	O3 R3	O3 R3	-	-	-	-
c[4]	-	-	-	-	-	-	D
c[5]	-	-	-	-	-	-	D
c[6]	-	-	-	-	-	-	D
E	-	-	-	E	E	E	

Tabla 6.18. Ejemplo: 7. Complementarias

R

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
c[1]	1	1	0	0	0	0
c[2]	1	1	0	0	0	0
c[3]	0	0	1	0	0	0
c[4]	0	0	0	1	0	0
c[5]	0	0	0	0	1	0
c[6]	0	0	0	0	0	1

S

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	**D**
c[1]	O2R2	O2R2	-	-	-	-	-
c[2]	O2R2	O2R2	-	-	-	-	-
c[3]	-	-	1to1	-	-	-	-
c[4]	-	-	-	1to1	-	-	-
c[5]	-	-	-	-	1to1	-	-
c[6]	-	-	-	-	-	1to1	-
E	-	-	-	-	-	-	

Características deseables de los métodos de medición

En la Tabla 6.19, se muestra un resumen sobre las características más relevantes. La regla intuitiva es que el método se considera mejor entre más respuestas afirmativas y los adjetivos representen "más".

Tabla 6.19. Características deseables de los métodos de medición

Características deseables	**Conteo Ponderado**	**Conteo Simple**	**Prueba de Independencia**	**Flujo**
La métrica máxima es la matriz de ceros y la de unos.	Sí	Sí	Sí	No
Sólo la matriz de ceros y la de unos dan la métrica máxima.	Sí	No	No	Sí
Flexible (pesos ajustables)	Sí.	No	No	Sí.
Considera el orden y la unión de las Discrepancias	Sí	No	Sí	Sí
La diagonal es la métrica mínima	Sí	Sí	Sí	No
Intuitiva (característica subjetiva)	"Muy"	"Muy"	"Algo"	"Algo"

6.3 Métricas de la eficiencia del modelador

Una vez calculadas las métricas de la Brecha[38], se recomienda analizar las Discrepancias, antes de pasar a la siguiente Disciplina. Esto con el fin de revisar cada Discrepancia para decidir si se eliminan o se justifican. Al realizar este análisis se estará en condiciones de evaluar la eficiencia (en función de la cantidad de defectos) en la modelación del dominio y el diseño, es decir, qué tantas Discrepancias no estaban justificadas cuando se corrigió el modelo.

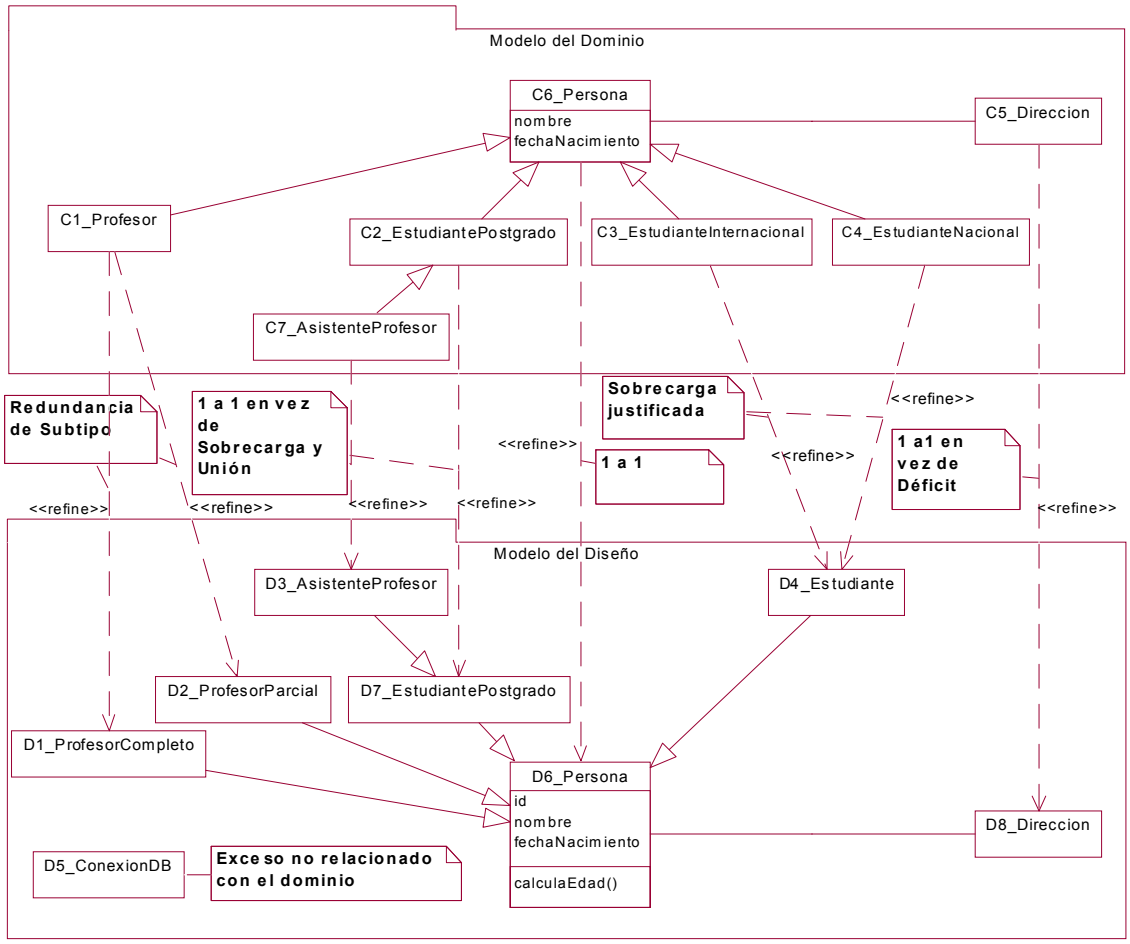


Figura 6.4. Modelo corregido del ejemplo de la facultad

[38] Recordar que se sugiere calcular las métricas en cada iteración una vez terminado el Modelo del Diseño.

Tabla 6.20. Identificación de las Discrepancias del modelo corregido de la facultad

C\D	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]	d[7]	d[8]	**Def**
c[1]	R2	R2	-	-	-	-	-	-	-
c[2]	-	-	-	-	-	-	1To1	-	-
c[3]	-	-	-	O2	-	-	-	-	-
c[4]	-	-	-	O2	-	-	-	-	-
c[5]	-	-	-	-	-	-	-	1To1	-
c[6]	-	-	-	-	-	1To1	-	-	-
c[7]	-	-	1To1	-	-	-	-	-	-
Exc	-	-	-	-	E	-	-	-	

Para ilustrar esta medición, se retomará el ejemplo de la facultad (ver Tabla 3.8 para la documentación de las clases). Se supone que después de la revisión de las Discrepancias, se realizaron algunos cambios que se aprecian en el modelo UML corregido de la Figura 6.4 y en la identificación de Discrepancias de la Tabla 6.20. El modelo anterior al análisis se puede consultar en la Figura 5.8 y en la Tabla 5.2.

Cabe resaltar que no basta con eliminar las Discrepancias indeseables, sino que se deben documentar aquéllas deseables para justificarlas. Se sugiere fuertemente, iniciar con la justificación para reducir la posibilidad de cambios futuros. Cabe resaltar que se debe fijar el vector de pesos entre las comparaciones, para mantener la consistencia entre las métricas.

Es de esperarse que las Redundancias de subtipo y los Excesos no orientados al dominio del problema, sean las Discrepancias justificadas más comunes. Como sucedió con las clases “C1_Profesor”, “D1_ProfesorCompleto” y “D2_ProfesorParcial”, para el primero; y “D5_ConexionDB” para el segundo. Al cambiar el modelo, se pueden crear, cambiar y eliminar varios elementos. En este caso, se agregaron una Clase Conceptual (“C7_AsistenteProfesor”) y dos Clases de Diseño (“D7_EstudiantePostgrado” y “D8_Direccion”).

Para concentrarnos en los cambios se eliminaron las columnas: orden medio, orden mínimo, orden máximo, razón ajustada (excepto la razón del modelo que se agrega a la columna razón), peso normalizado y peso ajustado. Sin embargo, algunas de estas columnas ofrecen información adicional.

6.3.1 Cálculo de la eficiencia del modelador

Para evaluar los cambios, se aplica la fórmula del error relativo. En la Tabla 6.21, se muestran las variables.

Tabla 6.21. Descripción de variables para las métricas del cambio

Variable	Descripción	Tipo de tabla
I	Conjunto de métricas iniciales	Entrada
J	Conjunto de métricas justificadas	Entrada
E	Conjunto de métricas eliminadas	Cálculo
ε_{abs}	Error absoluto	Salida
ε	Error relativo	Salida
Effi	Eficiencia (ver "Apéndice A. Glosario")	Salida complementaria

El hecho de que las Discrepancias iniciales se componen de justificadas y por eliminar, se representa con $I = J + E$. Dado que las Discrepancias eliminadas son indeseables, constituyen el error absoluto $\varepsilon_{abs} = E = I - J$. El error relativo, simplemente es $\varepsilon = \frac{\varepsilon_{abs}}{I} = 1 - \frac{J}{I}$. Para prevenir la división entre cero ($I = 0$), y dado que se espera que $J \leq I$, se define $\varepsilon = 0$ por conveniencia. Ahora bien, el complemento del error relativo puede representar la eficiencia del modelador, $Effi = 1 - \varepsilon$. Cabe recordar que la cantidad y la razón del modelo no consideran el renglón de la Unión, y la razón del modelo ha sido ajustada. Estos sencillos cálculos se muestran en la Tabla 6.22 a la Tabla 6.27.

Tabla 6.22. Resumen de métricas del modelo inicial ejemplo de la facultad (Discrepancias iniciales, I)

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	1	6	17%	11%	26%
Exceso	1	6	17%	6%	13%
Sobrecarga	2	6	33%	20%	46%
Redundancia	1	6	17%	5%	12%
Unión	6	36	17%	2%	4%
Modelo	**5**		**42%**	**43%**	**100%**

Tabla 6.23. Resumen de métricas del modelo corregido del ejemplo de la facultad (Discrepancias justificadas, J)

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	0	7	0%	0%	0%
Exceso	1	8	13%	4%	25%
Sobrecarga	1	8	13%	8%	45%
Redundancia	1	7	14%	4%	26%
Unión	4	56	7%	1%	4%
Modelo	**3**		**20%**	**17%**	**100%**

En la Tabla 6.23, se muestran las métricas que se obtuvieron después de las correcciones. Las clases agregadas se aprecian en los máximos. Dado que generalmente, las Discrepancias son problemáticas, se espera que la cantidad de Discrepancias del modelo disminuya, como sucede en este ejemplo.

Tabla 6.24. Diferencia entre el modelo inicial y el corregido (Discrepancias eliminadas, $\varepsilon_{abs} = E$)

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	1	-1	17%	11%	26%
Exceso	0	-2	4%	1%	-12%
Sobrecarga	1	-2	21%	13%	1%
Redundancia	0	-1	2%	1%	-14%
Unión	2	-20	10%	1%	0%
Modelo	**2**		**22%**	**27%**	**0%**

Tabla 6.25. Métricas del error relativo (ε)

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	100%	-17%	100%	100%	100%
Exceso	0%	-33%	25%	25%	-95%
Sobrecarga	50%	-33%	63%	63%	2%
Redundancia	0%	-17%	14%	14%	-123%
Unión	33%	-56%	57%	57%	-11%
Modelo	**40%**		**53%**	**62%**	**0%**

Tabla 6.26. Métricas de la eficiencia (*Effi*)

Tipo	Cantidad	Máximo	Razón	Peso de la Discrepancia	Contribución
Déficit	0%	117%	0%	0%	0%
Exceso	100%	133%	75%	75%	195%
Sobrecarga	50%	133%	38%	38%	98%
Redundancia	100%	117%	86%	86%	223%
Unión	67%	156%	43%	43%	111%
Modelo	**60%**		**47%**	**38%**	**100%**

Aunque convencionalmente, se omite el signo del error absoluto, se eligió dejarlo para resaltar los cambios reflejados en la Tabla 6.24 y la Tabla 6.25. La agregación de las clases se aprecia en los máximos negativos en la Tabla 6.24. Dado que es una diferencia, es de esperarse que las contribuciones se anulen.

En la Tabla 6.25, se muestra los errores relativos por tipo de Discrepancia y por tipo de métrica. Por ejemplo, el 50% de la cantidad de Sobrecargas implica que la mitad de las Discrepancias iniciales estaban justificadas y que la otra mitad fueron eliminadas. Dentro del mismo renglón, el -17%, significa la falta de una Clase Conceptual. El tipo de Discrepancia que menos cambió fue la Redundancia (14%). Cabe recordar que la Tabla 6.25 y la Tabla 6.26 son complementarias (suma nula). Por lo que no es de extrañar que las columnas de la razón y del peso de la Discrepancia son iguales (excepto, las métricas del Modelo que son las diferencias).

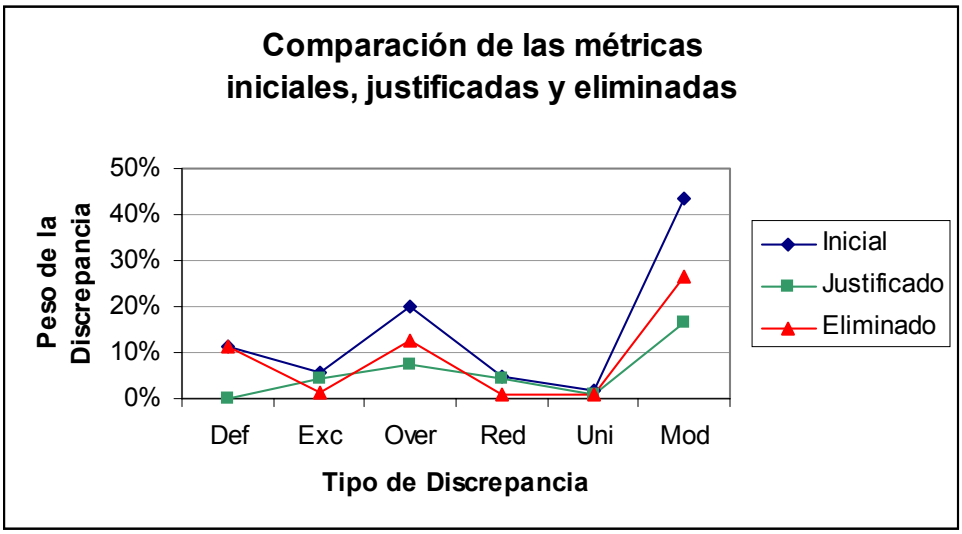


Figura 6.5. Comparación de las métricas iniciales, justificadas y eliminadas

En la Figura 6.5, se puede verificar visualmente que la suma de las métricas justificadas y eliminadas es la métrica inicial. Las Discrepancias eliminadas fueron las Abundantes en lo General, lo que generalmente es más problemático que si fueran Abundantes en lo particular. También se aprecia que se presentó en poca escala el fenómeno de la Unión de

las Discrepancias. Por la métrica del modelo, se puede decir que había más Discrepancias de las deberían (i.e. hubo más defectos que aciertos).

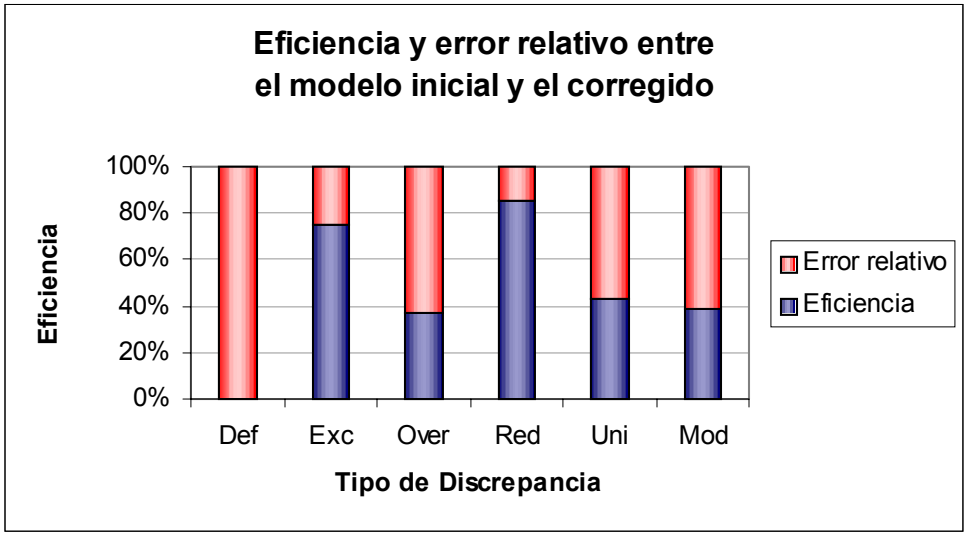


Figura 6.6. Eficiencia y error relativo entre el modelo inicial y el corregido

En la Figura 6.6, se resaltan los defectos para las Discrepancias Abundantes en lo General, pues los errores más grandes fueron en los Déficit (100%) y en las Sobrecargas (63%). La Unión no se considera pues su contribución en las Discrepancias eliminadas (ver Tabla 6.24) fue muy baja. Con una eficiencia del 38% (y un 62.5% de error) se puede decir que los productos del analista y el diseñador difieren considerablemente.

6.3.2 Síntesis de las métricas del cambio en el modelo

Recolectando las métricas del cambio en el modelo en la Tabla 6.27, se puede decir que debido a las correcciones:

- Hubo una reducción en el número de Discrepancias del 40% (error relativo de la cantidad).
- Hubo una reducción sin ponderar en la Brecha del 53% con respecto al modelo inicial (error relativo en la razón ajustada).
- Hubo una reducción ponderada en la Brecha del 62% con respecto al modelo inicial (error relativo en el peso de la Discrepancia).
- El 38% de las Discrepancias estaban justificadas (correctas), esta métrica se pudiera tomar como la eficiencia del modelador.

Tabla 6.27. Síntesis de las métricas del cambio en el modelo[39]

Modelo	Cantidad	Razón	Peso de la Discrepancia
Inicial	5	42%	43%
Justificado	3	20%	17%
Eliminado	2	22%	27%
Error relativo	***40%***	***53%***	***62%***
Eficiencia	60%	47%	***38%***

Una interpretación de estas métricas, pudiera ser que el modelador no es bueno o que la modelación del sistema es aun inestable.

[39] Se omitió la columna para el máximo del modelo pues no aplica y la contribución pues son alternaciones de 0% y 100%.

6.4 Métricas de software propuestas dentro del modelo de calidad de McCall

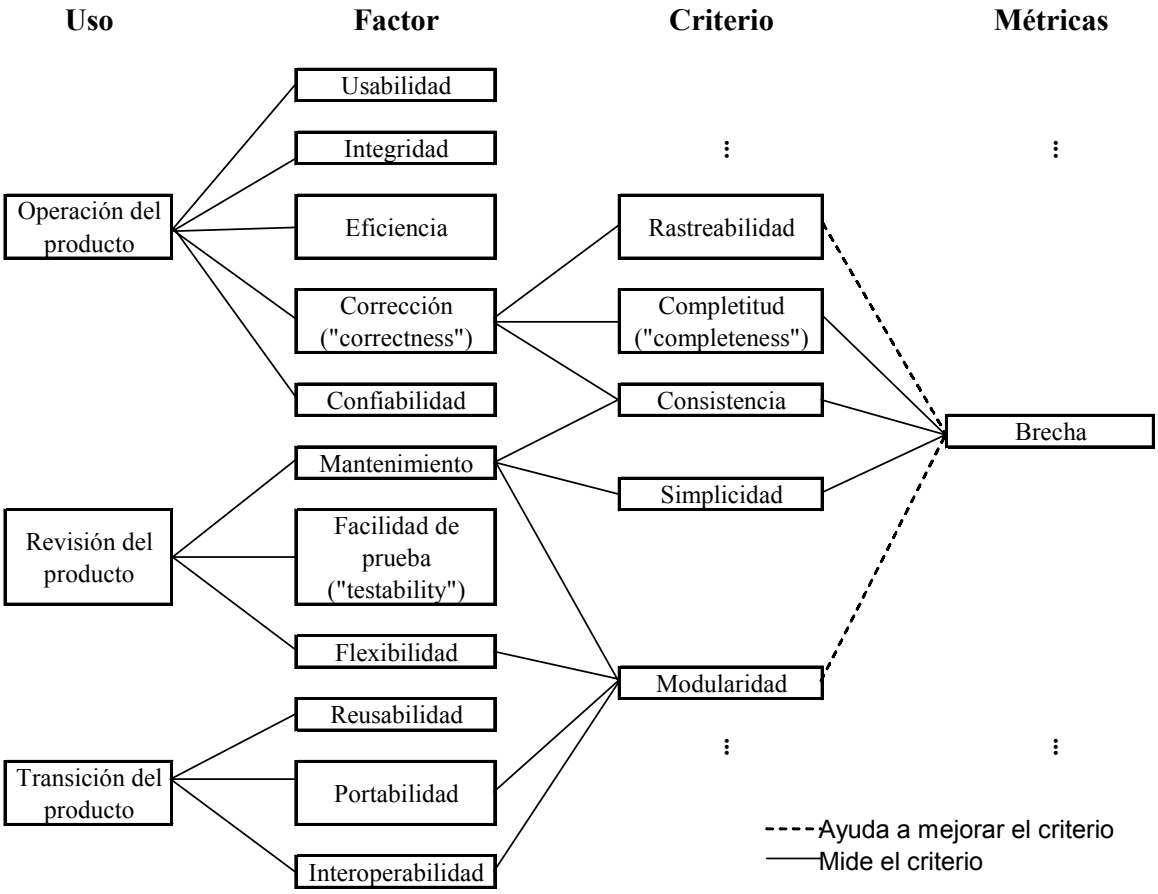


Figura 6.7. Modelo de calidad del software de McCall

En la Figura 6.7, se muestran todos los usos, todos los factores del modelo de calidad del software de McCall [Fenton97], y aquellos criterios relacionados con las métricas de la Brecha. Estas métricas miden los criterios de completitud, consistencia y simplicidad, que representan a los factores de corrección y mantenimiento. En particular, las Discrepancias No Representadas pueden evidenciar que el modelo es incompleto. Una Brecha amplia puede ser sinónimo de inconsistencias en la representación, mientras que una Brecha estrecha, puede significar una sencillez en el modelado. Además, mediante el análisis de Discrepancias se puede mejorar la rastreabilidad bidireccional y modularidad de los modelos.

6.5 Beneficios colaterales del análisis de Discrepancias

El análisis de las Discrepancias tiene beneficios secundarios similares a una revisión sistemática por el cuestionamiento de la justificación de los refinamientos. En el caso de estudio sobre las matrices, se tuvieron los siguientes beneficios (ver caso para mayor detalle):

- Renombrado más consistente.
- Reducción de atributos redundantes.
- Decisión de si un elemento es clase o atributo.
- Agrupación más adecuada (de sentido común).
- Crítica de la necesidad de atributos.

En resumidas cuentas, el análisis de Discrepancias ayuda a reflexionar y modularizar el análisis y el diseño del sistema de forma más ordenada.

6.6 Comentarios sobre la Brecha Representacional de Larman

[Larman01] piensa que una Brecha estrecha es útil en el diseño del sistema. Su argumentación es intuitiva al afirmar que la mayoría de los ingenieros de software saben que es cierto, "aunque es difícil de cuantificar".

En el presente trabajo, se considera que en vez de buscar una Brecha estrecha *per se*, se deberían de cotejar con algunos estándares de comparación ("benchmarks") dependientes del problema, requerimientos, tipo de Dominio y alcance del sistema. Para verificar estos valores faltaría tomar varias métricas de varios proyectos de varios Dominios, lo cuál queda fuera del alcance del trabajo actual, pero se puede utilizar el programa desarrollado.

Si bien [Larman01] plantea el fenómeno de la Brecha Representacional, no la define formalmente, ni la caracteriza (localiza, identifica o mide). Más aun, expone la dificultad de su medición. Por estas razones, se propone una definición formal, la caracterización de la Brecha Representacional y, en vez de buscar que sea estrecha, se considera que es mejor entenderla y controlarla de forma sistemática, para prevenir y corregir defectos en las Disciplinas tempranas del desarrollo del sistema.

Parte 3. Casos de estudio

Capítulo 7. Caso de estudio de matrices: “AbstractMatrix”

En este caso, se realizó una ingeniería en reversa del código fuente de un proyecto académico, sobre una parte de la librería de Álgebra Lineal. Esta librería fue escrita en el invierno del 2001 (aproximadamente hace dos años), no fue alterada y la estructura no fue realizada por una herramienta CASE. Además, el propósito inicial no fue el de crear una librería, sino que se fue agrupando como práctica personal, conforme el curso iba avanzando.

El objetivo de este caso de estudio es comparar el modelo conceptual de las matrices con el diseño extraído del código fuente.

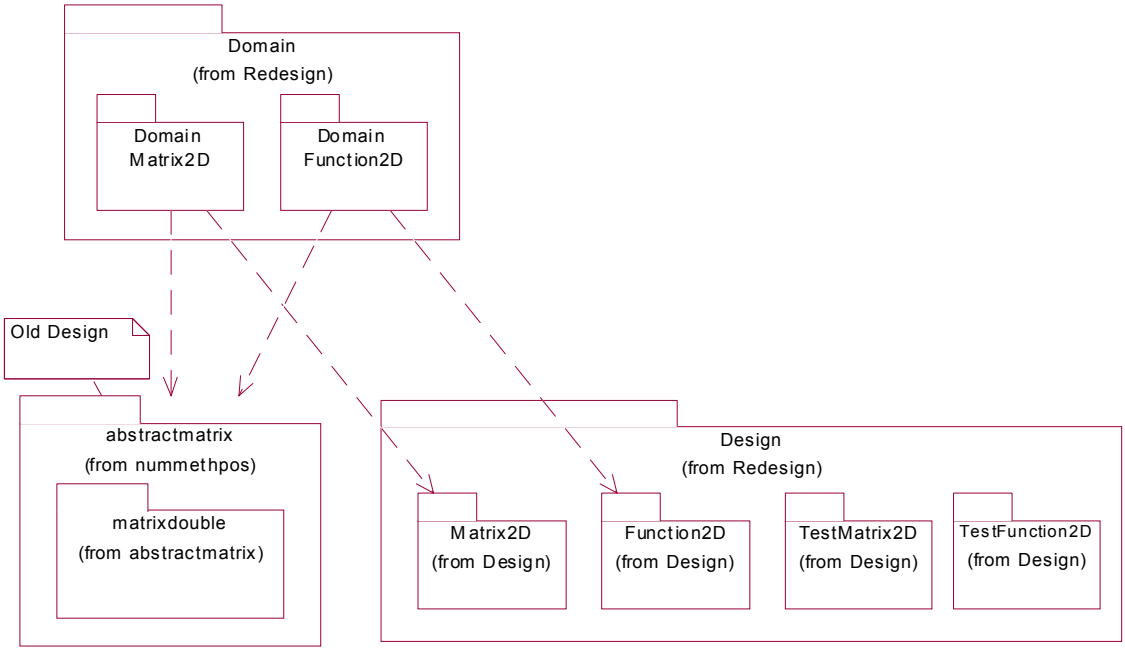


Figura 7.1. Organización de paquetes del dominio, diseño anterior y diseño actual del caso de matrices

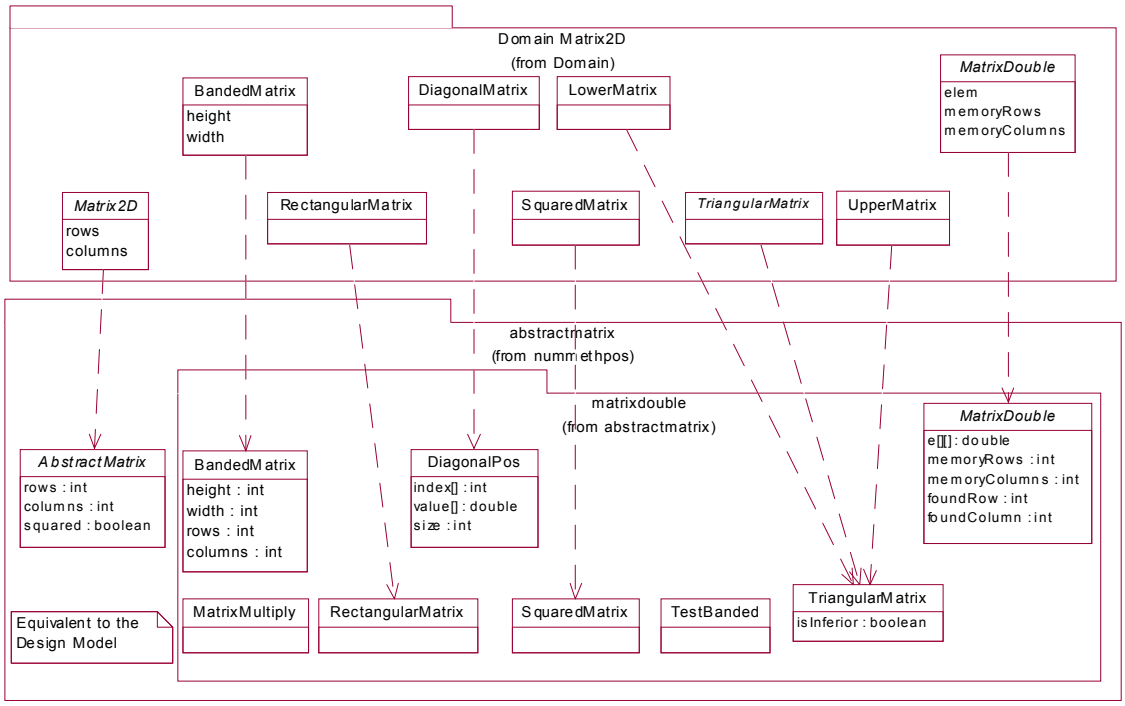


Figura 7.2. Refinamientos entre clases del diseño anterior en el caso de matrices

Este estudio, puede ser útil incluso para correcciones. En este caso, se acentuaron las siguientes:

- Renombrado más consistente.
  - Se renombró “DiagonalPos” por “DiagonalMatrix”. Además, al analizar los diagramas se pueden notar errores ortográficos (se sustituyó “finded” por “found”) y nombres fuera de los estándares.
- Reducción de atributos redundantes.
  - Al heredar “DiagonalMatrix” (en el Modelo del Dominio) de “MatrixDouble” se eliminaron sus atributos redundantes (quedando sólo los heredados). Al parecer, se encontró que “BandedMatrix” tenía dos atributos redundantes: “rows” y “columns”.
- Decisión de si un elemento es clase o atributo.
  - El atributo “isInferior” (debería ser “isLowerMatrix”) representaba en realidad a dos subclases: “LowerMatrix” y “UpperMatrix”. Al sustituirlo por las clases se puede eliminar la lógica tipo “switch-case” y simplificar el código.
- Agrupación más adecuada (de sentido común).
  - Dentro del paquete se tenían programas de prueba: “MatrixMultiply” y “TestBanded”.
- Crítica de la necesidad de atributos.
  - En el diseño anterior, en “MatrixDouble”, los atributos “foundRow” y “foundColumn” parecen (de primera vista) no estar justificados. Usualmente, deberían ser la salida de un método. También se cuestiona la conveniencia de

“squared” en “AbstractMatrix” pues en varias clases heredadas esto es por definición (“SquaredMatrix”, “DiagonalMatrix”)

En resumidas cuentas, nos ayuda a reflexionar y modularizar nuestro diseño de forma más ordenada.

Lo que diferencia las clases heredadas de “MatrixDouble” es el manejo de la memoria. Esto queda dentro de ciertos métodos principalmente en los constructores y accesores (“get()”, “set()”).

7.1 ObjectID

Cuando sea obvio que los atributos no corresponden pero las clases sí, se recomienda agregar el atributo por omisión “(ObjectID)” para establecer la correspondencia. Este fenómeno se presentó cuando se debía corresponder las Clases Conceptuales de “Function2D” con algunas Clases de Diseño heredadas de “MatrixDouble” como “Arithmetic” con “BandedMatrix”. En este caso, los atributos de las Clases Conceptuales eran para almacenamiento, mientras que las Clases de Diseño, no tenían atributos (pues son librerías de funciones), por lo que se optó usar “(ObjectID)” para establecer la correspondencia con el diseño anterior. Si se optara agregar este atributo a todas las clases, se incrementarían las discrepancias por Déficit y Exceso.

7.2 Correspondencia difícil

A veces, la correspondencia entre elementos puede ser difícil de detectar. Por ejemplo, la correspondencia entre atributos es más complicada cuando hay un cambio de herencia. En el diseño anterior, la Clase de Diseño sin herencia “DiagonalPos” tenía “index”, “value” y “size”. Estos atributos se eliminaron de la clase al ser heredados de “MatrixDouble”. Se tienen las siguientes posibilidades.

Tabla 7.1. La clase se puede representar por los atributos en conjunto. No se considera la herencia

	Class	DiagonalPos	DiagonalPos	DiagonalPos
Class	**Attribute**	index	value	size
DiagonalMatrix	(ObjectID)	1	1	1
MatrixDouble	elem			
MatrixDouble	memoryRows			
MatrixDouble	memoryColumns			

Tabla 7.2. La clase no se puede representar por los atributos. No se considera la herencia

	Class	DiagonalPos	DiagonalPos	DiagonalPos	DiagonalPos
Class	**Attribute**	(ObjectID)	index	value	size
DiagonalMatrix	(ObjectID)	1			
MatrixDouble	elem				
MatrixDouble	memoryRows				
MatrixDouble	memoryColumns				

Tabla 7.3. La clase se representa por los atributos de la superclase. Se considera la herencia

	Class	DiagonalPos	DiagonalPos	DiagonalPos	DiagonalPos
Class	**Attribute**	(ObjectID)	index	value	size
DiagonalMatrix	(ObjectID)	1			
MatrixDouble	elem			1	
MatrixDouble	memoryRows				1
MatrixDouble	memoryColumns				1

Aunque es la más simple (no agrega un atributo “(ObjectID)”), la primera es parcialmente correcta pues los atributos anteriores representan a la clase pero no hay una correspondencia en detalle. También, la segunda es parcialmente correcta aunque es simplista, no establece la correspondencia de los atributos, constituyendo una Discrepancia de Exceso, lo cual no es preciso. La última posibilidad es la más correcta, pues establece la correspondencia entre clases y entre los atributos heredados en detalle.

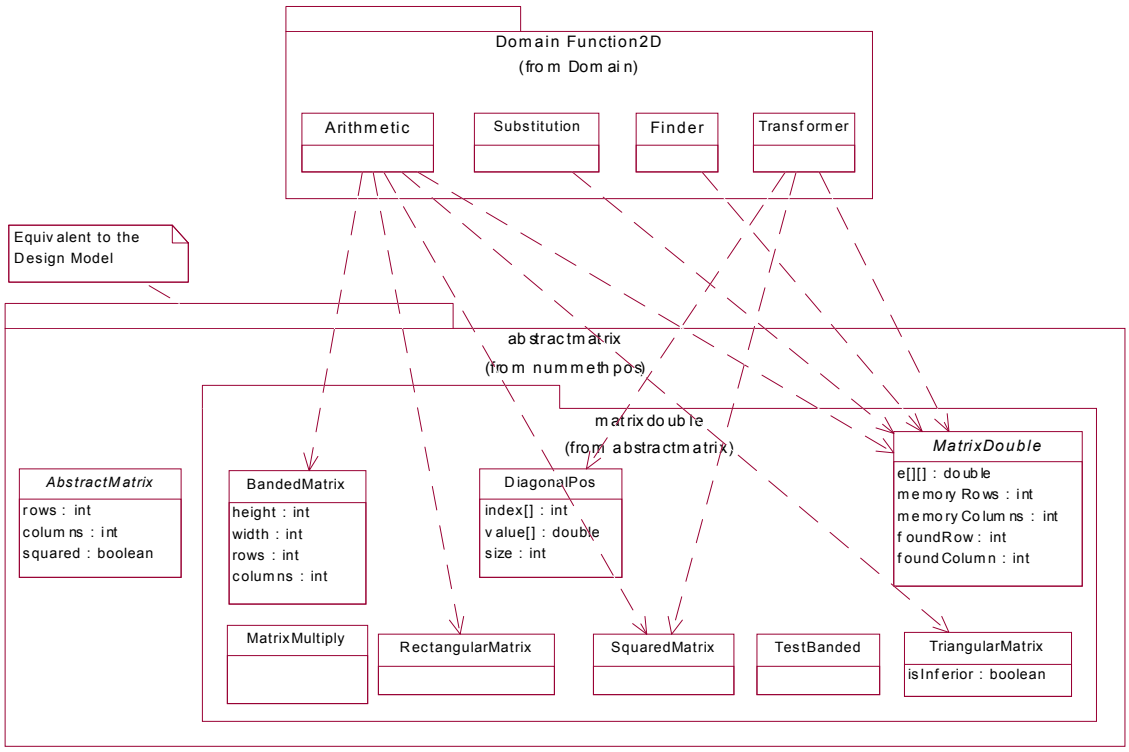


Figura 7.3. Refinamientos entre clases del paquete del dominio “Function2D” y el paquete del diseño anterior “abstractmatrix”

En este caso, la decisión que se soportó fue enfatizar la modularidad. Si bien, los métodos no se incluyen en el Modelo de Dominio, los métodos del Modelo del Diseño nos ayudaron a hacer la reingeniería del Modelo del Diseño formando el paquete de funciones de librería “Function2D”. El problema era que una clase estaba muy cargada de métodos que no en todos los casos fueran a ser usados, por lo que se separaron en clases de almacenamiento (“Matrix2D” con algunos métodos de acceso de datos –get, set- y de conversión básica) y de funcionalidad (“Function2D”).

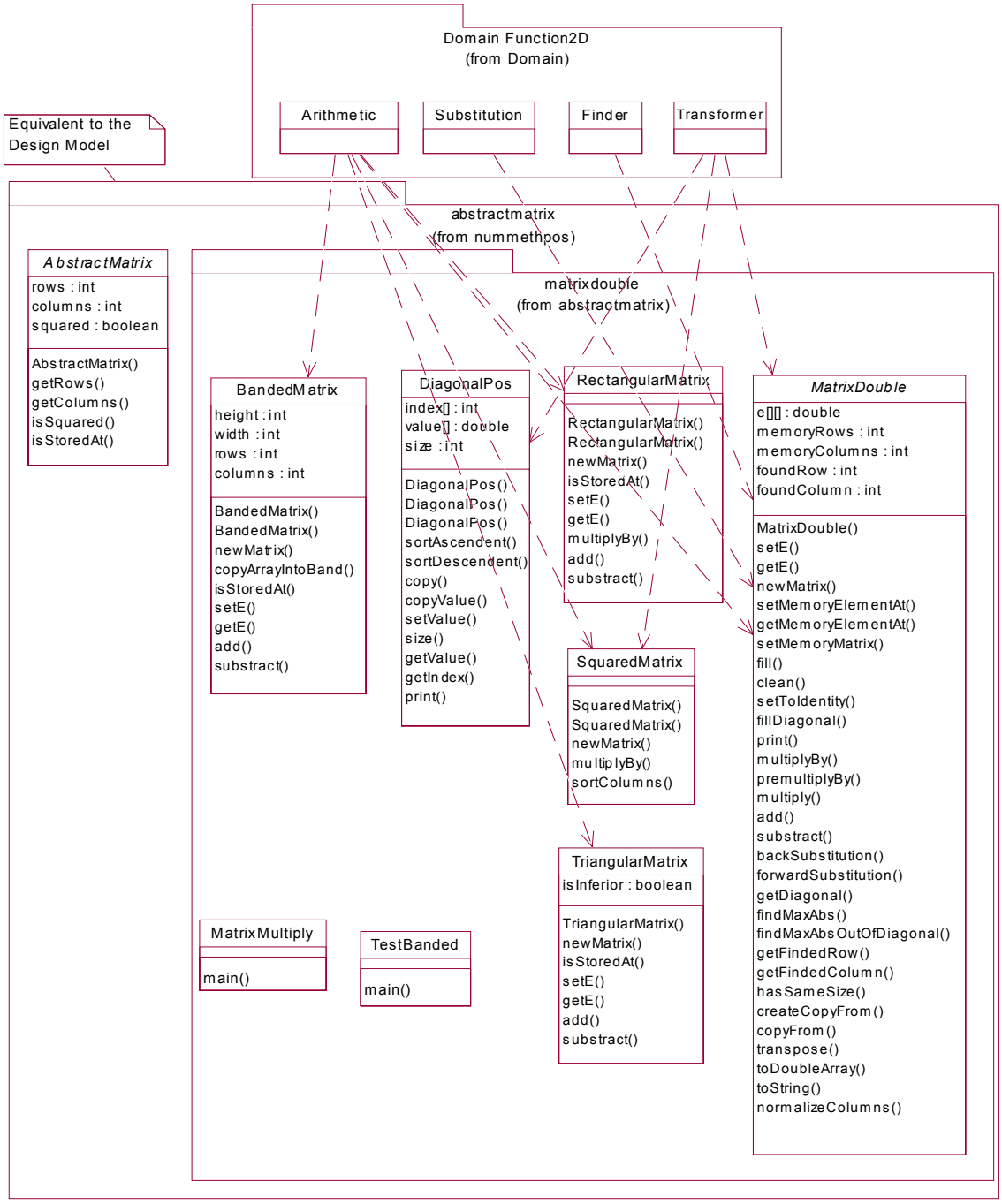


Figura 7.4. Métodos del paquete "abstractmatrix". Refinamientos entre clases del paquete del dominio "Function2D" y el paquete del diseño anterior "abstractmatrix"

Tabla 7.4. Matriz de refinamiento entre los paquetes del caso de matrices

Packages	OldDesign::abstractmatrix
Domain::Domain Matrix2D	1
Domain::Domain Function2D	1

Tabla 7.5. Matriz de refinamiento entre las clases del caso de matrices

Packages	Class	OldDesign::abstractmatrix	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble
		AbstractMatrix	BandedMatrix	DiagonalPos	MatrixDouble	MatrixMultiply	RectangularMatrix	SquaredMatrix	TestBanded	TriangularMatrix
Domain::Domain Matrix2D	BandedMatrix		1							
Domain::Domain Matrix2D	DiagonalMatrix			1						
Domain::Domain Matrix2D	LowerMatrix									1
Domain::Domain Matrix2D	MatrixDouble				1					
Domain::Domain Matrix2D	Matrix2D	1								
Domain::Domain Matrix2D	RectangularMatrix						1			
Domain::Domain Matrix2D	SquaredMatrix							1		
Domain::Domain Matrix2D	TriangularMatrix									1
Domain::Domain Matrix2D	UpperMatrix									1
Domain::Domain Function2D	Arithmetic		1		1		1	1		1
Domain::Domain Function2D	Substitution				1					
Domain::Domain Function2D	Finder				1					
Domain::Domain Function2D	Transformer			1	1			1		

Existen dos formas de medir el modelo

- Tomar las métricas para todo el modelo por elemento.
  - Relacionar todos los elementos (paquetes, clases, atributos) y obtener la métrica
- Ascender las métricas dependiendo del elemento.
  - Métodos
    - La correspondencia del elemento inferior no afecta al superior
    - La correspondencia del elemento inferior no afecta al superior
      - El valor de la métrica del elemento inferior (e.g. atributo) se coloca en la matriz superior (e.g. clases)
      - Esto puede ser complicado pero ofrece detalle por nivel del elemento
  - Algoritmo
    - Medir la correspondencia de atributos sólo de las clases mezcladas.
    - Medir la correspondencia de clases sólo de los paquetes mezclados.
    - Medir normalmente la correspondencia de paquetes

Tabla 7.6. Matriz de refinamiento entre los atributos del caso de matrices

		Packages	OldDesign::abstractmatrix	OldDesign::abstractmatrix	OldDesign::abstractmatrix		OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble		OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble		OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble	OldDesign::abstractmatrix::matrixdouble		OldDesign::abstractmatrix::matrixdouble
		Class	AbstractMatrix	AbstractMatrix	AbstractMatrix	BandedMatrix	BandedMatrix	BandedMatrix	BandedMatrix	BandedMatrix	DiagonalPos	DiagonalPos	DiagonalPos	DiagonalPos	MatrixDouble	MatrixDouble	MatrixDouble	MatrixDouble	MatrixDouble	MatrixDouble	MatrixMultiply	RectangularMatrix	SquaredMatrix	TestBanded	TriangularMatrix	TriangularMatrix
Packages	**Class**	**Attribute**	rows	columns	squared	(ObjectID)	height	width	rows	columns	(ObjectID)	index	value	size	(ObjectID)	e	memoryRows	memoryColumns	foundRow	foundColumn	(ObjectID)	(ObjectID)	(ObjectID)	(ObjectID)	(ObjectID)	isInferior
Domain::Domain Matrix2D	BandedMatrix	height					1																			
Domain::Domain Matrix2D	BandedMatrix	width						1																		
Domain::Domain Matrix2D	DiagonalMatrix	(ObjectID)									1															
Domain::Domain Matrix2D	MatrixDouble	elem											1			1										
Domain::Domain Matrix2D	MatrixDouble	memoryRows												1			1									
Domain::Domain Matrix2D	MatrixDouble	memoryColumns												1				1								
Domain::Domain Matrix2D	Matrix2D	rows	1																							
Domain::Domain Matrix2D	Matrix2D	columns		1																						
Domain::Domain Matrix2D	RectangularMatrix	(ObjectID)																				1				
Domain::Domain Matrix2D	SquaredMatrix	(ObjectID)																					1			
Domain::Domain Matrix2D	TriangularMatrix	(ObjectID)																							1	
Domain::Domain Matrix2D	LowerMatrix	(ObjectID)																								1
Domain::Domain Matrix2D	UpperMatrix	(ObjectID)																								1
Domain::Domain Function2D	Arithmetic	(ObjectID)				1									1							1	1		1	
Domain::Domain Function2D	Substitution	(ObjectID)													1											
Domain::Domain Function2D	Finder	(ObjectID)													1				1	1						
Domain::Domain Function2D	Transformer	(ObjectID)									1				1								1			

7.3 Identificación de las Discrepancias

Tabla 7.7. Identificación entre paquetes

	OldDesign.abstractmatrix	**Def**
Domain_Matrix2D	O2	-
Domain_Function2D	O2	-
Exc	-	

Tabla 7.8. Identificación entre clases

	AbstractMatrix	BandedMatrix	DiagonalPos	MatrixDouble	MatrixMultiply	RectangularMatrix	SquaredMatrix	TestBanded	TriangularMatrix	**Def**
BandedMatrix	-	O2	-	-	-	-	-	-	-	-
DiagonalMatrix	-	-	O2	-	-	-	-	-	-	-
LowerMatrix	-	-	-	-	-	-	-	-	O4	-
MatrixDouble	-	-	-	O5	-	-	-	-	-	-
Matrix2D	1To1	-	-	-	-	-	-	-	-	-
RectangularMatrix	-	-	-	-	-	O2	-	-	-	-
SquaredMatrix	-	-	-	-	-	-	O3	-	-	-
TriangularMatrix	-	-	-	-	-	-	-	-	O4	-
UpperMatrix	-	-	-	-	-	-	-	-	O4	-
Arithmetic	-	O2R5	-	O5R5	-	O2R5	O3R5	-	O4R5	-
Substitution	-	-	-	O5	-	-	-	-	-	-
Finder	-	-	-	O5	-	-	-	-	-	-
Transformer	-	-	O2R3	O5R3	-	-	O3R3	-	-	-
Exc	-	-	-	-	E	-	-	E	-	

Tabla 7.9. Identificación entre atributos. Parte 1 de 3

	AbstractMatrix.OID	AbstractMatrix.rows	AbstractMatrix.columns	AbstractMatrix.squared	BandedMatrix.OID	BandedMatrix.height	BandedMatrix.width	BandedMatrix.rows	BandedMatrix.columns
BandedMatrix_OID	-	-	-	-	O2	-	-	-	-
BandedMatrix_height	-	-	-	-	-	1To1	-	-	-
BandedMatrix_width	-	-	-	-	-	-	1To1	-	-
DiagonalMatrix_OID	-	-	-	-	-	-	-	-	-
MatrixDouble_OID	-	-	-	-	-	-	-	-	-
MatrixDouble_elem	-	-	-	-	-	-	-	-	-
MatrixDouble_memoryRows	-	-	-	-	-	-	-	-	-
MatrixDouble_memoryColumns	-	-	-	-	-	-	-	-	-
Matrix2D_OID	1To1	-	-	-	-	-	-	-	-
Matrix2D_rows	-	1To1	-	-	-	-	-	-	-
Matrix2D_columns	-	-	1To1	-	-	-	-	-	-
RectangularMatrix_OID	-	-	-	-	-	-	-	-	-
SquaredMatrix_OID	-	-	-	-	-	-	-	-	-
TriangularMatrix_OID	-	-	-	-	-	-	-	-	-
LowerMatrix_OID	-	-	-	-	-	-	-	-	-
UpperMatrix_OID	-	-	-	-	-	-	-	-	-
Arithmetic_OID	-	-	-	-	O2R5	-	-	-	-
Substitution_OID	-	-	-	-	-	-	-	-	-
Finder_OID	-	-	-	-	-	-	-	-	-
Transformer_OID	-	-	-	-	-	-	-	-	-
Exc	-	-	-	E	-	-	-	E	E

Tabla 7.10. Identificación entre atributos. Parte 2 de 3

	DiagonalPos.OID	DiagonalPos.index	DiagonalPos.value	DiagonalPos.size	MatrixDouble.OID	MatrixDouble.e	MatrixDouble.memoryRows	MatrixDouble.memoryColumns	MatrixDouble.foundRow	MatrixDouble.foundColumn	MatrixMultiply.OID
BandedMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
BandedMatrix_height	-	-	-	-	-	-	-	-	-	-	-
BandedMatrix_width	-	-	-	-	-	-	-	-	-	-	-
DiagonalMatrix_OID	O3	-	-	-	-	-	-	-	-	-	-
MatrixDouble_OID	O3R2	-	-	-	O5R2	-	-	-	-	-	-
MatrixDouble_elem	-	-	R2	-	-	R2	-	-	-	-	-
MatrixDouble_memoryRows	-	-	-	O2R2	-	-	R2	-	-	-	-
MatrixDouble_memoryColumns	-	-	-	O2R2	-	-	-	R2	-	-	-
Matrix2D_OID	-	-	-	-	-	-	-	-	-	-	-
Matrix2D_rows	-	-	-	-	-	-	-	-	-	-	-
Matrix2D_columns	-	-	-	-	-	-	-	-	-	-	-
RectangularMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
SquaredMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
TriangularMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
LowerMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
UpperMatrix_OID	-	-	-	-	-	-	-	-	-	-	-
Arithmetic_OID	-	-	-	-	O5R5	-	-	-	-	-	-
Substitution_OID	-	-	-	-	O5	-	-	-	-	-	-
Finder_OID	-	-	-	-	O5R3	-	-	-	R3	R3	-
Transformer_OID	O3R3	-	-	-	O5R3	-	-	-	-	-	-
Exc	-	E	-	-	-	-	-	-	-	-	E

Tabla 7.11. Identificación entre atributos. Parte 3 de 3

	RectangularMatrix.OID	SquaredMatrix.OID	TestBanded.OID	TriangularMatrix.OID	TriangularMatrix.isInferior	**Def**
BandedMatrix_OID	-	-	-	-	-	-
BandedMatrix_height	-	-	-	-	-	-
BandedMatrix_width	-	-	-	-	-	-
DiagonalMatrix_OID	-	-	-	-	-	-
MatrixDouble_OID	-	-	-	-	-	-
MatrixDouble_elem	-	-	-	-	-	-
MatrixDouble_memoryRows	-	-	-	-	-	-
MatrixDouble_memoryColumns	-	-	-	-	-	-
Matrix2D_OID	-	-	-	-	-	-
Matrix2D_rows	-	-	-	-	-	-
Matrix2D_columns	-	-	-	-	-	-
RectangularMatrix_OID	O2	-	-	-	-	-
SquaredMatrix_OID	-	O3	-	-	-	-
TriangularMatrix_OID	-	-	-	O2	-	-
LowerMatrix_OID	-	-	-	-	O2	-
UpperMatrix_OID	-	-	-	-	O2	-
Arithmetic_OID	O2R5	O3R5	-	O2R5	-	-
Substitution_OID	-	-	-	-	-	-
Finder_OID	-	-	-	-	-	-
Transformer_OID	-	O3R3	-	-	-	-
Exc	-	-	E	-	-	

7.4 Métricas de la Brecha

Para fines de comparación, en la Tabla 7.12, Tabla 7.13 y Tabla 7.14 se muestran los resúmenes de las métricas de la Brecha entre paquetes, clases y atributos. Dependiendo de lo que se deseé analizar, se deberá seleccionar la métrica, para recordar estos detalles, ver “Interpretación de la métrica” en la página 112. Para tener una visión de la magnitud de la Brecha, convendría concentrarse en las métricas del Modelo del conteo, razón y ponderada.

Tabla 7.12. Resumen de métricas de la Brecha entre paquetes[40]

Discrepancia	Contar	Máximo	Razón	Ponderada	Contribución
Déficit	0	2	0%	0%	0%
Exceso	0	1	0%	0%	0%
Sobrecarga	1	1	100%	60%	86%
Redundancia	0	2	0%	0%	0%
Unión	*2*	*2*	*100%*	10%	14%
Modelo	**1**		**50%**	**70%**	**100%**

En la Tabla 7.12, se muestra que hay pocos paquetes y por ende el rango es reducido (i.e. cero, uno o dos Discrepancias), por ello, las razones parecen binarias.. La métrica ponderada por tipo de Discrepancia es un reflejo casi directo de los pesos relativos asignados. Aunque parezca muy simple eliminar esta Discrepancia, hay que recordar que se trata de paquetes por lo que se tendrían que cambiar algunas clases y posiblemente sus atributos. A pesar de que son pocos paquetes, cabe notar sólo una Discrepancia puede indicar una brecha amplia (70%), lo que debería de tenerse en cuenta.

Tabla 7.13. Resumen de métricas de la Brecha entre clases

Discrepancia	Contar	Máximo	Razón	Ponderada	Contribución
Déficit	0	13	0%	0%	0%
Exceso	2	9	22%	7%	14%
Sobrecarga	6	9	67%	40%	75%
Redundancia	2	13	15%	5%	9%
Unión	*18*	*117*	*15%*	2%	3%
Modelo	**10**		**52%**	**54%**	**100%**

Si bien, el número de clases es mayor que el número de paquetes en la Tabla 7.13, la métrica de la razón del Modelo entre paquetes y entre clases es similar. La métrica de la razón de la Brecha es similar a la ponderada, debido a los pesos asignados (recordar que la métrica de la razón es un caso especial de la métrica ponderada).

[40] Las métricas de la Unión en itálicas no se toman en cuentan en los cálculos para las métricas del Modelo.

Tabla 7.14. Resumen de métricas de la Brecha entre atributos

Discrepancia	Contar	Máximo	Razón	Ponderada	Contribución
Déficit	0	20	0%	0%	0%
Exceso	6	25	24%	8%	21%
Sobrecarga	8	25	32%	19%	50%
Redundancia	7	20	35%	11%	27%
Unión	*27*	*500*	*5%*	1%	1%
Modelo	**21**		**46%**	**38%**	**100%**

De forma similar, en la Tabla 7.14, se tiene que el número de atributos es mayor que el número de clases. Esto es muy natural pues usualmente, los paquetes se componen de clases y estos de atributos. A pesar de que el número de Discrepancias entre los atributos es aproximadamente el doble que entre las clases, la métrica ponderada del Modelo es menor entre los atributos que entre las clases. Esto debe de señalar la importancia de una correcta asignación de pesos.

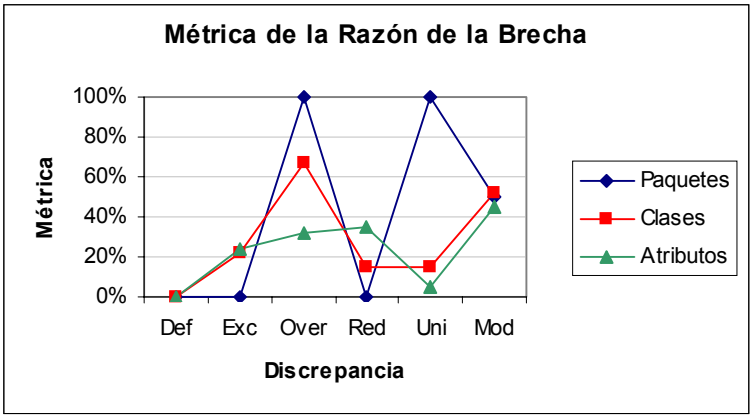


Figura 7.5. Comparación de métricas de la razón entre elementos seleccionados

Para tener un mejor entendimiento, en la Figura 7.5, se grafican las métricas de la razón de la Brecha, para comparar el comportamiento por el tipo de Discrepancia. Aunque se necesitarían más datos de varios proyectos, parece que hay una cierta similitud entre las métricas de los elementos (excepto para la métrica de la Unión entre paquetes que en este caso es binaria).

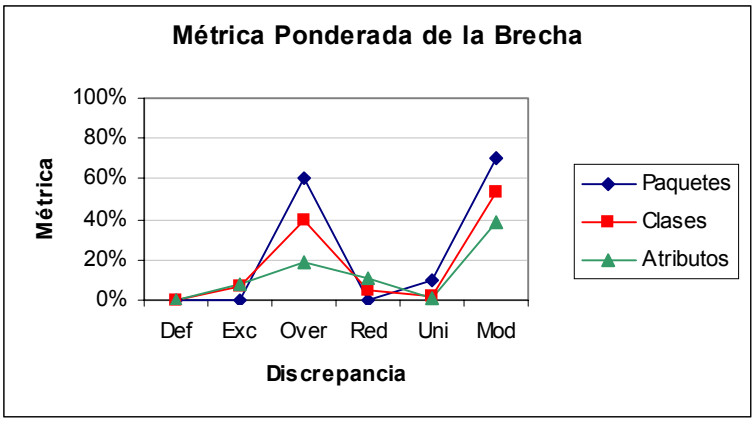


Figura 7.6. Comparación de métricas ponderadas entre elementos seleccionados

El fenómeno de los comportamientos similares de las métricas ponderadas de los elementos seleccionados (i.e. paquetes, clases y atributos) se resalta porque los pesos asignados condicionan estos valores. Por el mayor peso de la Sobrecarga (60%) sobre la Redundancia (30%) y la Unión (10%), se resaltan las diferencias entre las métricas de los elementos seleccionados. Este hecho se aprecia en la Figura 7.6.

La causa más razonable para explicar la similitud entre las métricas es la composición jerárquica de los elementos seleccionados (i.e. un paquete se compone de varias clases que a su vez se componen de varios atributos). Sin embargo, se necesitan muchos más datos para asegurar este fenómeno. Lo importante aquí es que se puede formar más conocimiento (i.e. relaciones empíricas) a partir de estas métricas.

Capítulo 8. Caso de estudio escolar: un sistema de administración escolar

El presente caso de estudio es ficticio y pretende mostrar los conceptos expuestos, más que detallar el Dominio Escolar. Por ello, se pueden detectar fácilmente muchas faltas de funcionalidad dado que es una simplificación de la realidad. Además, se introdujeron errores intencionalmente para obviar las correcciones en la siguiente iteración.

El objetivo en este caso de estudio es estudiar el comportamiento de las métricas a través de las iteraciones. Para cada iteración, se muestran los casos de uso, el Modelos de Dominio y el Modelo de Diseño, así como algunas notas sobre el desarrollo y el establecimiento de la correspondencia (como una forma breve de documentación).

8.1 Notas

- En el Modelo del Diseño, las clases se referirán a tablas relacionales.
- Los nombres de algunos elementos del modelado (como los diagramas y paquetes) tienen los sufijos E1..n para referirse a la iteración de la Fase de Elaboración. En el diagrama de casos de uso, la iteración E1 incluye a I1 (Iniciación 1). Rational Rose advierte el tener mismos nombres de elementos en espacios de nombrado múltiples (i.e. tener una clase "Alumno" en "Dominio E1" con distintos atributos que la clase "Alumno" en "Dominio E2").
- Las dependencias en todos los diagramas de clases por defecto tienen el estereotipo <<refine>> para tener diagramas más legibles.

8.1.1 DDL

- Bajo la generación de DDL (tablas de SQL), las Clases Conceptuales son todas transitivas (no se generan tablas de ellas); y se espera que la mayoría de las Clases de Diseño sean persistentes.
- En Rational Rose, las clases persistentes tienen una llave primaria ("primary key") que se forma con el nombre de la clase (tabla) y el sufijo "Id". Por ejemplo, la clase sin llave primaria definida Profesor, tendrá por omisión, la llave primaria ProfesorId.

8.2 Correspondencia entre paquetes de distinto nivel

Hay que tener cuidado al establecer una dependencia entre paquetes de distintos niveles, puede ser muy confuso, no se recomienda hacerlos. Sería mucho mejor, establecer la correspondencia por cada nivel de anidamiento de los paquetes. En su defecto, sólo establecer las correspondencias de los paquetes externos (de alto nivel).

8.3 Elaboración 1

8.3.1 Casos de uso textuales

UC1. El profesor registra las calificaciones de sus alumnos en el sistema
UC2. El profesor consulta las calificaciones de los alumnos inscritos en sus materias en el sistema
UC3. El alumno consulta las calificaciones de las materias inscritos en el sistema
UC4. El administrador escolar pueden consultar las calificaciones de todos los alumnos en el sistema
UC5. El administrador escolar corrige las calificaciones de forma extraordinaria en el sistema

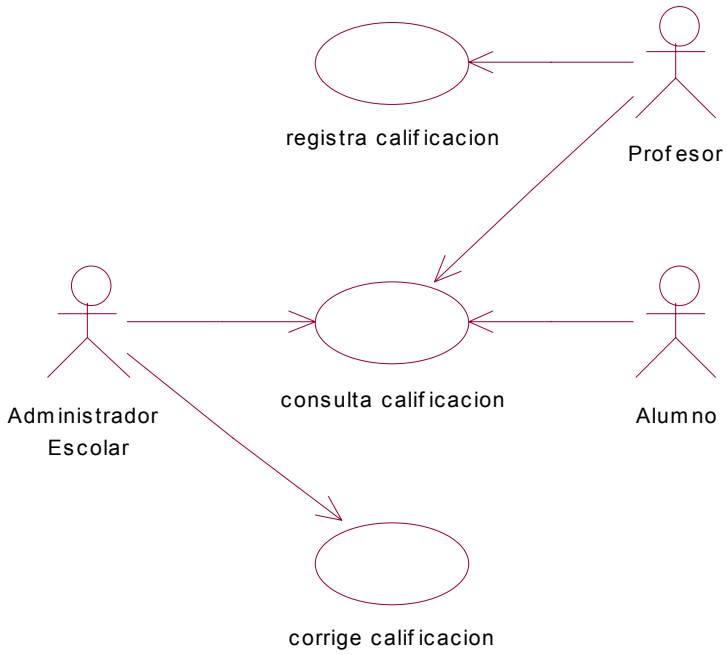


Figura 8.1. Diagrama de casos de uso (Escolar). Elaboración 1

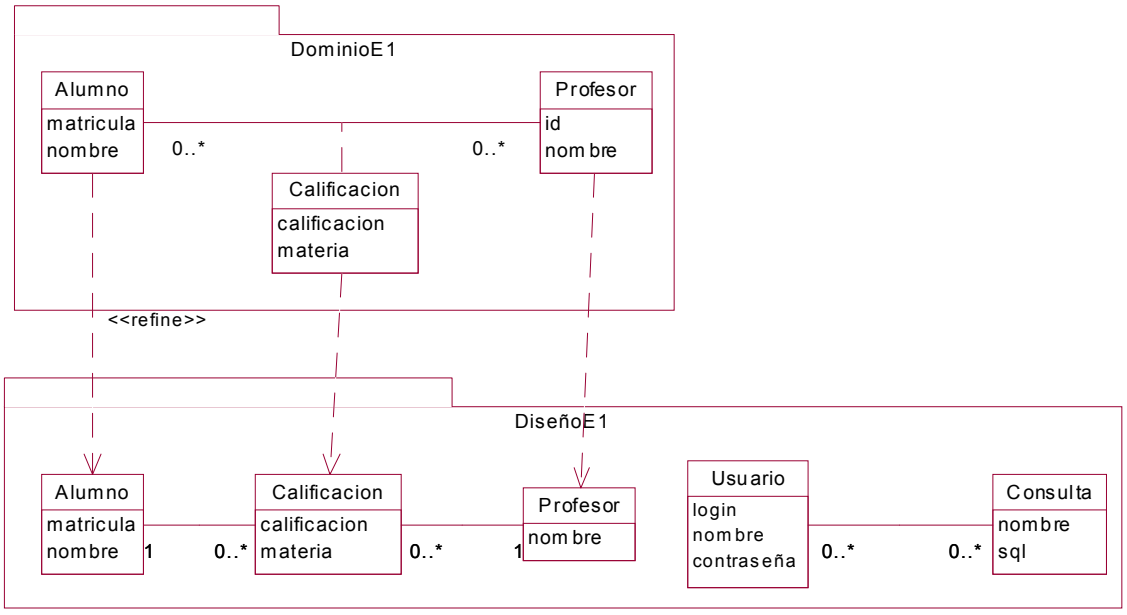


Figura 8.2. Diagrama de clases del Dominio y el Diseño. Elaboración 1

Notas

- Ingenuamente, "nombre" incluye el nombre de pila y los apellidos. El administrador escolar, menciona que las ordenaciones son por el apellido paterno (o primer apellido). No se encontró razón para tener los apellidos separados. De hecho, había personas que no deseaban poner apellido paterno. Se optó por separarlos en "nombre" y "apellidos".
- Al ver que este fenómeno ocurría de igual forma para las clases de Profesor y Usuario, se decide crear la clase Persona. Esto permite asignar un identificador único para todas las personas, lo cual implica a la organización eliminar un identificador para distintos tipos de personas; lo que supone más caracteres de representación, pero un sólo tipo de personas, con lo que se reduce la redundancia de información. Un beneficio adicional fue la reducción del número de credenciales. La "matricula" del Alumno se sustituye por "id" (e.g. CURP, Clave Única de Registro de Población).
- Hay profesores que son también alumnos.
- Hay alumnos que se inscriben en varios programas académicos
- Se considera alumno al que cursa una materia "libre", aunque no necesariamente esté escrito en un programa académico.
- Una materia "libre" puede no tener un programa académico.
- Hay materias cursadas y por cursar
- El administrador escolar autoriza la equivalencia de una materia con la de una materia del programa académico, siempre y cuando, tengan los mismo créditos.
- La actual estructura necesita que se repita la materia para cada pareja de Alumno y Profesor, por lo que se crea la clase Materia.
- Los derechos de acceso a la información deben de estar centralizados para las consultas válidas. Los componentes para la interfaz del usuario, deberá consultar estas tablas

Correspondencia

- Al localizar las Excesos con las clases Usuario y Consulta, se opta por aislar estas Discrepancias de clases en un paquete pasando a ser un Exceso en los paquetes, y no de clases.
- La correspondencia entre asociaciones depende del enfoque de la implementación (en este caso bases de datos relacionales). Además, algunas características de las asociaciones no son implementables en una cierta tecnología. Por ejemplo, una tabla no puede contener a otra internamente (agregación). De forma similar, una asociación muchos a muchos no es implementable directamente; se necesita crear una tabla intermedia (un patrón común).

Tabla 8.1. Matriz de correspondencia de paquetes del caso escolar. Elaboración 1

Paquetes	Diseño E1
Dominio E1	1

Tabla 8.2. Matriz de correspondencia de clases del caso escolar. Elaboración 1

	Paquetes	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1
Paquetes	**Clase**	Alumno	Calificacion	Profesor	Usuario	Consulta
Dominio E1	Alumno	1				
Dominio E1	Calificacion		1			
Dominio E1	Profesor			1		

Tabla 8.3. Matriz de correspondencia de atributos del caso escolar. Elaboración 1

		Paquetes	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1	Diseño E1
		Clase	Alumno	Alumno	Calificacion	Calificacion	Profesor	Profesor	Usuario	Usuario	Usuario	Consulta	Consulta
Paquetes	**Clase**	**Atributo**	matricula	nombre	calificacion	materia	id	nombre	login	nombre	contraseña	nombre	sql
Dominio E1	Alumno	matricula	1										
Dominio E1	Alumno	nombre		1									
Dominio E1	Calificacion	calificacion			1								
Dominio E1	Calificacion	materia				1							
Dominio E1	Profesor	id					1						
Dominio E1	Profesor	nombre						1					

8.4 Elaboración 2

8.4.1 Casos de uso textuales

UC1. El profesor registra las calificaciones de sus alumnos en el sistema

UC2. El profesor consulta las calificaciones de los alumnos inscritos en sus materias en el sistema

UC3. El alumno consulta las calificaciones de las materias inscritos en el sistema

UC4. El administrador escolar pueden consultar las calificaciones de todos los alumnos en el sistema

UC5. El administrador escolar corrige las calificaciones de forma extraordinaria en el sistema

*UC6. El administrador escolar asigna las materias de cada programa académico en el sistema

*UC7. El alumno ordena los programas académicos conforme a su preferencia. El sistema asigna el primer programa académico posible en base a su puntaje de admisión. El administrador escolar confirma con el alumno.

*UC8. El administrador escolar y el profesor deciden las materias a impartir en el periodo

*UC9. El administrador escolar y el alumno deciden las materias a cursar en el periodo

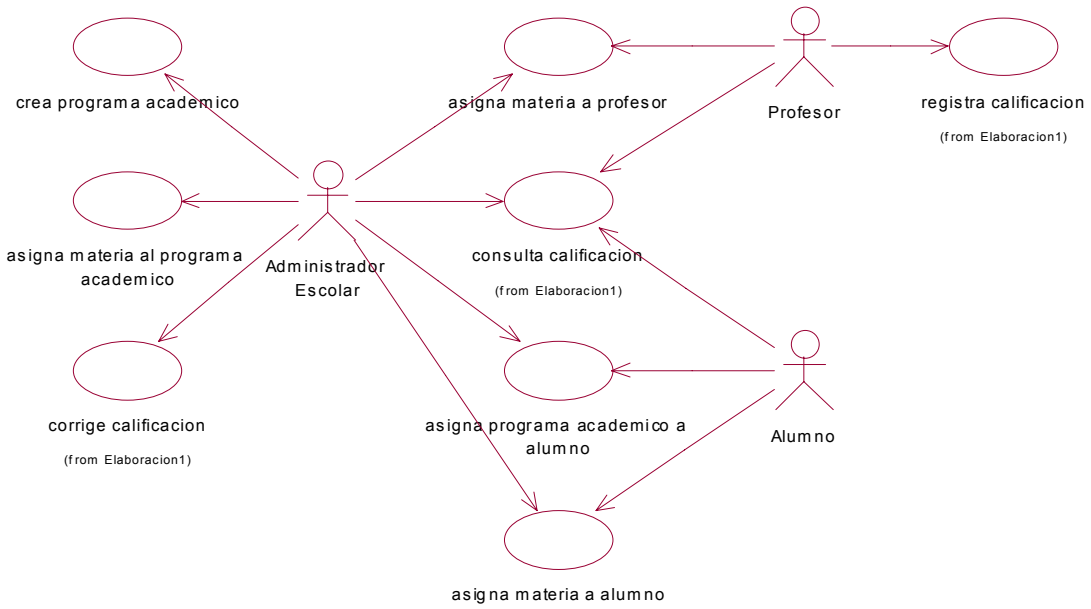


Figura 8.3. Diagrama de casos de uso del caso escolar. Elaboración 2

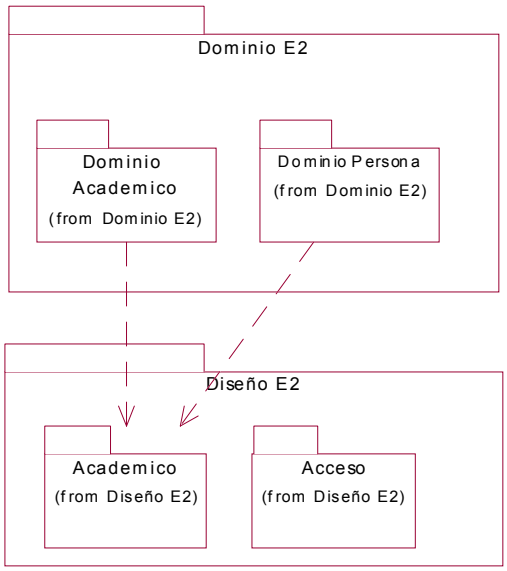


Figura 8.4. Diagrama de clases y correspondencia entre los paquetes del Dominio y del Diseño del caso escolar. Elaboración 2

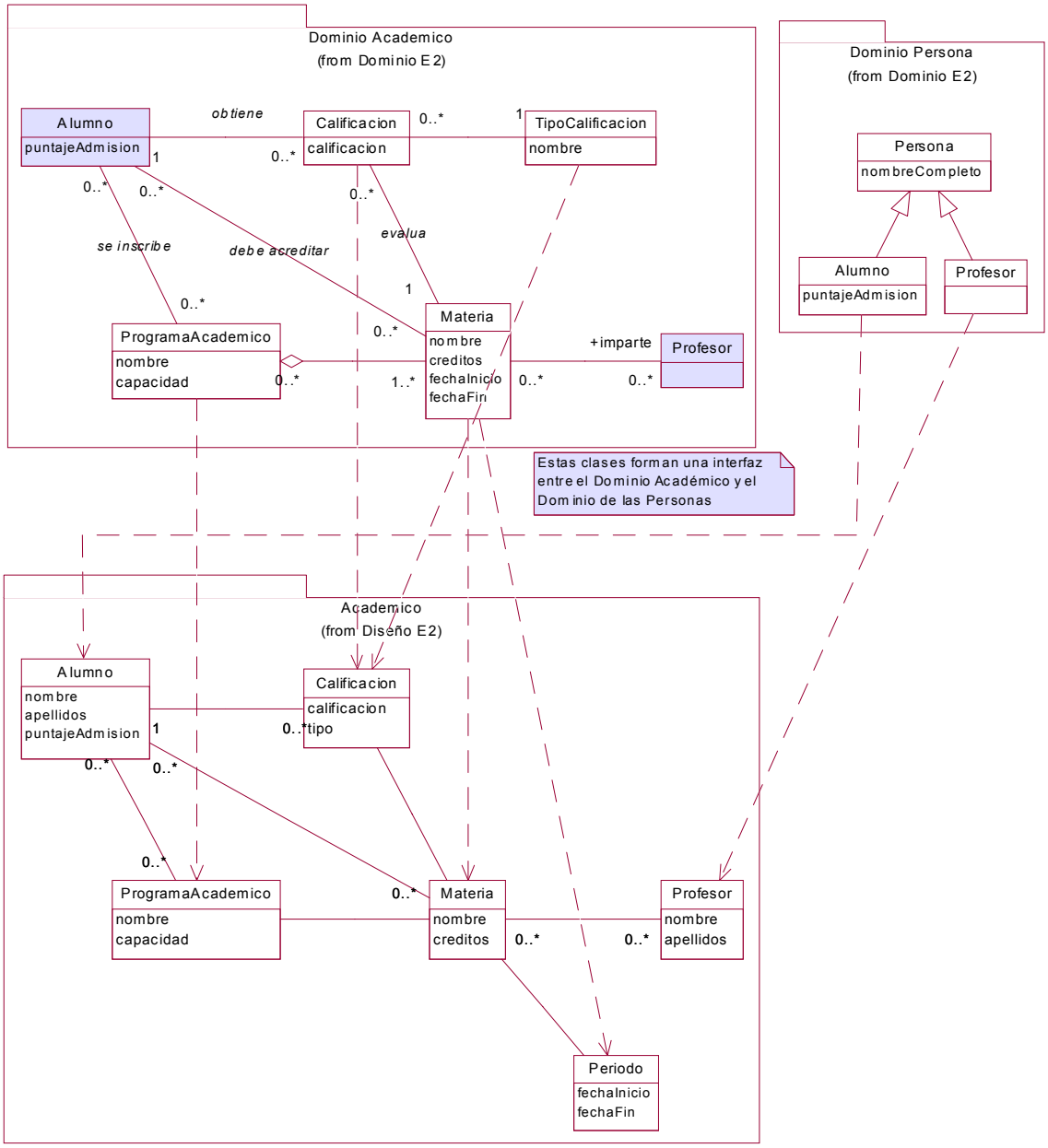


Figura 8.5. Diagrama de clases y correspondencia entre el Modelo del Dominio y el del Diseño del caso escolar. Elaboración 2[41]

[41] Las clases sombreadas son redundantes por cuestiones de legibilidad.

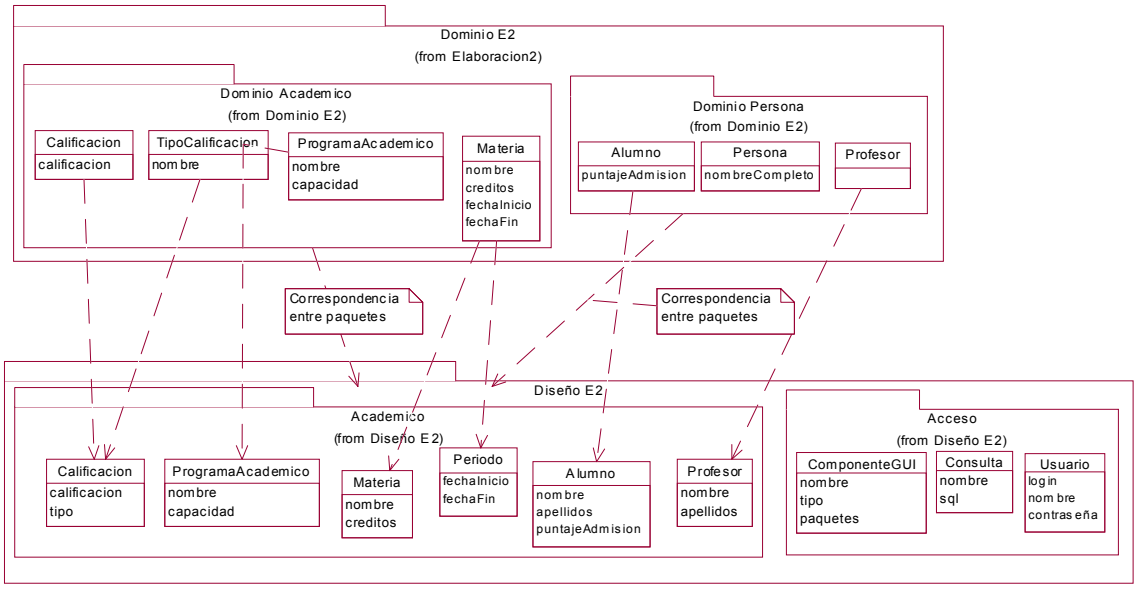


Figura 8.6. Correspondencia de paquetes y clases entre el Modelo del Dominio y el Modelo del Diseño del caso escolar. Diagrama de clases sin asociaciones. Elaboración 2

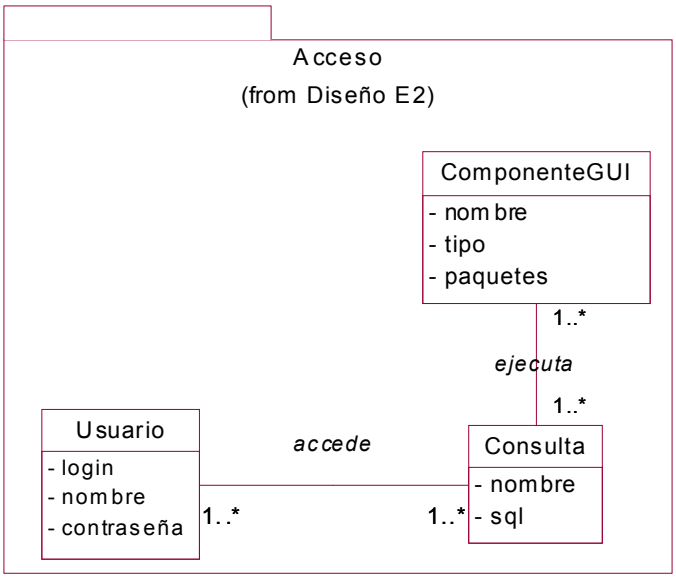


Figura 8.7. Paquete de diseño: Acceso. Elaboración 2

Notas

- En el diseño, por cuestiones de rendimiento y para reducir la indirección se optó por duplicar los miembros de Persona en Alumno y Profesor. Se optó por corregir periódicamente las inconsistencias entre las tablas heredadas.

- Dado que son pocos tipos y se espere que cambien poco, se optó por incluir TipoCalificacion en Calificacion.
- Dado que en la actualidad y en los próximos 5 años, se espera continuar con sólo tres periodos anuales: semestre de primavera, bimensual de verano y semestre de otoño; se optó por relacionar Materia con Periodo para poder listar las materias por un conjunto de periodos y evitar el error de excluir algunas materias por una mala especificación de las fechas en la consulta (e.g. al hacer una consulta del 1/Agosto/2005 al 15/Diciembre/2005, se excluirían a las materias que terminaran extemporáneamente, como el 10/Enero/2006). De momento, se consideró que esta precisión no era necesaria.
- Las materias para las que no se les ha asignado un curso, materias sin un periodo, simplemente se dejarán las fechas como nulas.
- Se optó por centralizar la habilitación/deshabilitación de los componentes de la interfaz gráfica. Por lo que se usarán componentes de librería desarrollados previamente para poder facilitar el control del acceso. Los componentes GUI enlistados deberán realizar una consulta, por lo que se excluyen los componentes exclusivamente de interfaz gráfica (de personalización visual, ventanas, etc.). Además se solicitó que todas las consultas pudieran ejecutarse desde el ambiente gráfico.
- Se vio la necesidad de conocer el domicilio de las personas.

Correspondencia

- Al parecer las asociaciones son muy similares entre el Modelo del Dominio y el del Diseño. Las Discrepancias de relaciones por regla son las mismas que las de clases. Por ende, nos concentraremos en las Discrepancias de las clases.
- Hay Discrepancias en las asociaciones que de momento, no parecen tener un gran impacto. Por ejemplo, la agregación en Dominio E2 entre ProgramaAcademico y Materia, se cambia a una relación de tablas entre ProgramaAcademico y Materia. Mientras estén asociadas con la misma multiplicidad, parecen no crear dificultades mayores.

Tabla 8.4. Matriz de correspondencia de paquetes. Elaboración 2

Paquetes	Diseño E2::Academico	Diseño E2::Acceso
Dominio E2::Dominio Academico	1	
Dominio E2::Dominio Persona	1	

Tabla 8.5. Matriz de correspondencia de clases. Elaboración 2

Paquetes	Clase	Calificacion	ProgramaAcademico	Materia	Periodo	Alumno	Profesor	ComponenteGUI	Consulta	Usuario
	Paquetes	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso
Dominio E2::Dominio Academico	Calificacion	1								
Dominio E2::Dominio Academico	TipoCalificacion	1								
Dominio E2::Dominio Academico	ProgramaAcademico		1							
Dominio E2::Dominio Academico	Materia			1	1					
Dominio E2::Dominio Persona	Alumno					1				
Dominio E2::Dominio Persona	Persona					1	1			
Dominio E2::Dominio Persona	Profesor						1			

Tabla 8.6. Matriz de correspondencia de atributos. Elaboración 2

Paquetes	Clase	Atributo	calificacion	tipo	nombre	capacidad	id	nombre	creditos	fechaInicio	fechaFin	id	nombre	apellidos	puntajeAdmision	id	nombre	apellidos	nombre	tipo	paquetes	nombre	sql	login	nombre	contraseña
		Paquetes	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Academico	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso	Diseño E2::Acceso
		Clase	Calificacion	Calificacion	ProgramaAcademico	ProgramaAcademico	Materia	Materia	Materia	Periodo	Periodo	Alumno	Alumno	Alumno	Alumno	Profesor	Profesor	Profesor	ComponenteGUI	ComponenteGUI	ComponenteGUI	Consulta	Consulta	Usuario	Usuario	Usuario
Dominio E2::Dominio Academico	Calificacion	calificacion	1																							
Dominio E2::Dominio Academico	TipoCalificacion	nombre		1																						
Dominio E2::Dominio Academico	ProgramaAcademico	nombre			1																					
Dominio E2::Dominio Academico	ProgramaAcademico	capacidad				1																				
Dominio E2::Dominio Academico	Materia	id					1																			
Dominio E2::Dominio Academico	Materia	nombre						1																		
Dominio E2::Dominio Academico	Materia	creditos							1																	
Dominio E2::Dominio Academico	Materia	fechaInicio								1																
Dominio E2::Dominio Academico	Materia	fechaFin									1															
Dominio E2::Dominio Persona	Alumno	puntajeAdmision													1											
Dominio E2::Dominio Persona	Persona	id										1				1										
Dominio E2::Dominio Persona	Persona	nombreCompleto											1	1			1	1								
Dominio E2::Dominio Persona	Profesor	(ObjectID)														1	1	1								

Para el caso en que una clase no tenga atributos, por defecto tendrá un identificador de la clase (ObjectID) como atributo. Este caso se aprecia entre la Clase Conceptual Profesor y la Clase de Diseño Profesor.

8.4.2 Relevancia del empleo a clases que a otros elementos

Creemos que es mejor obtener una métrica de forma general (con poco detalle, a grosso modo), que una métrica detallada guardada en el cajón. Por ello, la recomendación es obtener la correspondencia de clases. Si se desea mayor detalle con mayor tiempo de evaluación, se puede optar también la correspondencia de atributos.

8.5 Elaboración 3

En esta iteración, el analista y el diseñador trabajaron de forma separada por conflictos laborales. Luego, se separaron de la empresa. A consecuencia de ello, los siguientes diagramas fueron los productos con percepciones separadas de un mismo problema que el nuevo encargado tuvo que entender y hacer corresponder.

Al nuevo encargado se le dice que el analista era a veces muy detallista y con funcionalidad a futuro, mientras que el diseñador era más práctico y con la funcionalidad mínima. Se tiene noticia que el analista se tomo el tiempo para consultar algunos manuales sobre urbanismo, para las direcciones. Mientras que el diseñador mencionó que una estructura similar funcionó bien en otra aplicación.

8.5.1 Casos de uso textuales

UC1. El profesor registra las calificaciones de sus alumnos en el sistema
UC2. El profesor consulta las calificaciones de los alumnos inscritos en sus materias en el sistema
UC3. El alumno consulta las calificaciones de las materias inscritos en el sistema
UC4. El administrador escolar puede consultar las calificaciones de todos los alumnos en el sistema
UC5. El administrador escolar corrige las calificaciones de forma extraordinaria en el sistema
UC6. El administrador escolar asigna las materias de cada programa académico en el sistema
UC7. El alumno ordena los programas académicos conforme a su preferencia. El sistema asigna el primer programa académico posible en base a su puntaje de admisión. El administrador escolar confirma con el alumno.
UC8. El administrador escolar y el profesor deciden las materias a impartir en el periodo
UC9. El administrador escolar y el alumno deciden las materias a cursar en el periodo
*UC10. El administrador escolar solicita estadísticas sobre la proveniencia (lugar) de los alumnos por programa académico
*UC11. El administrador escolar solicita estadísticas sobre la proveniencia (lugar) de los profesores para envío de correspondencia y tener los datos generales.
*UC12. El administrador escolar envía mensajes (correspondencia) al alumno
*UC13. El administrador escolar envía mensajes (correspondencia) al profesor
*UC14. El administrador escolar solicita estadísticas por materia (del conocimiento)

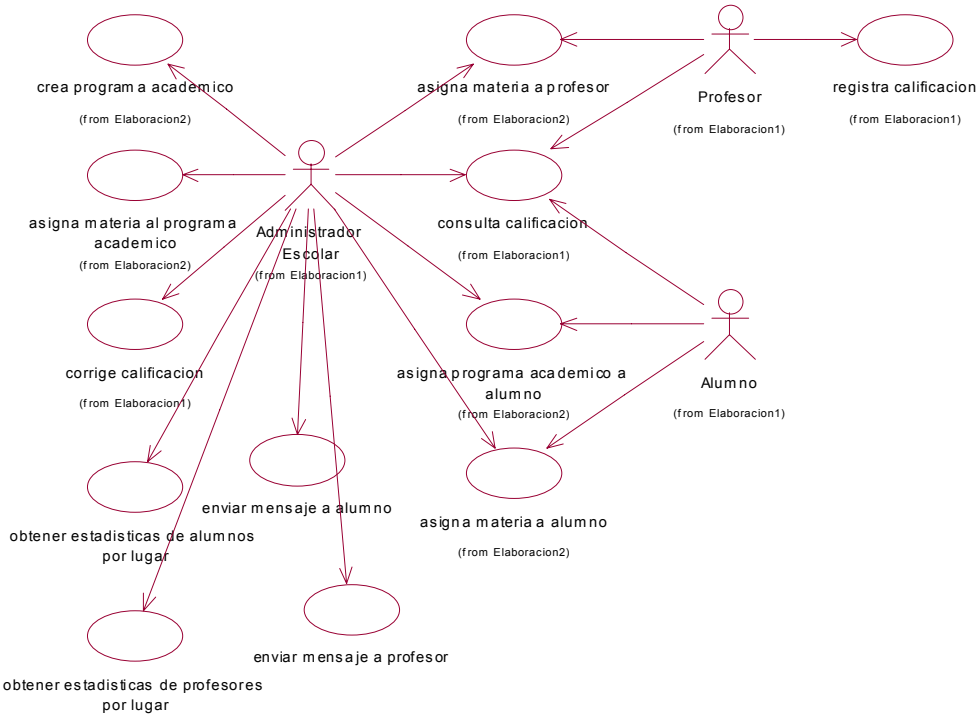


Figura 8.8. Diagrama de casos de uso del caso escolar. Elaboración 3

Notas

- Se visiona que a corto plazo, el enfoque centralizado de Persona es mejor para agrupar a otros tipos de personas como Empleado, Directivo, etc. A pesar de que las consultas se alarguen un poco.
- Por motivos de consistencia y estandarización, se clasificaron las materias como libros, según el Sistema de Clasificación de la “Biblioteca del Congreso”.
- El diseñador optó por un árbol extensible para materias en las que la clasificación fuera insuficiente, es decir se quisiera más detalle, como un capítulo de un libro. Además es posible que no se contara con un libro en especial, sino un área más general.

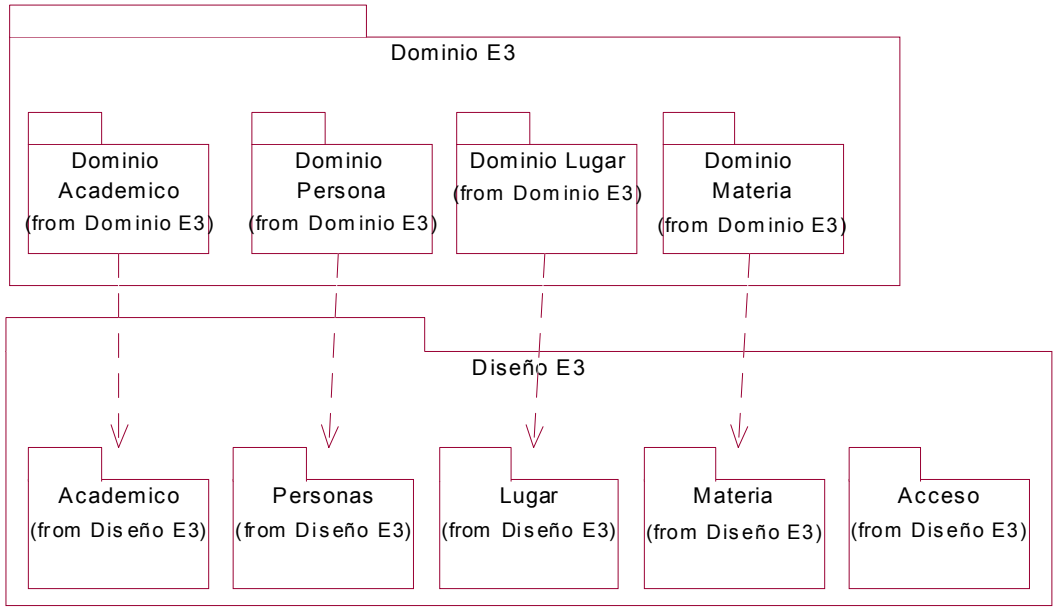


Figura 8.9. Refinamientos entre paquetes del caso escolar. Elaboración 3

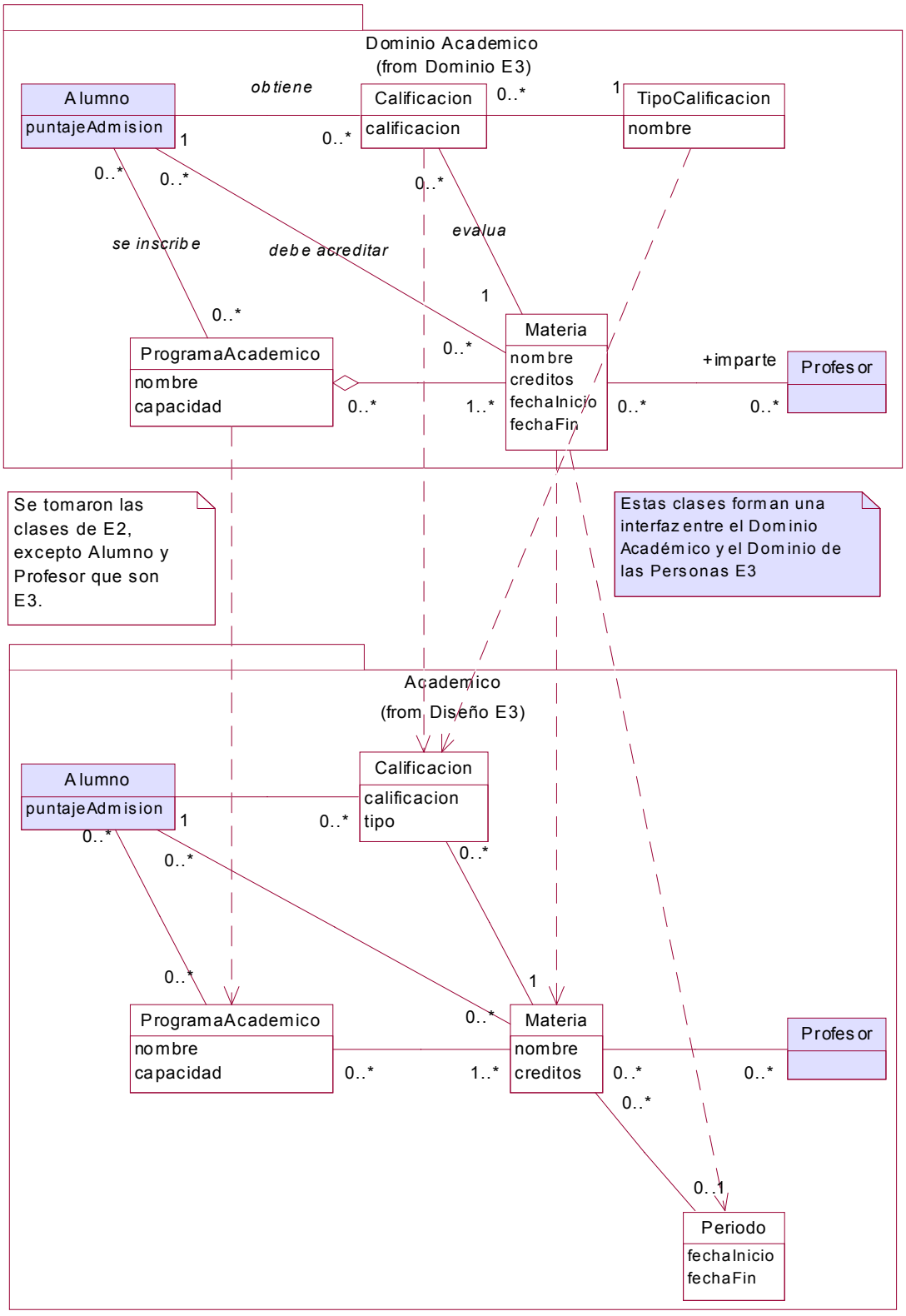


Figura 8.10. Refinamientos entre clases de los paquetes "Dominio Académico" y "Académico" del caso escolar. Elaboración 3

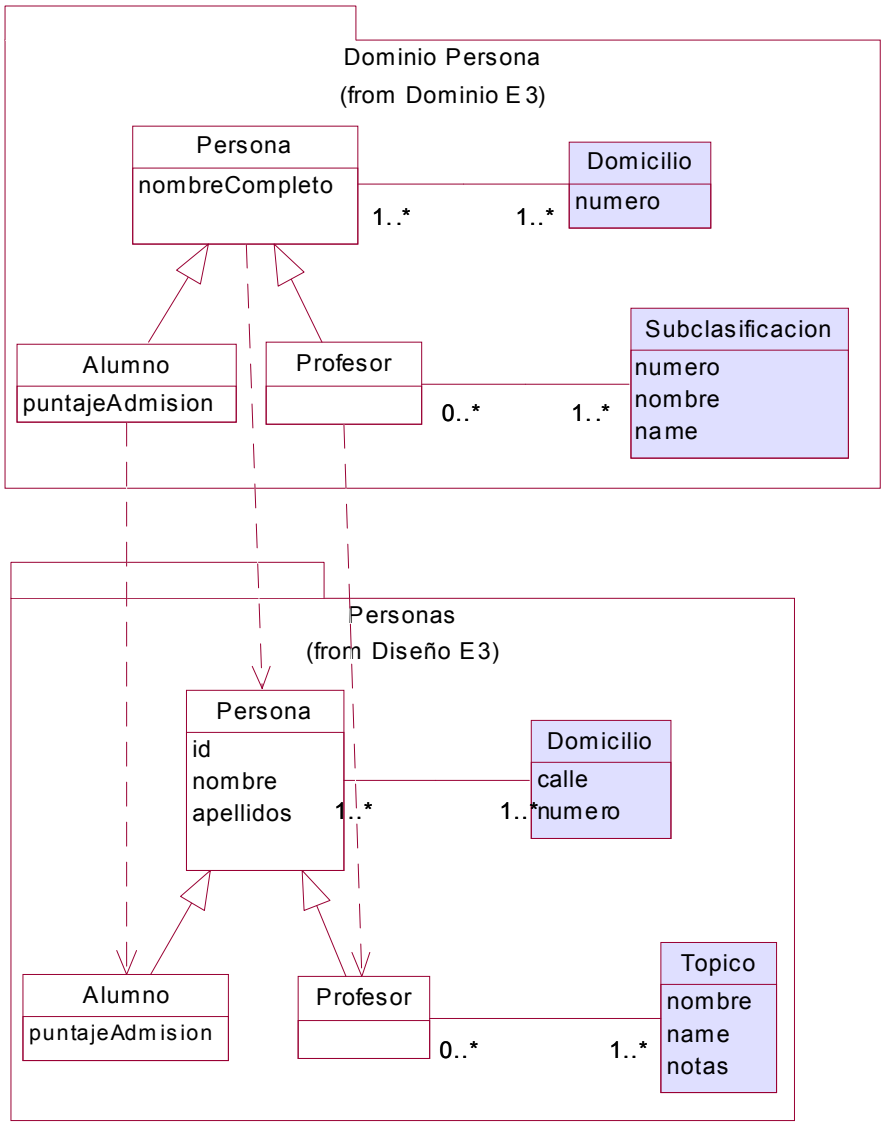


Figura 8.11. Refinamientos entre clases de los paquetes "Dominio Persona" y "Personas" del caso escolar. Elaboración 3

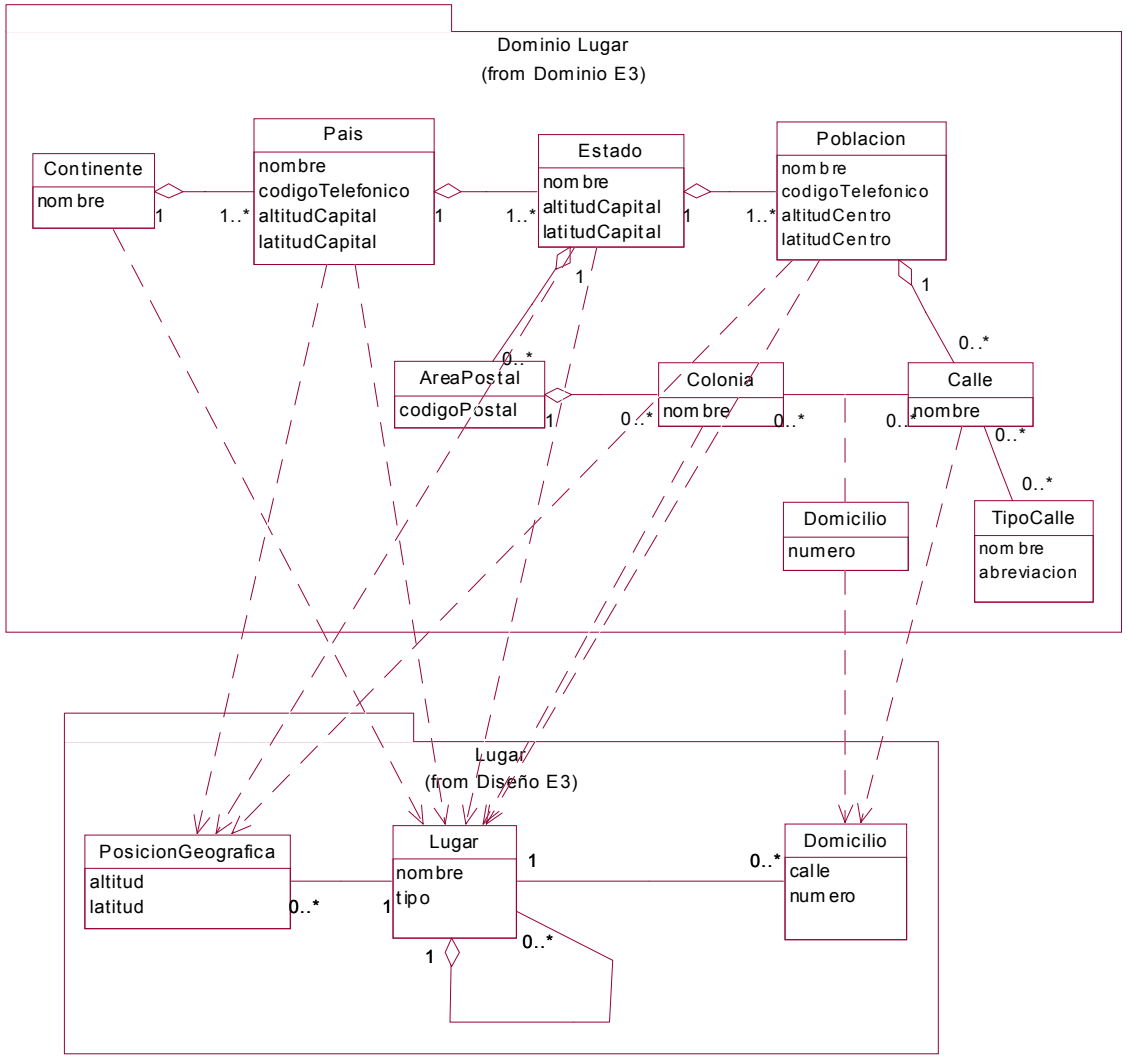


Figura 8.12. Refinamientos entre clases de los paquetes “Dominio Lugar” y “Lugar” del caso escolar. Elaboración 3

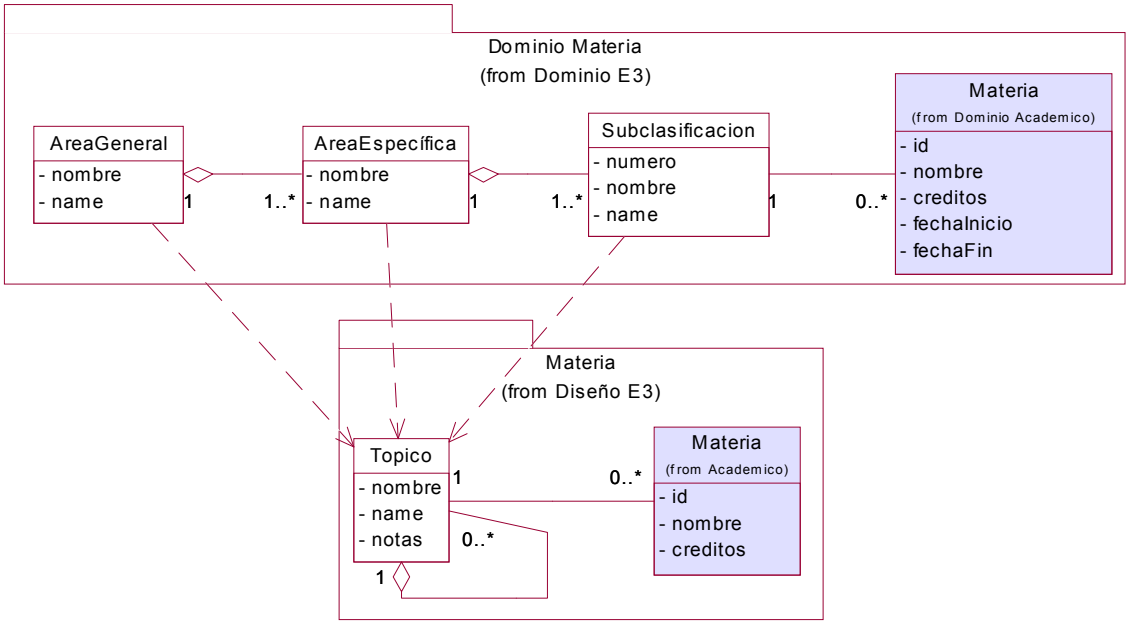


Figura 8.13. Refinamientos entre clases de los paquetes "Dominio Materia" y "Materia" del caso escolar. Elaboración 3

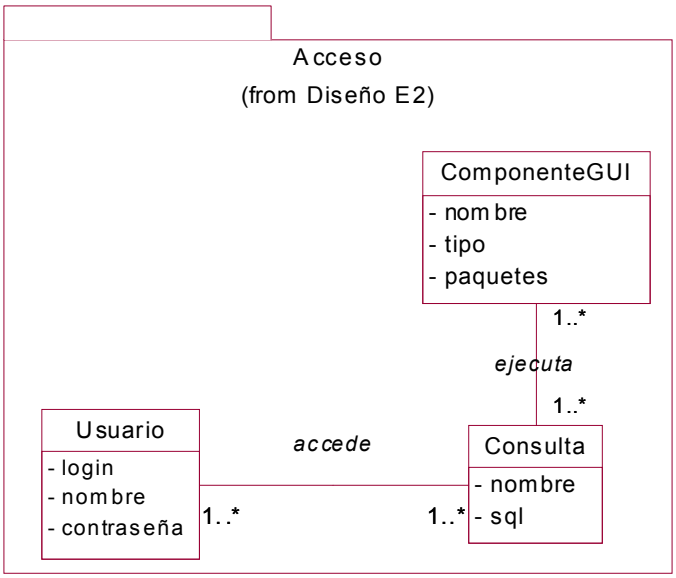


Figura 8.14. Paquete de diseño: Acceso. Elaboración 3 (sin cambios de E2)

Tabla 8.7. Matriz de correspondencia de paquetes. Elaboración 3

Paquetes	Diseño E3::Academico	Diseño E3::Personas	Diseño E3::Materia	Diseño E3::Lugar	Diseño E3::Acceso
Dominio E3::Dominio Academico	1				
Dominio E3::Dominio Persona		1			
Dominio E3::Dominio Materia			1		
Dominio E3::Dominio Lugar				1	

Tabla 8.8. Matriz de correspondencia de clases. Elaboración 3

Paquetes	Clase	Diseño E3::Academico	Diseño E3::Academico	Diseño E3::Academico	Diseño E3::Academico	Diseño E3::Personas	Diseño E3::Personas	Diseño E3::Personas	Diseño E3::Lugar	Diseño E3::Lugar	Diseño E3::Lugar	Diseño E3::Materia	Diseño E3::Acceso	Diseño E3::Acceso	Diseño E3::Acceso
		Calificacion	ProgramaAcademico	Materia	Periodo	Alumno	Persona	Profesor	Domicilio	Lugar	PosicionGeografica	Topico	ComponenteGUI	Consulta	Usuario
Dominio E3::Dominio Academico	Calificacion	1													
Dominio E3::Dominio Academico	TipoCalificacion	1													
Dominio E3::Dominio Academico	ProgramaAcademico		1												
Dominio E3::Dominio Academico	Materia			1	1										
Dominio E3::Dominio Persona	Alumno					1									
Dominio E3::Dominio Persona	Persona						1								
Dominio E3::Dominio Persona	Profesor							1							
Dominio E3::Dominio Lugar	AreaPostal														
Dominio E3::Dominio Lugar	Calle								1						
Dominio E3::Dominio Lugar	Colonia									1					
Dominio E3::Dominio Lugar	Continente									1					
Dominio E3::Dominio Lugar	Domicilio								1						
Dominio E3::Dominio Lugar	Estado									1	1				
Dominio E3::Dominio Lugar	Pais									1	1				
Dominio E3::Dominio Lugar	Poblacion									1	1				
Dominio E3::Dominio Lugar	TipoCalle														
Dominio E3::Dominio Materia	AreaGeneral											1			
Dominio E3::Dominio Materia	AreaEspecifica											1			
Dominio E3::Dominio Materia	Subclasificacion											1			

8.6 Identificación de las Discrepancias entre los elementos seleccionados en varias iteraciones

8.6.1 Paquetes por iteración

Tabla 8.9. Identificación entre paquetes. Elaboración 1

	DiseñoE1	**Def**
DominioE1	1To1	-
Exc	-	

Tabla 8.10. Identificación entre paquetes. Elaboración 2

	Academico	Acceso	**Def**
Academico	O2	-	-
Persona	O2	-	-
Exc	-	E	

Tabla 8.11. Identificación entre paquetes. Elaboración 3

	Academico	Personas	Lugar	Materia	Acceso	**Def**
Academico	1To1	-	-	-	-	-
Persona	-	1To1	-	-	-	-
Lugar	-	-	1To1	-	-	-
Materia	-	-	-	1To1	-	-
Exc	-	-	-	-	E	

La situación de la Tabla 8.9 es trivial. La Sobrecarga (de segundo orden) de la Tabla 8.10 se debe a que el Paquete de Diseño "Acceso" es una librería. Esto también refleja una carencia de modularidad del Paquete de Diseño "Academico". Aunque la Sobrecarga se elimina en la Tabla 8.11 (i.e. tercera iteración de la fase de Elaboración), el Exceso se sigue manteniendo. Para la siguiente iteración, se sugiere separar el Paquete de Diseño "Acceso" de un paquete que agrupe a los demás Paquetes de Diseño. Esto con la finalidad de que la Brecha entre los paquetes (dentro del nuevo paquete) sea nula.

8.6.2 Clases por iteración

Tabla 8.12. Identificación entre clases. Elaboración 1

	Alumno	Calificacion	Profesor	Usuario	Consulta	**Def**
Alumno	1To1	-	-	-	-	-
Calificacion	-	1To1	-	-	-	-
Profesor	-	-	1To1	-	-	-
Exc	-	-	-	E	E	

Debido a la falta de agrupación de las Clases de Diseño "Usuario" y "Consulta", se tienen dos Excesos en la Tabla 8.12. Estas Discrepancias estaban "ocultadas" por el nivel de paquetes (ver Tabla 8.9). Es decir, la falta de un Paquete de Diseño para estas clases (e.g. "Acceso") redujo la Brecha entre paquetes pero aumentó la Brecha entre clases.

Tabla 8.13. Identificación entre clases. Elaboración 2

	Academico.Calificacion	Academico.ProgramaAcademico	Academico.Materia	Academico.Periodo	Academico.Alumno	Academico.Profesor	Acceso.ComponenteGUI	Acceso.Consulta	Acceso.Usuario	**Def**
Academico_Calificacion	O2	-	-	-	-	-	-	-	-	-
Academico_TipoCalificacion	O2	-	-	-	-	-	-	-	-	-
Academico_ProgramaAcademico	-	1To1	-	-	-	-	-	-	-	-
Academico_Materia	-	-	R2	R2	-	-	-	-	-	-
Persona_Alumno	-	-	-	-	O2	-	-	-	-	-
Persona_Persona	-	-	-	-	O2R2	O2R2	-	-	-	-
Persona_Profesor	-	-	-	-	-	O2	-	-	-	-
Exc	-	-	-	-	-	-	E	E	E	

La Tabla 8.13 es más interesante en el sentido de una diversidad de Discrepancias: tres Sobrecargas de orden 2, dos Redundancias de orden 2, tres Excesos y la Unión de dos Sobrecargas con dos Redundancias. Esta Unión se debió a la herencia de la Clase Conceptual "Persona". Por ello, aunque no se analice la correspondencia entre generalizaciones, se deben de tomar en cuenta. Los Excesos son causados por pertenecer a la librería "Acceso", por lo que pudieran ser excluidos de la métrica.

Tabla 8.14. Identificación entre clases. Elaboración 3

	Academico.Calificacion	Academico.ProgramaAcademico	Academico.Materia	Academico.Periodo	Personas.Alumno	Personas.Persona	Personas.Profesor	Lugar.Domicilio	Lugar.Lugar	Lugar.PosicionGeografica	Materia.Topico	Acceso.ComponenteGUI	Acceso.Consulta	Acceso.Usuario	**Def**
Academico_Calificacion	O2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Academico_TipoCalificacion	O2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Academico_ProgramaAcademico	-	1To1	-	-	-	-	-	-	-	-	-	-	-	-	-
Academico_Materia	-	-	R2	R2	-	-	-	-	-	-	-	-	-	-	-
Persona_Alumno	-	-	-	-	1To1	-	-	-	-	-	-	-	-	-	-
Persona_Persona	-	-	-	-	-	1To1	-	-	-	-	-	-	-	-	-
Persona_Profesor	-	-	-	-	-	-	1To1	-	-	-	-	-	-	-	-
Lugar_AreaPostal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	D
Lugar_Calle	-	-	-	-	-	-	-	O2	-	-	-	-	-	-	-
Lugar_Colonia	-	-	-	-	-	-	-	-	O5	-	-	-	-	-	-
Lugar_Continente	-	-	-	-	-	-	-	-	O5	-	-	-	-	-	-
Lugar_Domicilio	-	-	-	-	-	-	-	O2	-	-	-	-	-	-	-
Lugar_Estado	-	-	-	-	-	-	-	-	O5R2	O3R2	-	-	-	-	-
Lugar_Pais	-	-	-	-	-	-	-	-	O5R2	O3R2	-	-	-	-	-
Lugar_Poblacion	-	-	-	-	-	-	-	-	O5R2	O3R2	-	-	-	-	-
Lugar_TipoCalle	-	-	-	-	-	-	-	-	-	-	-	-	-	-	D
Materia_AreaGeneral	-	-	-	-	-	-	-	-	-	-	O3	-	-	-	-
Materia_AreaEspecifica	-	-	-	-	-	-	-	-	-	-	O3	-	-	-	-
Materia_Subclasificacion	-	-	-	-	-	-	-	-	-	-	O3	-	-	-	-
Exc	-	-	-	-	-	-	-	-	-	-	-	E	E	E	

En la Tabla 8.14, se aprecia que las Discrepancias de los paquetes de “Persona” fueron eliminadas. Mientras que en la Figura 8.12, a simple vista se aprecian muchas Discrepancias (i.e. Brecha amplia) entre los paquetes de “Lugar”. En especial se aprecia una Unión y los Déficit de “AreaPostal” y “TipoCalle”. Al enfocarse en los paquetes “Lugar” y “Materia”, se puede apreciar (por las Sobrecargas) que el diseñador “usualmente” representa el trabajo del analista con menos elementos.

8.6.3 Atributos por iteración

Tabla 8.15. Identificación entre atributos. Elaboración 1. Parte 1 de 2

Parte 1 | Parte 2

	Alumno.OID	Alumno.matricula	Alumno.nombre	Calificacion.OID	Calificacion.calificacion	Calificacion.materia	Profesor.OID	Profesor.nombre
Alumno_OID	1To1	-	-	-	-	-	-	-
Alumno_matricula	-	1To1	-	-	-	-	-	-
Alumno_nombre	-	-	1To1	-	-	-	-	-
Calificacion_OID	-	-	-	1To1	-	-	-	-
Calificacion_calificacion	-	-	-	-	1To1	-	-	-
Calificacion_materia	-	-	-	-	-	1To1	-	-
Profesor_OID	-	-	-	-	-	-	1To1	-
Profesor_nombre	-	-	-	-	-	-	-	1To1
Exc	-	-	-	-	-	-	-	-

Tabla 8.16. Identificación entre atributos. Elaboración 1. Parte 2 de 2

Parte 1 | **Parte 2**

	Usuario.OID	Usuario.login	Usuario.nombre	Usuario.contraseña	Consulta.OID	Consulta.nombre	Consulta.sql	**Def**
Alumno_OID	-	-	-	-	-	-	-	-
Alumno_matricula	-	-	-	-	-	-	-	-
Alumno_nombre	-	-	-	-	-	-	-	-
Calificacion_OID	-	-	-	-	-	-	-	-
Calificacion_calificacion	-	-	-	-	-	-	-	-
Calificacion_materia	-	-	-	-	-	-	-	-
Profesor_OID	-	-	-	-	-	-	-	-
Profesor_nombre	-	-	-	-	-	-	-	-
Exc	E	E	E	E	E	E	E	

Si bien en la Tabla 8.15, se muestran sólo relaciones uno a uno (i.e. Brecha nula), los Excesos de la Tabla 8.16, por incluir elementos de librería se hace palpable.

Tabla 8.17. Identificación entre atributos. Elaboración 2. Parte 1 de 3

Parte 1 | Parte 2 | Parte 3

	Calificacion.OID	Calificacion.calificacion	Calificacion.tipo	ProgramaAcademico.OID	ProgramaAcademico.nombre	ProgramaAcademico.capacidad	Materia.OID	Materia.nombre	Materia.creditos	Periodo.OID	Periodo.fechaInicio	Periodo.fechaFin
Calificacion_OID	O2	-	-	-	-	-	-	-	-	-	-	-
Calificacion_calificacion	-	1To1	-	-	-	-	-	-	-	-	-	-
TipoCalificacion_OID	O2	-	-	-	-	-	-	-	-	-	-	-
TipoCalificacion_nombre	-	-	1To1	-	-	-	-	-	-	-	-	-
ProgramaAcademico_OID	-	-	-	1To1	-	-	-	-	-	-	-	-
ProgramaAcademico_nombre	-	-	-	-	1To1	-	-	-	-	-	-	-
ProgramaAcademico_capacidad	-	-	-	-	-	1To1	-	-	-	-	-	-
Materia_OID	-	-	-	-	-	-	R2	-	-	R2	-	-
Materia_nombre	-	-	-	-	-	-	-	1To1	-	-	-	-
Materia_creditos	-	-	-	-	-	-	-	-	1To1	-	-	-
Materia_fechaInicio	-	-	-	-	-	-	-	-	-	-	1To1	-
Materia_fechaFin	-	-	-	-	-	-	-	-	-	-	-	1To1
Alumno_OID	-	-	-	-	-	-	-	-	-	-	-	-
Alumno_puntajeAdmision	-	-	-	-	-	-	-	-	-	-	-	-
Persona_OID	-	-	-	-	-	-	-	-	-	-	-	-
Persona_nombreCompleto	-	-	-	-	-	-	-	-	-	-	-	-
Profesor_OID	-	-	-	-	-	-	-	-	-	-	-	-
Exc	-	-	-	-	-	-	-	-	-	-	-	-

La Tabla 8.17 muestra una Brecha estrecha, salvo la misma Sobrecarga de las clases de “Calificacion” y “TipoCalificacion”, y la Redundancia entre las clases de “Materia” y “Periodo” sólo que a detalle de los atributos.

Tabla 8.18. Identificación entre atributos. Elaboración 2. Parte 2 de 3

Parte 1 **Parte 2** Parte 3

	Alumno.OID	Alumno.nombre	Alumno.apellidos	Alumno.puntajeAdmision	Profesor.OID	Profesor.nombre	Profesor.apellidos
Calificacion_OID	-	-	-	-	-	-	-
Calificacion_calificacion	-	-	-	-	-	-	-
TipoCalificacion_OID	-	-	-	-	-	-	-
TipoCalificacion_nombre	-	-	-	-	-	-	-
ProgramaAcademico_OID	-	-	-	-	-	-	-
ProgramaAcademico_nombre	-	-	-	-	-	-	-
ProgramaAcademico_capacidad	-	-	-	-	-	-	-
Materia_OID	-	-	-	-	-	-	-
Materia_nombre	-	-	-	-	-	-	-
Materia_creditos	-	-	-	-	-	-	-
Materia_fechaInicio	-	-	-	-	-	-	-
Materia_fechaFin	-	-	-	-	-	-	-
Alumno_OID	O2	-	-	-	-	-	-
Alumno_puntajeAdmision	-	-	-	1To1	-	-	-
Persona_OID	O2R2	-	-	-	O2R2	-	-
Persona_nombreCompleto	-	R4	R4	-	-	R4	R4
Profesor_OID	-	-	-	-	O2	-	-
Exc	-	-	-	-	-	-	-

En la Tabla 8.18, se tienen más Discrepancias. Con el atributo por omisión ObjectId (OID) se resalta la carencia de la herencia de la Clase Conceptual Persona. Ésta, provoca dos Sobrecargas y una Redundancia (con las Clases de Diseño “Alumno” y “Profesor”), así como una Redundancia de cuarto orden entre los atributos de “nombre”, “apellidos” y “nombreCompleto”.

Tabla 8.19. Identificación entre atributos. Elaboración 2. Parte 3 de 3

Parte 1	Parte 2	**Parte 3**

	ComponenteGUI.OID	ComponenteGUI.nombre	ComponenteGUI.tipo	ComponenteGUI.paquetes	Consulta.OID	Consulta.nombre	Consulta.sql	Usuario.OID	Usuario.login	Usuario.nombre	Usuario.contraseña	**Def**
Calificacion_OID	-	-	-	-	-	-	-	-	-	-	-	-
Calificacion_calificacion	-	-	-	-	-	-	-	-	-	-	-	-
TipoCalificacion_OID	-	-	-	-	-	-	-	-	-	-	-	-
TipoCalificacion_nombre	-	-	-	-	-	-	-	-	-	-	-	-
ProgramaAcademico_OID	-	-	-	-	-	-	-	-	-	-	-	-
ProgramaAcademico_nombre	-	-	-	-	-	-	-	-	-	-	-	-
ProgramaAcademico_capacidad	-	-	-	-	-	-	-	-	-	-	-	-
Materia_OID	-	-	-	-	-	-	-	-	-	-	-	-
Materia_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Materia_creditos	-	-	-	-	-	-	-	-	-	-	-	-
Materia_fechaInicio	-	-	-	-	-	-	-	-	-	-	-	-
Materia_fechaFin	-	-	-	-	-	-	-	-	-	-	-	-
Alumno_OID	-	-	-	-	-	-	-	-	-	-	-	-
Alumno_puntajeAdmision	-	-	-	-	-	-	-	-	-	-	-	-
Persona_OID	-	-	-	-	-	-	-	-	-	-	-	-
Persona_nombreCompleto	-	-	-	-	-	-	-	-	-	-	-	-
Profesor_OID	-	-	-	-	-	-	-	-	-	-	-	-
Exc	E	E	E	E	E	E	E	E	E	E	E	

En la Tabla 8.19, simplemente se vuelve a ver el efecto de la inclusión de la librería visto en la Tabla 8.11 de paquetes y en la Tabla 8.14 de clases. Cabe resaltar que la Tabla 8.17, Tabla 8.18 y Tabla 8.19 provienen de la misma.

Tabla 8.20. Identificación entre atributos. Elaboración 3. Parte 1 de 4

Parte 1	Parte 2
Parte 3	Parte 4

	Calificacion.OID	Calificacion.calificacion	Calificacion.tipo	ProgramaAcademico.OID	ProgramaAcademico.nombre	ProgramaAcademico.capacidad	Materia.OID	Materia.nombre	Materia.creditos
Calificacion_OID	O2	-	-	-	-	-	-	-	-
Calificacion_calificacion	-	1To1	-	-	-	-	-	-	-
TipoCalificacion_OID	O2	-	-	-	-	-	-	-	-
TipoCalificacion_nombre	-	-	1To1	-	-	-	-	-	-
ProgramaAcademico_OID	-	-	-	1To1	-	-	-	-	-
ProgramaAcademico_nombre	-	-	-	-	1To1	-	-	-	-
ProgramaAcademico_capacidad	-	-	-	-	-	1To1	-	-	-
Materia_OID	-	-	-	-	-	-	R2	-	-
Materia_nombre	-	-	-	-	-	-	-	1To1	-
Materia_creditos	-	-	-	-	-	-	-	-	1To1
Materia_fechaInicio	-	-	-	-	-	-	-	-	-
Materia_fechaFin	-	-	-	-	-	-	-	-	-
Alumno_OID	-	-	-	-	-	-	-	-	-
Alumno_puntajeAdmision	-	-	-	-	-	-	-	-	-
Persona_OID	-	-	-	-	-	-	-	-	-
Persona_nombreCompleto	-	-	-	-	-	-	-	-	-
Profesor_OID	-	-	-	-	-	-	-	-	-
...									
Exc	-	-	-	-	-	-	-	-	-

Los atributos de la Tabla 8.20, se mantuvieron iguales que los de la Tabla 8.17, debido a que son Discrepancias justificadas. Por ejemplo, aunque la Clase Conceptual "TipoCalificacion" es una entidad aparte de "Calificacion", por decisiones de diseño, se fusionó con la Clase de Diseño "Calificacion" como un atributo.

Tabla 8.21. Identificación entre atributos. Elaboración 3. Parte 2 de 4

Parte 1	**Parte 2**
Parte 3	Parte 4

	Periodo.OID	Periodo.fechaInicio	Periodo.fechaFin	Alumno.OID	Alumno.puntajeAdmision	Persona.OID	Persona.nombre	Persona.apellidos	Profesor.OID
Calificacion_OID	-	-	-	-	-	-	-	-	-
Calificacion_calificacion	-	-	-	-	-	-	-	-	-
TipoCalificacion_OID	-	-	-	-	-	-	-	-	-
TipoCalificacion_nombre	-	-	-	-	-	-	-	-	-
ProgramaAcademico_OID	-	-	-	-	-	-	-	-	-
ProgramaAcademico_nombre	-	-	-	-	-	-	-	-	-
ProgramaAcademico_capacidad	-	-	-	-	-	-	-	-	-
Materia_OID	R2	-	-	-	-	-	-	-	-
Materia_nombre	-	-	-	-	-	-	-	-	-
Materia_creditos	-	-	-	-	-	-	-	-	-
Materia_fechaInicio	-	1To1	-	-	-	-	-	-	-
Materia_fechaFin	-	-	1To1	-	-	-	-	-	-
Alumno_OID	-	-	-	1To1	-	-	-	-	-
Alumno_puntajeAdmision	-	-	-	-	1To1	-	-	-	-
Persona_OID	-	-	-	-	-	1To1	-	-	-
Persona_nombreCompleto	-	-	-	-	-	-	R2	R2	-
Profesor_OID	-	-	-	-	-	-	-	-	1To1
...									
Exc	-	-	-	-	-	-	-	-	-

Al crearse la Clase de Diseño Persona (y su herencia) en la Tabla 8.21, se eliminan las Discrepancias del paquete “Persona” (ver Tabla 8.18). Sólo se conservó, la Redundancia entre el atributo “nombre”, “apellidos” y “nombreCompleto”, como una decisión de diseño. Otra opción de Atributos de Diseño sería tener “nombre”, “apellidoPaterno” y “apellidoMaterno” (como campo “nulo”).

Tabla 8.22. Identificación entre atributos. Elaboración 3. Parte 3 de 4

Parte 1	Parte 2
Parte 3	Parte 4

	Domicilio.OID	Domicilio.calle	Domicilio.numero	Lugar.OID	Lugar.nombre	Lugar.tipo	PosicionGeografica.OID	PosicionGeografica.altitud	PosicionGeografica.latitud	Topico.OID	Topico.nombre	Topico.name	Topico.notas
...													
AreaPostal_OID	-	-	-	-	-	-	-	-	-	-	-	-	-
AreaPostal_codigoPostal	-	-	-	-	-	-	-	-	-	-	-	-	-
Calle_OID	O3	-	-	-	-	-	-	-	-	-	-	-	-
Calle_nombre	-	O2	-	-	-	-	-	-	-	-	-	-	-
Colonia_OID	-	-	-	O5	-	-	-	-	-	-	-	-	-
Colonia_nombre	-	-	-	-	O5R2	O5R2	-	-	-	-	-	-	-
Continente_OID	-	-	-	O5	-	-	-	-	-	-	-	-	-
Continente_nombre	-	-	-	-	O5R2	O5R2	-	-	-	-	-	-	-
Domicilio_OID	O3	-	-	-	-	-	-	-	-	-	-	-	-
Domicilio_numero	-	-	1To1	-	-	-	-	-	-	-	-	-	-
Estado_OID	-	-	-	O5R2	-	-	O3R2	-	-	-	-	-	-
Estado_nombre	-	-	-	-	O5R2	O5R2	-	-	-	-	-	-	-
Estado_altitudCapital	-	-	-	-	-	-	-	O3	-	-	-	-	-
Estado_latitudCapital	-	-	-	-	-	-	-	-	O3	-	-	-	-
Pais_OID	-	-	-	O5R2	-	-	O3R2	-	-	-	-	-	-
Pais_nombre	-	-	-	-	O5R2	O5R2	-	-	-	-	-	-	-
Pais_codigoTelefonico	-	-	-	-	-	-	-	-	-	-	-	-	-
Pais_altitudCapital	-	-	-	-	-	-	-	O3	-	-	-	-	-
Pais_latitudCapital	-	-	-	-	-	-	-	-	O3	-	-	-	-
Poblacion_OID	-	-	-	O5R2	-	-	O3R2	-	-	-	-	-	-
Poblacion_nombre	-	-	-	-	O5R2	O5R2	-	-	-	-	-	-	-
Poblacion_codigoTelefonico	-	-	-	-	-	-	-	-	-	-	-	-	-
Poblacion_altitudCentro	-	-	-	-	-	-	-	O3	-	-	-	-	-
Poblacion_latitudCentro	-	-	-	-	-	-	-	-	O3	-	-	-	-
TipoCalle_OID	O3	-	-	-	-	-	-	-	-	-	-	-	-
TipoCalle_nombre	-	O2	-	-	-	-	-	-	-	-	-	-	-
AreaGeneral_OID	-	-	-	-	-	-	-	-	-	O3	-	-	-
AreaGeneral_nombre	-	-	-	-	-	-	-	-	-	-	O3	-	-
AreaGeneral_name	-	-	-	-	-	-	-	-	-	-	-	O3	-
AreaEspecifica_OID	-	-	-	-	-	-	-	-	-	O3	-	-	-
AreaEspecifica_nombre	-	-	-	-	-	-	-	-	-	-	O3	-	-
AreaEspecifica_name	-	-	-	-	-	-	-	-	-	-	-	O3	-
Subclasificacion_OID	-	-	-	-	-	-	-	-	-	O3	-	-	-
Subclasificacion_numero	-	-	-	-	-	-	-	-	-	-	-	-	1To1
Subclasificacion_nombre	-	-	-	-	-	-	-	-	-	-	O3	-	-
Subclasificacion_name	-	-	-	-	-	-	-	-	-	-	-	O3	-
Exc	-	-	-	-	-	-	-	-	-	-	-	-	-

En la iteración Elaboración 3, se agregaron los atributos de la Tabla 8.22. La presencia de diversas Discrepancias en estos paquetes posiblemente, se deba a que el analista y el diseñador trabajaron de forma separada (ver "Elaboración 3" en la página 163). Con este análisis, el verificador tiene evidencia de que se están tomando rumbos distintos.

Tabla 8.23. Identificación entre atributos. Elaboración 3. Parte 4 de 4

Parte 1	Parte 2
Parte 3	**Parte 4**

	ComponenteGUI.OID	ComponenteGUI.nombre	ComponenteGUI.tipo	ComponenteGUI.paquetes	Consulta.OID	Consulta.nombre	Consulta.sql	Usuario.OID	Usuario.login	Usuario.nombre	Usuario.contraseña	**Def**
...												
AreaPostal_OID	-	-	-	-	-	-	-	-	-	-	-	D
AreaPostal_codigoPostal	-	-	-	-	-	-	-	-	-	-	-	D
Calle_OID	-	-	-	-	-	-	-	-	-	-	-	-
Calle_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Colonia_OID	-	-	-	-	-	-	-	-	-	-	-	-
Colonia_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Continente_OID	-	-	-	-	-	-	-	-	-	-	-	-
Continente_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Domicilio_OID	-	-	-	-	-	-	-	-	-	-	-	-
Domicilio_numero	-	-	-	-	-	-	-	-	-	-	-	-
Estado_OID	-	-	-	-	-	-	-	-	-	-	-	-
Estado_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Estado_altitudCapital	-	-	-	-	-	-	-	-	-	-	-	-
Estado_latitudCapital	-	-	-	-	-	-	-	-	-	-	-	-
Pais_OID	-	-	-	-	-	-	-	-	-	-	-	-
Pais_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Pais_codigoTelefonico	-	-	-	-	-	-	-	-	-	-	-	D
Pais_altitudCapital	-	-	-	-	-	-	-	-	-	-	-	-
Pais_latitudCapital	-	-	-	-	-	-	-	-	-	-	-	-
Poblacion_OID	-	-	-	-	-	-	-	-	-	-	-	-
Poblacion_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Poblacion_codigoTelefonico	-	-	-	-	-	-	-	-	-	-	-	D
Poblacion_altitudCentro	-	-	-	-	-	-	-	-	-	-	-	-
Poblacion_latitudCentro	-	-	-	-	-	-	-	-	-	-	-	-
TipoCalle_OID	-	-	-	-	-	-	-	-	-	-	-	-
TipoCalle_nombre	-	-	-	-	-	-	-	-	-	-	-	-
AreaGeneral_OID	-	-	-	-	-	-	-	-	-	-	-	-
AreaGeneral_nombre	-	-	-	-	-	-	-	-	-	-	-	-
AreaGeneral_name	-	-	-	-	-	-	-	-	-	-	-	-
AreaEspecifica_OID	-	-	-	-	-	-	-	-	-	-	-	-
AreaEspecifica_nombre	-	-	-	-	-	-	-	-	-	-	-	-
AreaEspecifica_name	-	-	-	-	-	-	-	-	-	-	-	-
Subclasificacion_OID	-	-	-	-	-	-	-	-	-	-	-	-
Subclasificacion_numero	-	-	-	-	-	-	-	-	-	-	-	-
Subclasificacion_nombre	-	-	-	-	-	-	-	-	-	-	-	-
Subclasificacion_name	-	-	-	-	-	-	-	-	-	-	-	-
Exc	E	E	E	E	E	E	E	E	E	E	E	

En la Tabla 8.23, se aprecia la misma situación sobre el paquete de librería. Los Déficit que se señalan son parte de la separación del analista y el diseñador, explicación similar a la de la Tabla 8.22.

8.7 Evolución de las métricas de la Brecha del Modelo

8.7.1 Por iteración

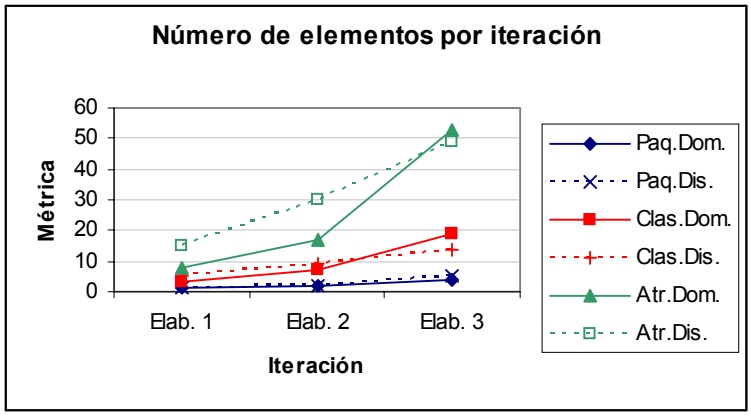


Figura 8.15. Número de elementos por iteración. Las líneas continuas se refieren a los elementos del Dominio, mientras que las punteadas, a los elementos del Diseño

En la Figura 8.15, se muestra el crecimiento a través del tiempo de todos los elementos seleccionados (paquetes, clases y atributos). Esto probablemente implica que el crecimiento del sistema aun es inestable. Cuando el número de elementos es distinto entre el Modelo del Dominio y el Modelo del Diseño, hay una Brecha no nula. Si la cantidad de elementos del Diseño es mayor que del Dominio, posiblemente se tengan más Excesos y Redundancias (i.e. Discrepancias Abundante en lo Particular o en el Diseño) como sucedió en la mayoría de los casos, sobretodo en las iteraciones de Elaboración 1 y Elaboración 2 para atributos.

Promediando el crecimiento de los elementos de la Elaboración 3 con respecto a la Elaboración 1, se tiene que los paquetes aumentaron a un 450% (400% para los elementos del Dominio, 500% para los elementos del Diseño), las clases 457% (633%, 280%) y los atributos 495% (663%, 327%). Por lo que se puede decir que el crecimiento de los elementos seleccionados fue al menos del 450% (i.e. al menos hay cuatro elementos en la Elaboración 3 por cada elemento de la Elaboración 1).

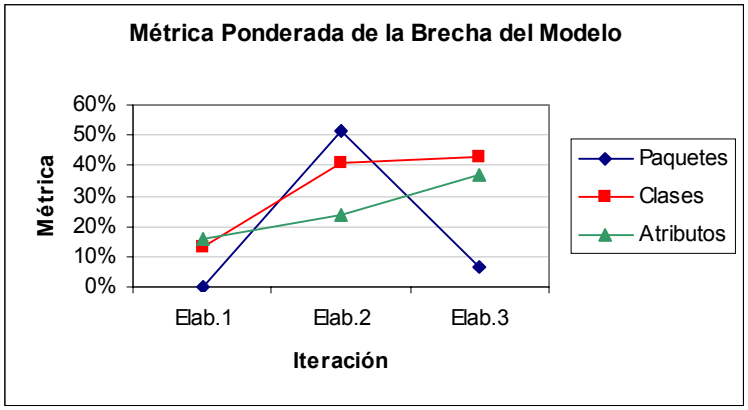


Figura 8.16. Métrica ponderada del Modelo por iteración

En la Figura 8.16, se muestra como la métrica pesada cambia en las iteraciones. La métrica pesada entre los paquetes es la que se reduce drásticamente en la Elaboración 3. Mientras que las demás tienden a crecer. Sin embargo, para la última iteración se aprecia un control en los paquetes y las clases. Aunque el orden de las métricas de la razón por iteración es distinto, el tamaño del rango de la métrica es a lo más de 36%. Al parecer el tamaño del rango crece conforme al tiempo: 16% para la primera iteración, 28% para la segunda y 36% para la tercera.

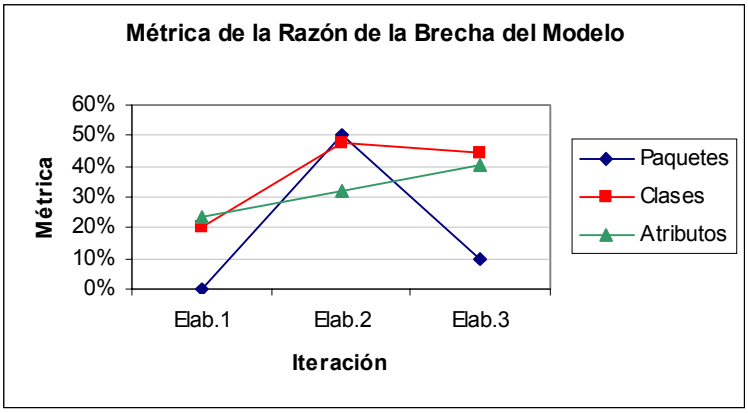


Figura 8.17. Métricas de la razón del Modelo por iteración

En la Figura 8.17, se resalta que hay una reducción en las métricas de la razón entre las clases. El contraste de esta curva con la de la Figura 8.16, es que se eliminaron más Discrepancias de poco peso. Las Discrepancias de la primera iteración son de peso bajo pues en la Figura 8.16, la métrica pesada es menor que la de la razón.

En la Figura 8.16 y Figura 8.17, se aprecia que las métricas pesadas y de la razón entre clases y atributos, son más similares que entre la de los paquetes y los demás elementos. Para asegurar que estas métricas entre clases y atributos son muy similares se necesitaría de más datos.

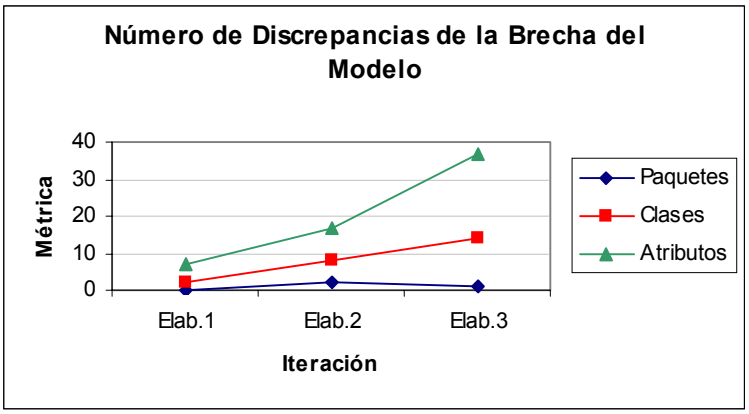


Figura 8.18. Número de Discrepancias del Modelo por iteración

Si bien en la Figura 8.18, se aprecia que conforme pasa el tiempo, hay mayor número de Discrepancias, es debido al crecimiento del número de elementos de la Figura 8.15. La métrica ponderada (Figura 8.16) y la métrica de la razón (Figura 8.17) normaliza estas cantidades.

8.7.2 Por tipo de Discrepancia

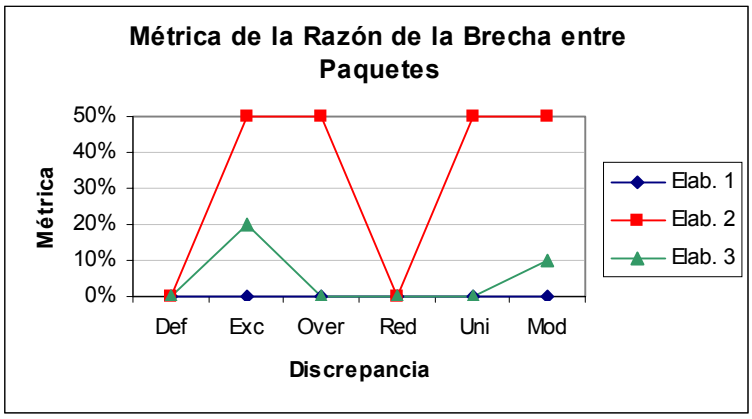


Figura 8.19. Métrica de la razón de la Brecha entre paquetes

Similar a la métrica entre paquetes de la Figura 8.16, en la Figura 8.19, se inicia con una Brecha estrecha para luego ampliarse en la segunda iteración y ser corregida en la tercera. En esta gráfica, simplemente se muestra mayor detalle con los tipos de Discrepancias y para cada iteración.

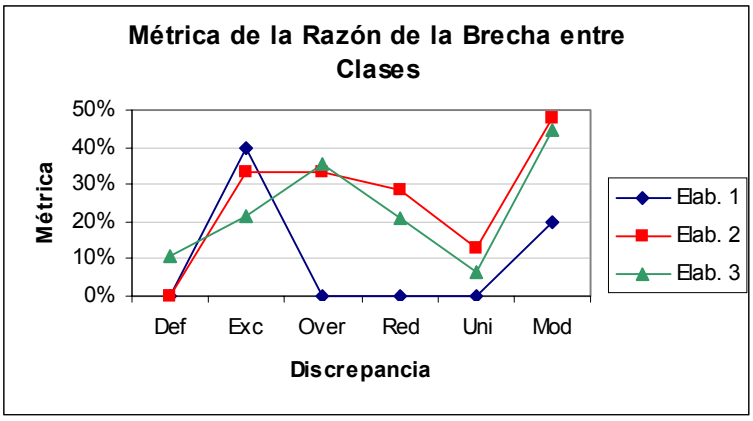


Figura 8.20. Métrica de la razón de la Brecha entre clases

En general, en la Figura 8.20, se presenta una situación similar a la Brecha entre paquetes posiblemente debida a la propagación hacia las clases. Excepto por los Excesos y los Déficit, se tiene un comportamiento similar: se inicia con una Brecha estrecha, al crecer el número de clases, se incrementan las Discrepancias ampliando la Brecha para en la tercera iteración reducir la Brecha. Por ejemplo, en la Elaboración 1, no hay Redundancias, luego se alcanza un 29% de ellas para reducirlas a un 21%. Cabe notar que las métricas entre clases y atributos son más similares entre ellas, que con la métrica entre los paquetes.

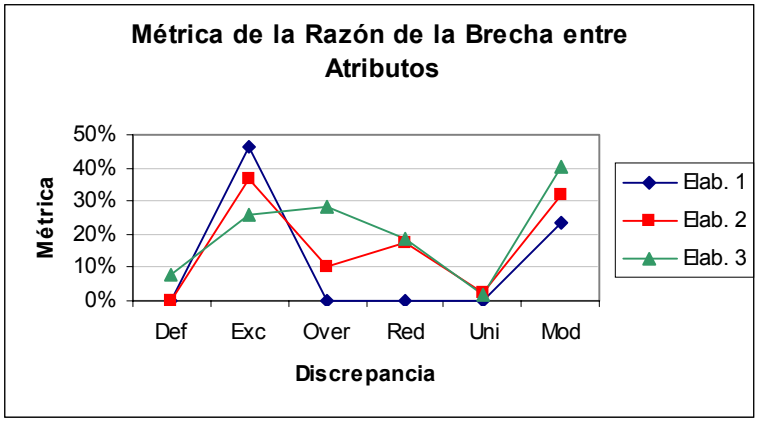


Figura 8.21. Métrica de la razón de la Brecha entre atributos

En la Figura 8.21, se muestra que la mayor parte de las Discrepancias son los Excesos en la primera y segunda iteración. Mientras que en la tercera hay una mayor variedad de Discrepancias.

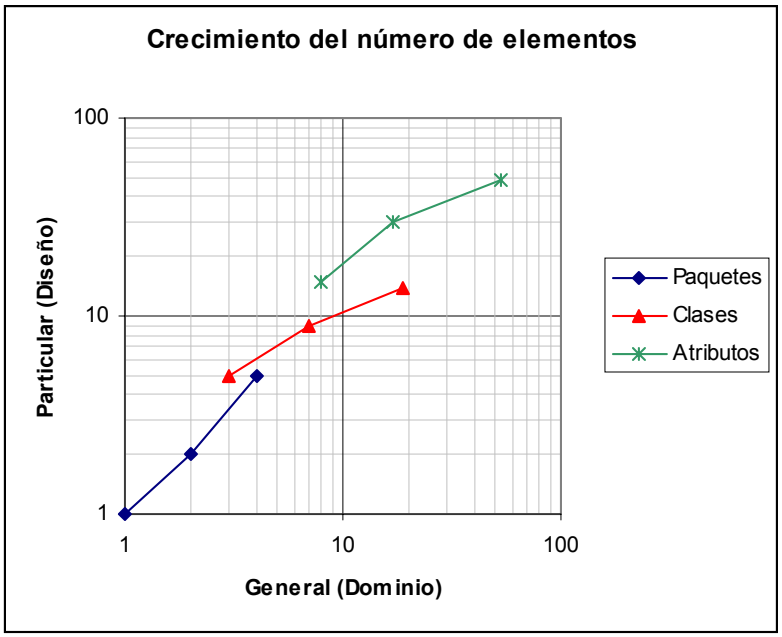


Figura 8.22. Crecimiento del número de elementos del Modelo General contra el del Modelo Particular

Cada curva de la Figura 8.22, inicia en la Elaboración. 1 y termina en la Elaboración 3. Se empleó la escala logarítmica para los ejes para comparar las cantidades por elemento. Se puede apreciar que hay más atributos que clases y que hay más clases que paquetes. Si se tiene un número igual de elementos generales y particulares (e.g. Clases Conceptuales y Clases de Diseño), el punto se localiza en la línea a 45 grados.

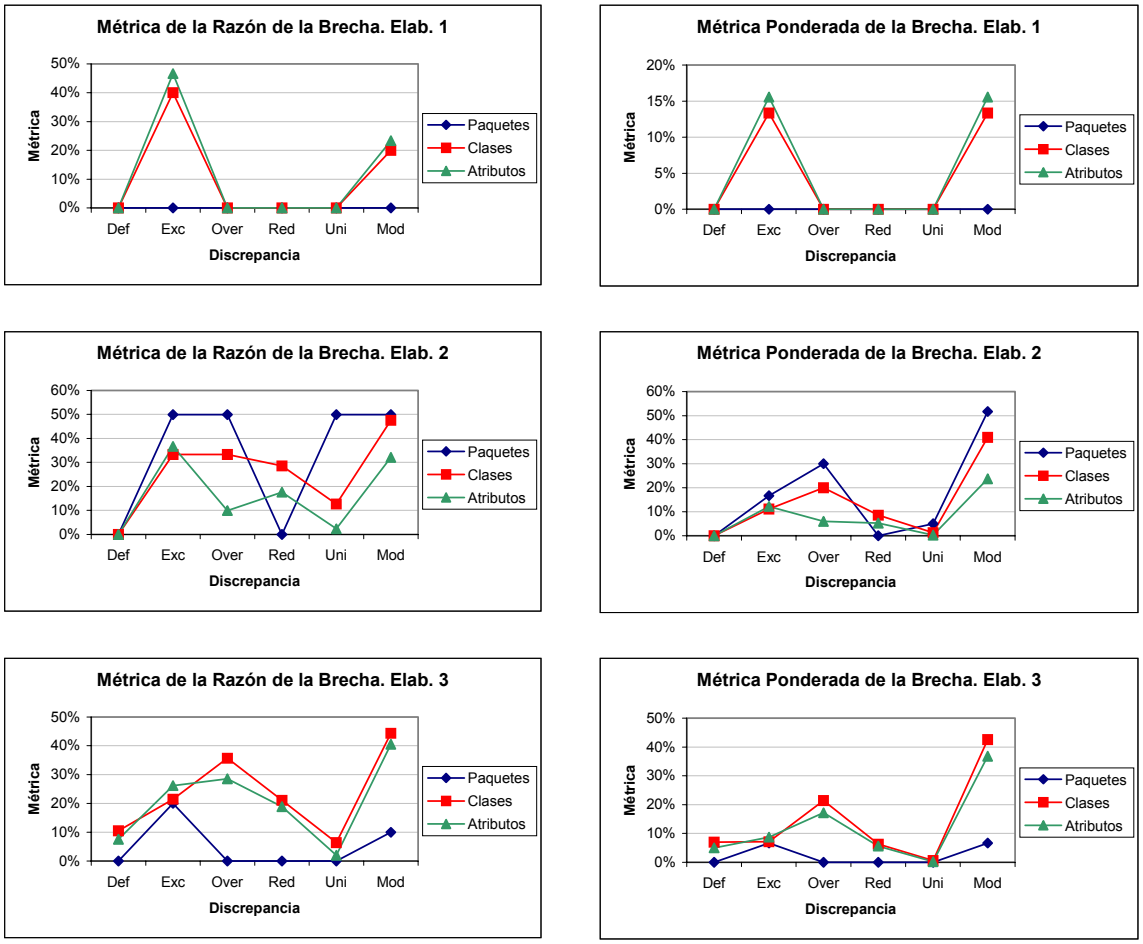


Figura 8.23. Las métricas de la razón y las ponderadas entre clases y atributos parecen seguir un comportamiento más similar que entre paquetes

En el conjunto de gráficas de la Figura 8.23, se puede apreciar que, en general, las métricas de la razón por tipo de Discrepancias entre las clases y los atributos son similares. Al menos, son más similares que comparándolas con las métricas entre paquetes. Por esta razón, parece que las métricas entre paquetes y entre clases son las que contribuyen más a describir las Discrepancias. Es decir, si no se calculan las métricas entre atributos, no hay tanta pérdida de información sobre las Discrepancias como con los paquetes y las clases.

Parte 4. Culminación

Capítulo 9. Conclusiones

9.1 Definición de la Brecha Representacional

Una vez que se ha estudiado y caracterizado la Brecha Representacional en algunos casos de estudio, se está en condiciones de proponer una definición más formal y acotada que la proporcionada por [Larman01]:

"La Brecha Representacional es una distancia entre un modelo general y un modelo particular de las etapas del desarrollo de sistemas que considera un conjunto de discrepancias de refinamiento"

Para detallar la definición anterior se menciona que:

- El "conjunto de discrepancias de refinamiento" se puede modelar mediante un digrafo bipartito, donde los vértices representan a los elementos de modelado y los arcos, los refinamientos del conjunto de elementos del modelo general (i.e. elementos por refinar), al conjunto de elementos del modelo particular (i.e. elementos refinados).
- La "distancia" es una medida entre modelos de distintas Disciplinas (etapas). Se entiende "distancia" como una magnitud de la "diferencia entre unas cosas y otras" [Mundinteractivo04]

9.2 Contribución de la tesis

9.2.1 Objetivos cumplidos

En general, el presente trabajo caracteriza la Brecha Representacional entre las Disciplinas del desarrollo del software siguiendo el Proceso Unificado, mediante las Discrepancias Ontológicas. Esto, para encontrar defectos en las Disciplinas tempranas con lo que se evita su propagación a otros elementos y Disciplinas, obteniendo como resultado una reducción en el tiempo y el costo del desarrollo del software[42].

En particular, se localizó[43], identificó[44], midió[45] y redefinió[46] la Brecha Representacional entre el Modelo del Dominio y el Modelo del Diseño, caracterizando ciertos elementos, por orden de importancia: clases, paquetes y atributos. Además, se midió el error del modelador cuando se corrigieron los modelos. Coloquialmente, se detectó qué tanto corresponde lo que hizo el analista con lo que hizo el diseñador, y qué tan eficiente es el modelador.

---

[42] Este razonamiento de causa-efecto se explica con mayor detalle en [Humphrey95] y en "Costos de la identificación de defectos de software", página 43
[43] Ver "Capítulo 4. Localizando las Discrepancias" en la página 81
[44] Ver "Capítulo 5. Identificando las Discrepancias" en la página 85
[45] Ver "Capítulo 6. Midiendo las Discrepancias" en la página 104
[46] Ver "Definición de la Brecha Representacional" en la página 189

9.2.2 Contribuciones durante la investigación

A continuación se enlistan las contribuciones a detalle y algunas observaciones sobre la investigación:

- Preliminares (pág. 60)
  - Se justificó la selección de los modelos en función de la importancia de la propagación de defectos (pág. 60).
  - Se justificó la selección de los elementos en función de sus características de independencia, visualización y mutua composición (pág. 62).
  - Se ilustró la Brecha Representacional mediante diagramas de Venn (pág. 72).
  - Se propuso categorías de métricas dependiendo de las Disciplinas donde se encuentran las entidades (pág. 74):
    - Intradisciplinarias
    - Interdisciplinarias
    - Multidisciplinarias
  - Se explicó el trabajo con un ejemplo (i.e. de la facultad, pág. 79).
- Se localizaron los elementos participantes en las Discrepancias (pág. 81) mediante refinamientos (para paquetes y clases), que se representan en matrices de refinamiento (para paquetes, clases y atributos).
- Se identificaron las Discrepancias (pág. 85).
  - Se mejoró el modelo Bunge-Wand-Weber (pág. 86):
    - Se propuso una definición formal para identificar el tipo y el orden de las Discrepancias.
    - Se percibió la similaridad entre los tipos de Discrepancias. Se propuso una categorización de las Discrepancias en:
      - Abundantes en lo General
      - Abundantes en lo Particular
      - Representadas
      - No Representadas
    - Se percibió la Unión entre las Discrepancias Representadas.
    - Se percibió la complementariedad de las Discrepancias.
    - Se ilustró el modelo BWW y el modelo las mejoras propuestas.
  - Se propusieron guías y soluciones a ciertos problemas surgidos en los casos de estudio, dependiendo del elemento (pág. 98):
    - Paquetes. Guías para la correspondencia entre paquetes anidados
    - Atributos. Agregar el atributo por omisión, ObjectId, para establecer correspondencias entre clases sin atributos o sin atributos relacionados.
    - Relaciones. Guías para la consideración de las generalizaciones y asociaciones
- Se midió la Brecha Representacional mediante las Discrepancias (pág. 104)
  - Se propuso el algoritmo del Conteo Ponderado que considera las mejoras al modelo de BWW (pág. 104).
    - Se sugirieron guías razonadas y una restricción para la asignación de pesos.
    - Se interpretaron las métricas.
    - Se detalló el algoritmo con pseudocódigo

        - Se desarrolló un programa en el lenguaje R (pág. 210).
    - Se mencionaron observaciones y desventajas de otros métodos probados (pág. 119).
    - Se propusieron métricas sobre la eficiencia del modelador, cuando se comparan las métricas de un modelo con las métricas del modelo corregido (pág. 128).
    - Se mencionaron algunos beneficios colaterales del análisis de Discrepancias (pág. 136).
    - Se justificaron las correcciones a la Brecha Representacional de [Larman01] (pág. 136).
- Se probó el trabajo con dos casos de estudio (pág. 137).
- Se reportaron las consideraciones o rumbos más razonables del trabajo futuro (pág. 192).

Capítulo 10. Trabajo Futuro

Como se puede apreciar a continuación, se tienen varias líneas de investigación.

- Investigación de campo:
  - Metodología para el verificador en el establecimiento de los refinamientos.
    - Sugerir las condiciones en las que es o no válido un refinamiento.
  - Repercusiones en el Proceso Unificado en el Modelado del Dominio y el Diseño.
    - Detallar los pasos que se deben modificar del Proceso Unificado.
  - Evolución de las métricas.
    - Describir el comportamiento de las métricas a través del tiempo y comparar los cambios con respecto a los requerimientos.
    - Establecer estándares de comparación y sugerir ciertos valores para determinados dominios.
- Investigación teórico-práctica:
  - Diseño de patrones para controlar las Discrepancias
    - Detallar las condiciones más comunes para la eliminación y justificación de las Discrepancias
    - Por ejemplo, se detectó que la Redundancia de Construcciones de Subtipo es un patrón relacionado con una Discrepancia de la herencia. Es decir, cuando hay una herencia en el Modelo de Diseño pero no en el Modelo del Dominio, se incurre en una Redundancia.
  - Aplicación entre otras Disciplinas.
    - Las Disciplinas más susceptibles son el Diseño y la Implementación.
    - Aplicación a los elementos no seleccionados.
    - Definir las reglas y experimentar la propuesta para la Brecha entre asociaciones, generalizaciones, etc.
- Análisis estadístico:
  - Pruebas estadísticas para la comparación de los modelos.
    - Comparar una muestra de varias parejas de modelos de Dominio y de Diseño en distintos momentos (e.g. iteraciones, antes y después de las revisiones) para determinar estadísticamente qué tan grandes son los cambios.
  - Los órdenes de las Discrepancias Representadas.
    - Relacionar el esfuerzo estimado con el control del orden de las Discrepancias Representadas. Esto es, cuánto se tarda eliminar o justificar una Discrepancia para cada orden (e.g. 15 minutos por Discrepancia de orden 2, 120 minutos por Discrepancia de orden 6).
  - Si se desea más detalle en el comportamiento de los modeladores, se pueden tomar otros estadísticos como la desviación estándar, curtosis, sesgo, etc.
- Automatización:

- Automatización desde CASE.
  - Desarrollar un prototipo para medir la Brecha desde un modelador de UML (e.g. programar un script en Rational Rose).
- Establecimiento de refinamientos de forma semi-automática.
  - Desarrollar un prototipo que sugiera los refinamientos que parezcan más convenientes (e.g. distancia de Hamming entre los nombres de los elementos).

Apéndices

Apéndice A. Glosario

Término usado, concepto	Áreas: acrónimo; "translation"; sinónimos	Descripción, definición
completitud	Calidad: "completeness"; saciedad (no existe "completez")	Grado en que se cumplen los requisitos.
corrección	Calidad: "correctness"; exactitud	Ausencia de errores en el sistema. Que tan cerca el sistema está de la especificación. [Hogan97] Se entiende como la satisfacción del cliente.
defecto	Calidad	"Carencia o imperfección de las cualidades propias de algo" [Mundinteractivo04]
efectividad	Calidad:	"Capacidad para producir el efecto deseado." [Mundinteractivos04] "Eficacia + eficiencia = efectividad" [Pamplona02]
eficacia	Calidad:	"Capacidad para obrar o para conseguir un resultado determinado." [Mundinteractivos04] "La eficiencia enfatiza en la optima utilización de los recursos, en tanto que la eficacia se materializa en la obtención de resultados." [Pamplona02]
eficiencia	Calidad:	"Capacidad para lograr un fin empleando los mejores medios posibles: no siempre eficacia es sinónimo de eficiencia." [Mundinteractivos04] Mide el grado de optimización del sistema [Hogan97] Ver "eficacia"
mantenible	Calidad:	Se evalúa el impacto de un cambio. Un software es mantenible, si tiene alta probabilidad de que se cumplan las metas. [Fenton97]
mantenimiento, facilidad de	Calidad:	Facilidad de corregir los errores [Hogan97]
modelador	Calidad:	Dentro del contexto, se refiere al analista y diseñador que producen el Modelo del Dominio y el Modelo del Diseño, respectivamente. "Que modela o da forma". "Formar o configurar el carácter de acuerdo con unos rasgos o principios determinados" [Mundinteractivos04]
validación	Calidad:	Revisión de que el programa se apega a su especificación. [Sommerville96] "¿Estamos construyendo el producto correcto (indicado)?" [Sommerville96]
verificación	Calidad:	Revisión de que el programa, como se implementó, cumpla las expectativas del cliente del software. "¿Estamos construyendo el producto correctamente (bien)?" [Sommerville96]
verificador	Calidad:	Dentro del contexto, se refiere a quien realiza el análisis de las

		Discrepancias, que pudiera ser el analista, diseñador, líder del proyecto o el encargado de la calidad del sistema, ya sea interno o externo. "Comprobar la verdad o autenticidad de algo". [Mundinteractivos04]
cálculo	Métricas:	Es una cuantificación indirecta, en la que se toman mediciones y se les combinan en un elemento cuantificable que refleja algún atributo al que se trata de entender. Capítulo 1 en [Fenton97]
cuantificación	Métricas:	"Expresión numérica de una magnitud: la cuantificación de los daños nos llevará un tiempo." [Mundinteractivos04]
detectar	Métricas:	"Poner de manifiesto, mediante aparatos o por métodos físicos o químicos, lo que no puede ser observado directamente: han detectado indicios de plomo en las aguas. Captar, descubrir, percibir: no detectó tu ironía" [Mundinteractivos04]
directa, medición	Métricas:	Es la medición de un atributo de una entidad que no involucra a ningún otro atributo o entidad [Fenton97]
indirecta, medición	Métricas:	Es la medición de uno o varios atributo de una o varias entidades que involucran a otros atributos o entidades. Es usualmente útil en resaltar las interacciones entre las medidas directas. [Fenton97]
medición	Métricas:	Proceso en el que los números y los símbolos son asignados a los atributos de las entidades del mundo real, de tal forma que los describen de acuerdo a reglas definidas claramente. Es una cuantificación directa. Capítulo 1 en [Fenton97]
medida	Métricas:	Número o símbolo asignado a una entidad por la medición para caracterizarla. [Fenton97]
métrica	Métricas:	Desde la perspectiva del análisis matemático: Regla usada para describir que tan apartados se encuentran dos puntos. Formalmente, es una función *m* definida en pares de objetos *x* y *y*, tal que *m(x, y)* representa la distancia entre *x* y *y*. Las propiedades que satisfacen están en [Fenton97 p. 103]. Capítulo 1 en [Fenton97]
tipos de escalas de medición	Métricas:	Clasificación del sistema compuesto por la correspondencia ("mapping") de la medición, su sistema de relaciones (el dominio y el rango) numérica y empírica. Los tipos principales son: • Nominal • Ordinal • Intervalo • Razón • Absoluto [Fenton97]
Brecha Representacional	Ontología, UP: Brecha; "Representational Gap", "Semantic Gap"; Brecha Semántica	"El intervalo entre el modelo mental del dominio que pensamos y su representación en el software" [Larman01]. En UP, el modelo mental (el problema) se representa mediante el Modelo del Dominio; y el modelo del software (la solución), con el Modelo del Diseño.
Abundante en lo General	Ontología: Abundante en el Dominio	Categoría de las Discrepancias de Sobrecarga y Déficit. En este contexto, se eligió en vez de "sobrante" o "con residuos", pues los elementos de más pueden estar justificados.
Abundante en lo Particular	Ontología: Abundante en el Diseño	Categoría de las Discrepancias de Redundancia y Exceso. Ver "Abundante en lo General"

caracterización	Ontología:	En nuestro contexto, es la determinación de los atributos sobre identificación, localización y medición de la Brecha Representacional. "1. Determinación de los atributos peculiares de una persona o cosa, de modo que se distinga claramente de las demás: caracterización de los personajes. 2. Adecuación de un actor en sus rasgos físicos, mediante maquillaje y vestuario, al personaje que ha de representar: para la caracterización del personaje tuvo que pasar tres horas maquillándose." [Mundinteractivos04]
Concepto Ontológico	Ontología: "Ontological Concept".	Categoría de ideas o cosas del mundo real que se supone existen en alguna área de interés.
Construcción del Modelado	Ontología: "Modeling Construct", Construcción	Especificación formal explícita que representa a un Concepto Ontológico. Entidad que "en UML se utiliza para representar el dominio del problema, el proceso de desarrollo, el diseño intermedio o el sistema de información propuesto" [Opdahl02]
Discrepancias Ontológicas	Ontología: "Ontological Discrepancies"; Discrepancias	Coloquialmente, son las diferencias entre las representaciones de las cosas del dominio. Propiamente, las diferencias entre las Construcciones del Modelado y los Conceptos Ontológicos.
distancia cognitiva	Ontología:	"Cantidad de esfuerzo intelectual que se necesita gastar por los desarrolladores de software, para llevar un sistema de software de una etapa del desarrollo a otra. "De esta definición, debe ser claro que la distancia cognitiva no es una medida formal que pueda expresarse con números y unidades. Más bien, es una noción informal que se basa en la intuición sobre el esfuerzo relativo requerido para cumplir varias tareas del desarrollo del software." [Krueger92]
identificación	Ontología:	En nuestro contexto, se referirá al reconocimiento del tipo y el orden de la Discrepancia. "1. Reconocimiento de la identidad de alguien: procedan a la identificación de los detenidos. 2. Consideración de dos cosas distintas para que aparezcan como una misma: no debéis hacer esa identificación entre lectura y aburrimiento. 3. Dar a conocer la identidad propia, especialmente con algún documento: el policía requirió identificación a todos los afectados. 4. Documento de identidad: lo siento, no llevo encima la identificación." [Mundinteractivos04]
localización	Ontología: Detección	En nuestro contexto, se referirá a la determinación de los elementos de los modelos participantes en el refinamiento (relación de correspondencia). Con especial énfasis en la determinación de las Clases Conceptuales y de Diseño. "1. Determinación del lugar en que se halla una persona o cosa: los bibliotecarios se encargan de la localización de los libros. 2. Delimitación, ubicación: la localización del problema es esencial para su resolución." [Mundinteractivos04]
Ontología	Ontología:	"Una especificación formal explícita de cómo representar a los objetos, conceptos y otras entidades que se supone existen en alguna área de interés; y sobre las relaciones que se mantienen entre ellos. Para sistemas AI, lo que 'existe' es lo que puede ser representado. […] Podemos describir la ontología de un

		programa mediante la definición de un conjunto de términos representacionales" (Inteligencia Artificial retomado de la Filosofía) [Howe03] Coloquialmente, se refiere a cómo representar las cosas en un área de interés.
clase	OO, UML:	En UML, "descriptor de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamientos" en el glosario de [Larman01]
dependencia	OO, UML: "dependency"	"Indica una relación semántica entre dos o más elementos del modelo. Indica una situación en la que un cambio del elemento proveedor (fuente) puede requerir un cambio o indicar un cambio al significado del elemento cliente (destino) en la dependencia." [Rumbaugh99]
dependiente, elemento	OO, UML:	Necesita de uno (autoreferencia) o más elementos para existir. Se asume que es un elemento de modelado UML.
independiente, elemento	OO, UML:	Puede existir en el modelo de forma aislada, no necesita de la existencia de otros elementos. Se asume que es un elemento de modelado UML.
refinamiento	OO, UML: "refine"	"Enunciado de que una correspondencia existe entre elementos en dos niveles semánticos diferentes". "Es una relación entre dos versiones de un concepto en diferentes etapas de desarrollo o en diferentes niveles de abstracción. Los dos conceptos pueden no significar que coexistan en el modelo detallado final. Uno de ellos es usualmente una versión menos terminada de la otra. En principio, hay una correspondencia desde el concepto menos terminado hacia el concepto más terminado. Esto no significa que la traducción es automática. Usualmente, el concepto más detallado contiene decisiones de diseño hechas por el diseñador, decisiones que se pudieron haber hecho de muchas formas. En principio, los cambios a un modelo pueden ser validados contra el otro, con las desviaciones marcadas. En la práctica, las herramientas de hoy no pueden hacer todo esto, aunque algunas correspondencias simples pueden reforzarse. Por lo tanto, un refinamiento es mayormente un recordatorio al modelador que múltiples modelos se relacionan de una forma predictiva." [Rumbaugh99]
refinar	OO, UML:"refine"	En nuestro contexto, quien valida es el encargado de establecer las dependencias de refinamiento (flechas) entre las Clases Conceptuales y las de Diseño. "1. Hacer más fina o más pura una sustancia o materia, eliminando impurezas y mezclas: refinar el aceite. 2. Perfeccionar o educar. También prnl.: es una buena escritora, pero creo que aún debe refinar su estilo." [Mundinteractivos04]
Análisis	OO, UP: Dominio y Requerimientos	"Investigación de un dominio que resulta en los modelos que describen las características estáticas y dinámicas. Enfatiza preguntas del tipo 'qué' en vez del 'cómo'" glosario de [Larman01] Se empleará para abreviar, a las Disciplinas de Modelado del Dominio y de Requerimientos
Clase Conceptual	OO, UP: Concepto, Clase de Análisis, de Dominio, de	Ver "Concepto". Término empleado en [Larman01]

	Concepto	
Concepto	OO, UP: Clase Conceptual	"Una categoría de ideas o cosas. En este libro, usadas para designar las cosas del mundo real más que entidades de software. Una intención de concepto es una descripción de sus atributos, operaciones y semántica. Una extensión del concepto es el conjunto de instancias u objetos de ejemplo que son miembros del concepto. Frecuentemente se define como un sinónimo de clase de dominio " glosario de [Larman01]
objeto	OO:	"En UML, una instancia de una case que encapsula el estado y el comportamiento. Coloquialmente, un ejemplo de una cosa." glosario de [Larman01]
OOA	OO: "Object-Oriented Analysis"	"La investigación del dominio o sistema del problema en términos de los conceptos de dominio, como las clases conceptuales, asociaciones y cambios de estado" glosario de [Larman01]
OOD	OO: "Object-Oriented Design"	"La especificación de una solución lógica de software en términos de objetos de software, como sus clases, atributos, métodos y colaboraciones" glosario de [Larman01]
OOP	OO: "Object-Oriented Programming"	"Un lenguaje de programación que soporta los conceptos de encapsulamiento, herencia y polimorfismo" glosario de [Larman01]
Ciencias de la Computación	Otros:	Provee las bases teóricas para la construcción del software. Capítulo 1 en [Fenton97]
distancia	Otros	"Diferencia entre unas cosas y otras. Espacio o periodo de tiempo que media entre dos cosas o sucesos." [Mundinteractivo04]
enfoque ingenieril	Otros:	Cada actividad es entendida y controlada, de forma que hay pocas irregularidades (sorpresas) mientras se desarrolla el producto. Capítulo 1 en [Fenton97]
Ingeniería de Software	Otros:	Describe una colección de técnicas que aplican un enfoque ingenieril a la construcción y apoyo de productos de software. Incluye las actividades de administración, cálculo de costos, planeación, modelación, análisis, especificación, diseño, implementación, pruebas (experimentación) y mantenimiento. Se enfoca al implementación del software de una forma controlada y científica. Capítulo 1 en [Fenton97]
UML	UML, OO: Unified Modeling Language; Lenguaje del Modelado Unificado	Modelo de desarrollo ágil e iterativo
Disciplina	UP: "discipline", "phase" (en el SEI); etapa, nivel.	Son las áreas que se cubren en cada iteración del proceso UP. Antes se le llamaba "workflow". Diferente de la Fase de UP. Diferente del nivel del SEI.
Disciplinas tempranas	UP: etapas tempranas	Disciplina del Modelado del Dominio, Requerimientos y Diseño de UP
Diseño	UP:	"Un proceso que usa los productos de análisis para producir una especificación para implementar un sistema. Una descripción lógica de como un sistema trabajará" glosario de [Larman01]
Dominio	UP:	"Límite formal que define una materia en particular o área de interés" glosario de [Larman01]
Iniciación	UP:	Primera Fase del proceso UP

	"inception"	
Modelo del Dominio	UP: "Domain Model", "Conceptual Models" , "Domain Object Models", "Analysis Object Models"	"Representación visual de las clases conceptuales o de los objetos del mundo real en el dominio de interés" [Larman01] "El Modelo del Dominio provee un diccionario visual del vocabulario y de los conceptos del dominio, del que se obtiene inspiración para el nombrado de algunas cosas en el diseño del software"
Reglas del Dominio	UP: "business rules"	Antes se le llamaba "business", aunque es un área de mayor uso, el término no es tan general.
RUP	UP: Rational Unified Process; UP	Ver UP
UP	UP: Unified Process; Proceso Unificado	Proceso de desarrollo que (generalmente) prescribe UML como su notación. Se emplea en vez de RUP para mantenerlo abierto, no hacer referencia a la compañía (Rational)

Apéndice B. Glosario sobre grafos

Término usado, concepto	**Área: acrónimo; "translation"; sinónimos**	**Descripción, definición**
adyacente	Grafos:	Ver "vértices adyacentes", "aristas adyacentes"
arco	Grafos: arista dirigida	Es una arista, en la que uno de sus puntos finales es designado como la "cola" y el otro, la "cabeza"; la dirección es desde su cola a su cabeza. Definición. [Gross98]
arco múltiple	Grafos: "multiarc"	Es un conjunto de dos o más arcos teniendo la misma cola y la misma cabeza. Definición. [Gross98]
arista múltiple	Grafos: "multi-edge"	Es una colección de dos o más aristas que tienen puntos finales idénticos. Definición. [Gross98]
aristas adyacentes	Grafos:	Son dos aristas que tienen un punto final en común. Definición. [Gross98]
arista propia	Grafos:	Es una arista que no es un ciclo a sí mismo [Gross98]
ciclo a sí mismo	Grafos: "self-loop"	Es una arista que une un único punto final a sí mismo. Definición. [Gross98]
correspondencia	Grafos: "matching"; pareo, emparejamiento	Es un subconjunto de aristas de un grafo *G* que son mutuamente no adyacentes. Definición. [Gross98]
correspondencia en *G*	Grafos:	Es un conjunto *M* de aristas en un grafo *G*, en donde no hay dos aristas en el conjunto *M* que tengan un punto final en común. Revisión de definición. [Gross98]
correspondencia máxima	Grafos: correspondencia máxima del grafo	Es una correspondencia con el máximo número de aristas. Definición. [Gross98]
digrafo	Grafos: grafo dirigido	Es un grafo que tiene todas sus aristas dirigidas. Definición. [Gross98] "Un grafo dirigido o digrafo es el conjunto finito y no vacío *V* (de vértices) junto con un conjunto *E* (disjunto de *V*) de pares ordenados de distintos elementos de *V*. En este caso, se refiere a los elementos de *E* como arcos." [Behzad81] No confundir con "dígrafo" ("signo ortográfico compuesto de dos grafemas para representar un único sonido: la "ll" es un dígrafo" [Mundinteractivos04])
grado de un vértice	Grafos: "degree"; valencia	Es el número de aristas propias que inciden en un vértice *v* más dos veces el número de ciclos a sí mismo (ciclos auto referenciados, auto-ciclos). El grado de un vértice *v* en un grafo *G*, se denota por deg(*v*) –o grado(*v*)- Definición. [Gross98]
grafo	Grafos:	Un grafo $G = (V, E)$ es una estructura matemática consistente de dos conjuntos *V* y *E*. Los elementos de *V* son llamados "vértices" (o nodos) y los elementos de *E* son llamados "aristas" (o lados). Cada arista tiene un conjunto de uno o dos vértices asociados a ella, que son llamados sus puntos finales (o terminaciones). Definición. [Gross98]

grafo bipartito	Grafos:	Es un grafo *G* cuyo conjunto de vértices *V* puede ser particionado en dos subconjuntos *U* y *W*, tal que cada arista de *G* tiene un punto final en *U* y un punto final en *W*. El par *U*, *W* es llamado una bipartición (de vértices) de *G*, así como *U* y *W* son llamados los conjuntos de bipartición. Definición. [Gross98] Un grafo bipartito no puede tener ningún ciclo a sí mismo. Proposición. [Gross98].
grafo bipartito completo	Grafos:	Es un grafo bipartito simple, tal que cada vértice en uno de los conjuntos de bipartición se une cada vértice en el otro conjunto de partición. Cualquier grafo bipartito completo que tenga *m* vértices en una de los conjuntos de bipartición y *n* vértices en el otro, se denota por $K_{m,n}$. Definición. [Gross98]
grafo simple	Grafos:	Un grafo o digrafo es simple si no tiene ciclos a sí mismo, ni aristas múltiples. Definición. [Gross98]
múltiple	Grafos:	Ver "arista múltiple"
principio de correspondencia	Grafos: "correspondence principle"	Si existe una correspondencia uno a uno entre dos conjuntos, entonces tienen el mismo número de elementos. [Bogart83]
secuencia del grado de un grafo	Grafos:	Es la sucesión formada por los grados de los vértices colocados en orden no decreciente. Definición. [Gross98]
simple	Grafos:	Ver "grafo simple"
vecino de una arista	Grafos:	Un vecino de una arista *e* es otra arista que comparte uno o dos de sus puntos finales con *e*. Definición. [Gross98]
vértices adyacentes	Grafos:	Son dos vértices que están unidos por una arista. Definición. [Gross98]

Apéndice C. Metamodelo de UML

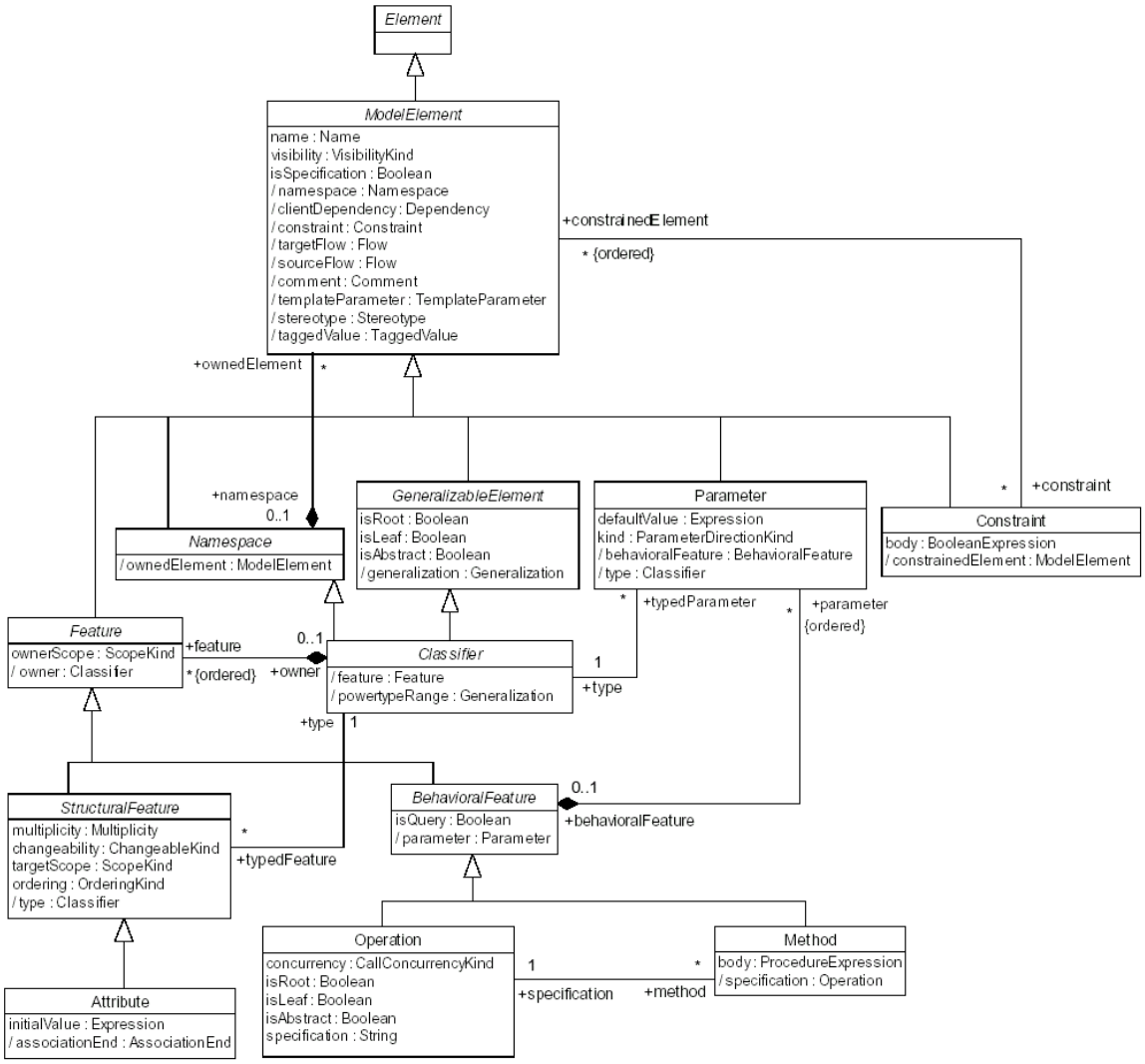


Figure 5-2 Core Package - Backbone

Figura 0.1. Paquete central. Base central

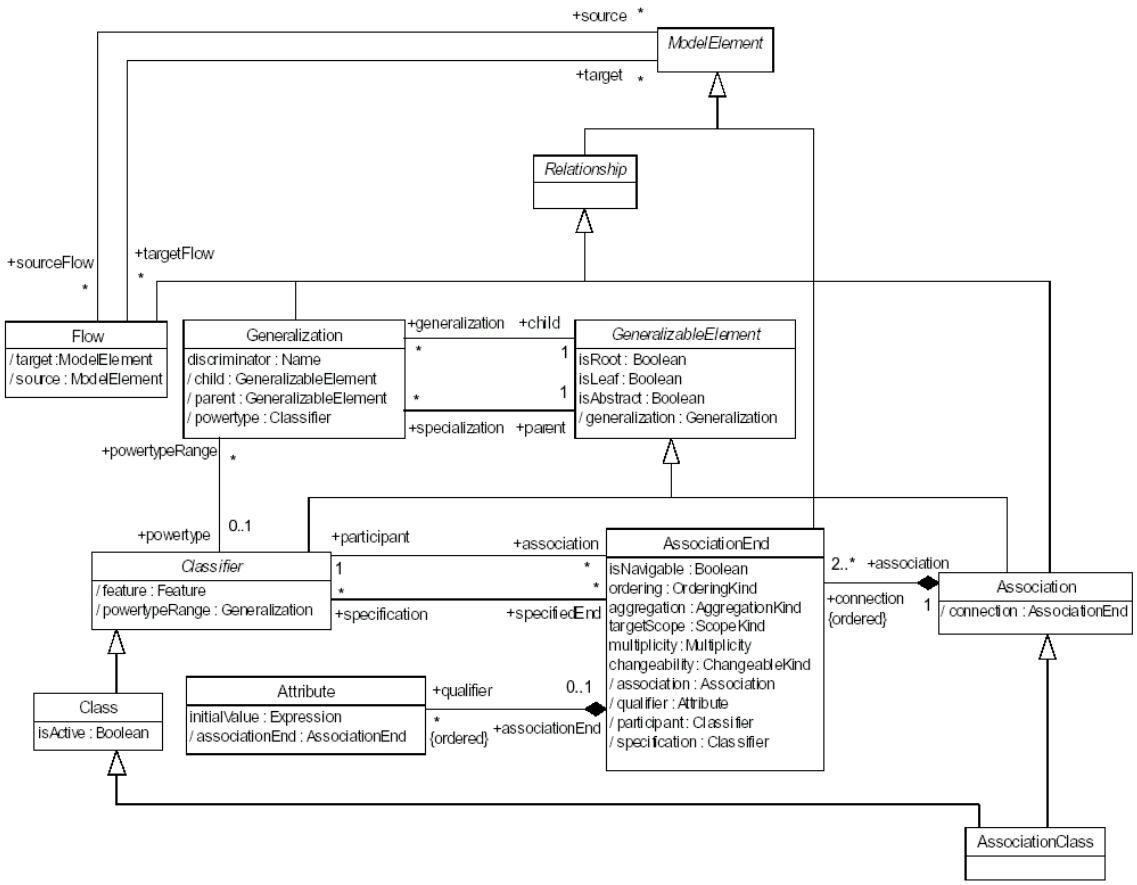


Figure 5-3 Core Package - Relationships

Figura 0.2. Paquete central. Relaciones

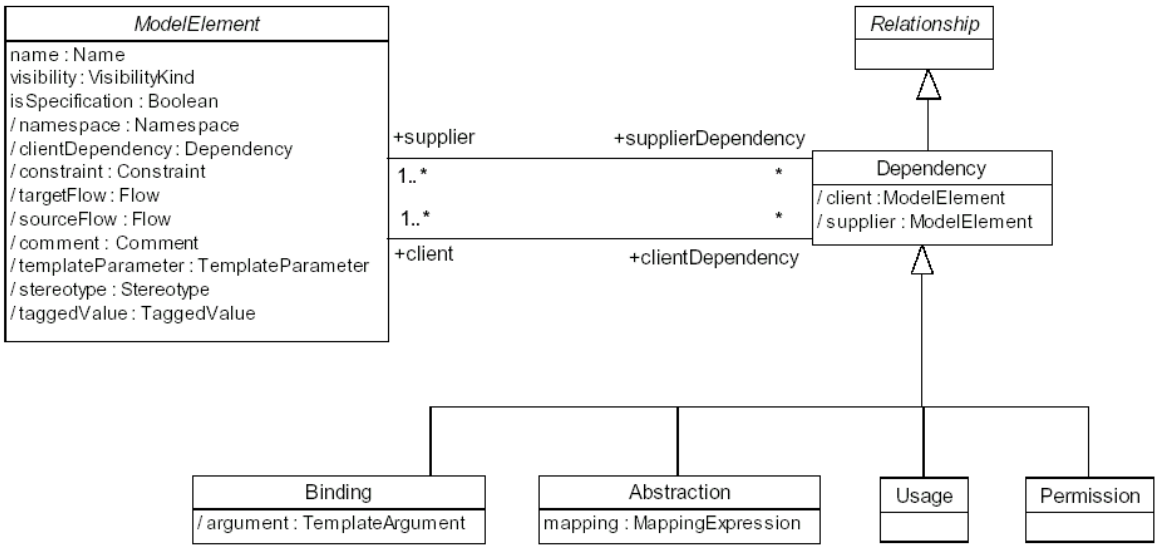


Figure 5-4 Core Package - Dependencies

Figura 0.3. Paquete central. Dependencias

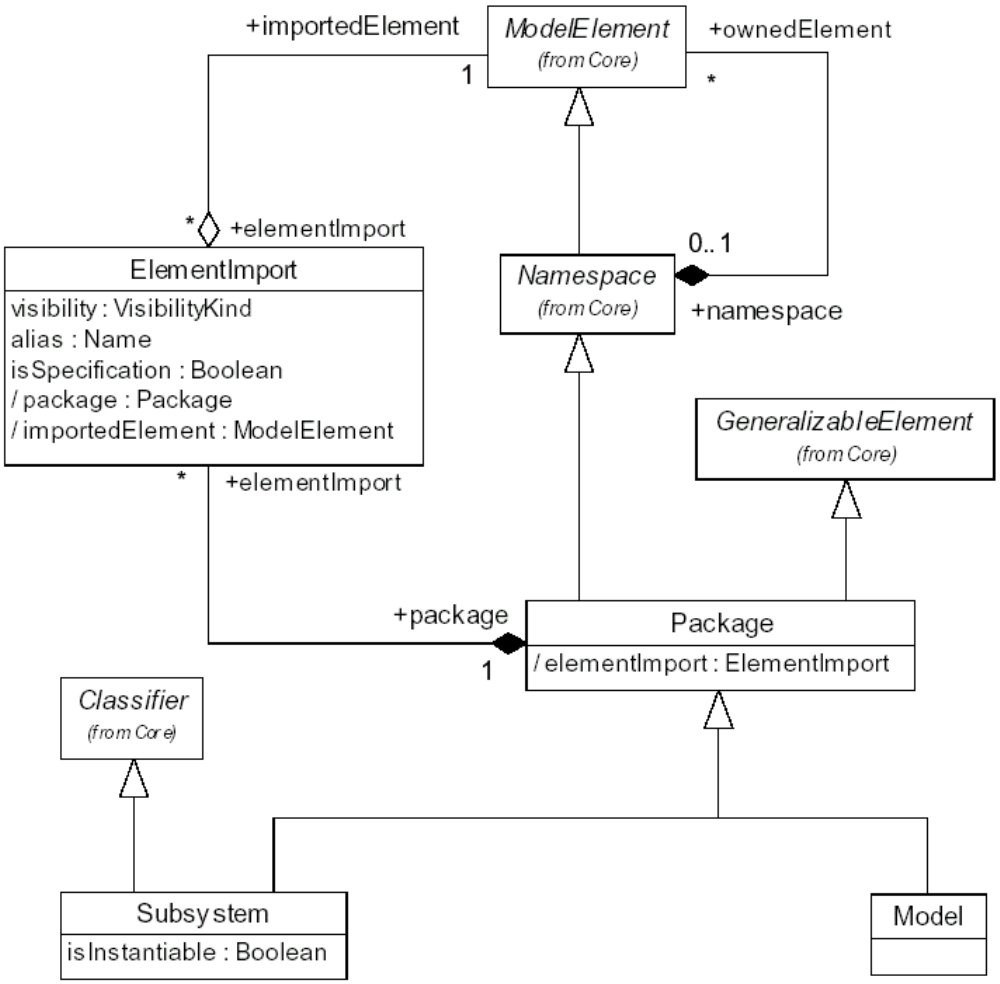


Figure 5-21 Model Management

Figura 0.4. Administración del Modelo

Apéndice D. Modelo del Dominio "NextGen POS (Point of Sale)"

Este caso de estudio se tomó de [Larman01]. Cabe notar que las clases usualmente tienen entre cero a tres atributos.

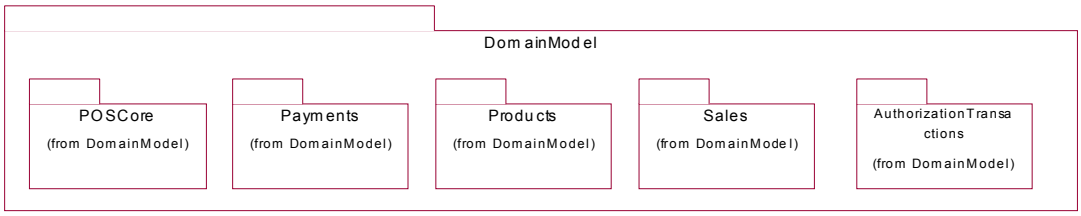


Figura 0.5. Paquetes del Modelo del Dominio "NextGen POS"

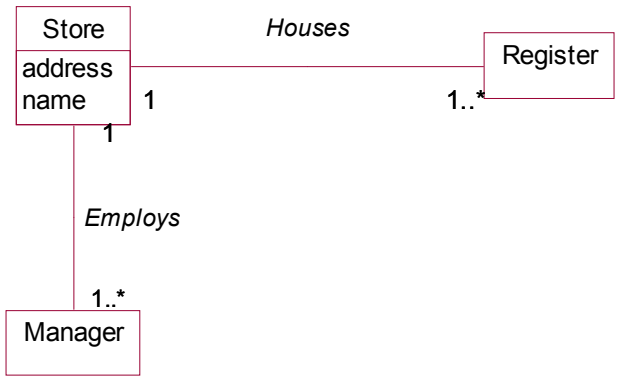


Figura 0.6. Diagrama de clases del paquete "POSCore", en el Modelo del Dominio "NextGen POS"

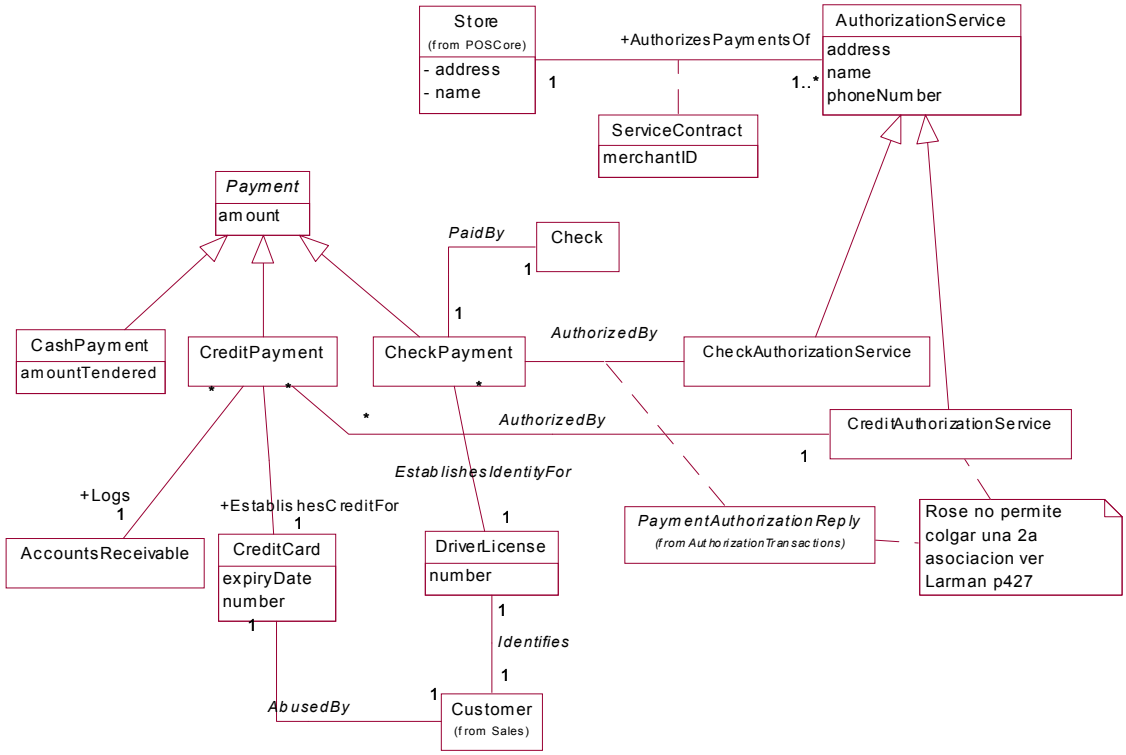


Figura 0.7. Diagrama de clases del paquete "Payments", en el Modelo del Dominio "NextGen POS"

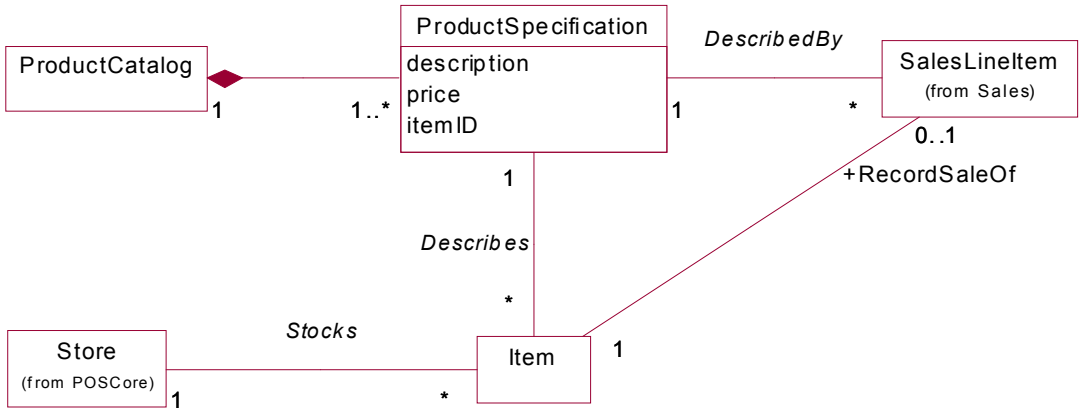


Figura 0.8. Diagrama de clases del paquete "Products", en el Modelo del Dominio "NextGen POS"

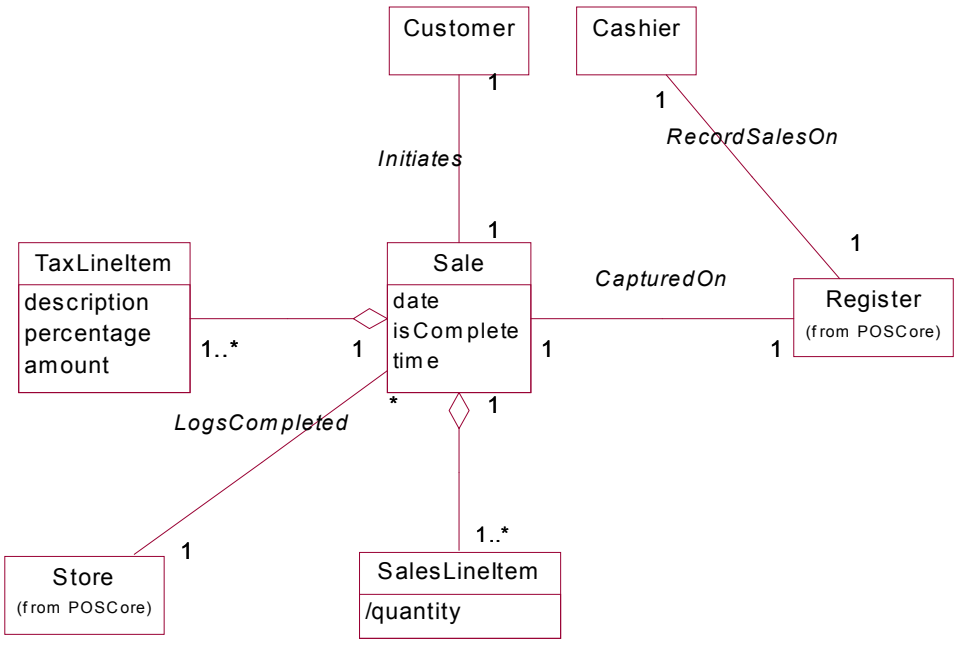


Figura 0.9. Diagrama de clases del paquete "Sales", en el Modelo del Dominio "NextGen POS"

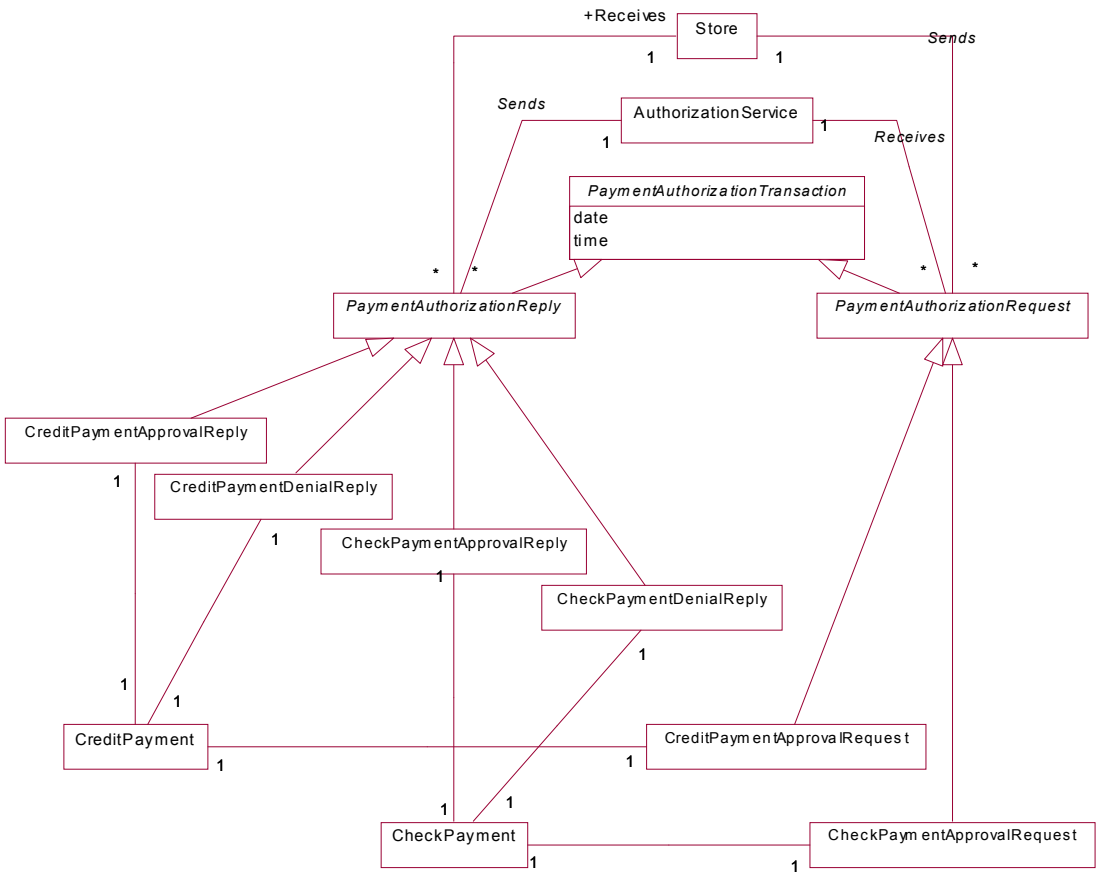


Figura 0.10. Diagrama de clases del paquete "AuthorizationTransaction", en el Modelo del Dominio "NextGen POS"

Apéndice E. Código en R

Se ofrece un listado del programa que calcula las métricas a partir de un archivo de texto. En la se enlistan las convenciones. R es un lenguaje y ambiente para cómputo estadístico y gráficas. Dado que es un proyecto GNU es sofware libre. Para ejecutar el siguiente programa, se puede bajar una copia de http://www.r-project.org/

Tabla 0.1. Convenciones del programa que calcula las métricas de la Brecha Representacional

Prefijo	Descripción
m	Matriz
v	Vector
s	Cadena (string)
ret	Regresa el objeto más representativo de la función (return)
n	Número de elementos

```
# Calculates metrics of software for the Representational Gap
# between elements of two models of similar nature
# @author Omar Posada Villarreal
# @version 1.3.1.1 2004.03.10 fix bug: max showed inverted
#       in metrSummDisc, weigMetrSummDisc
# @version 1.3.1 2004.02.26 fix bugs: min, max, validation contribution
# @version 1.3 2004.02.12 Min, max, mean for Over, Red
# @version 1.2 2004.02.12 DEORU, contribution

# NOTES:
# Ensure that R is in the correct path: R > File > Change Dir
# return First, the (compound) past data
# D:/Posada/TesisMetricas/AvancesMetricas
# mR: Conceptual Classes by rows; Design Classes by columns
# [      d[1] ... d[nD] ]
# [ c[1]  0       1     ]
# [ ...                 ]
# [ c[nC] 1       1     ]

# HEADER
# DEORU ((DefExcOverRed)Uni)
# ret: Returns the main result
#
#locDisc = function(mR) {
#counDisc = function(mR,
#       retCounDisc = function(mR,
#ratDisc = function(mR,
#       retRatDisc = function(mR,
#uniDiscSimp = function(mR,
#metrSummDisc = function(mR,
#weigDisc = function(vWeig, vIndRat) {
#weigMetrSummDisc = function(mR,
#joinOverRed = function(mR, vsOver, vsRed,
#strDisc = function(mR,
#       retStrDisc = function(mR,
#joinStrDisc = function(mR,

# Locate discrepancies
locDisc = function(mR) {
        ## Variable declaration
        nConc = nrow(mR)
        nDes = ncol(mR)
```

```
        # 0: no disc. 1: exists disc
        vDef = array(0, nConc) # aux col
        vExc = array(0, nDes)  #aux row
        vOver = array(0, nDes) #aux row
        vRed = array(0, nConc) # aux col

        # 0: no disc 2..n: order of disc
        vOrdOver = array(0, nDes)     #aux row
        vOrdRed = array(0, nConc)     # aux col

        ## Functionality
        vConcSum = rowSums(mR)
        vDesSum = colSums(mR)

        # Locate redundancies (with order) and deficits
        for (i in 1:nConc) {
                if (vConcSum[i] > 1) {
                        vOrdRed[i] = vConcSum[i]
                        vRed[i] = 1
                } else if (vConcSum[i] < 1) {
                        vDef[i] = 1
                }
        }

        # Locate overloads (with order) and excess
        for (j in 1:nDes) {
                if (vDesSum[j] > 1) {
                        vOrdOver[j] = vDesSum[j]
                        vOver[j] = 1
                } else if (vDesSum[j] < 1) {
                        vExc[j] = 1
                }
        }

        return (nConc, nDes,
                vDef, vExc, vOver, vRed, vOrdOver, vOrdRed)
}

# Count discrepancies
# @param loc Must be validated
counDisc = function(mR,
                loc = locDisc(mR)) {
        # Count redundancies and deficits, and sum their order
        nDef = sum(loc$vDef)
        ordRed = sum(loc$vOrdRed)
        nRed = sum(loc$vRed)

        # Count overloads and excess, and sum their order
        nExc = sum(loc$vExc)
        ordOver = sum(loc$vOrdOver)
        nOver = sum(loc$vOver)

        nMod = nDef + nExc + nOver + nRed

                # Reduce variable path ($)
                nConc = loc$nConc
                nDes = loc$nDes
        return (nConc, nDes, loc,
                nDef, nExc, nOver, nRed, nMod, ordOver, ordRed)
}

# ret: Returns the main result
retCounDisc = function(mR,
                loc = locDisc(mR)) {
        coun = counDisc(mR, loc)
        vDEORM = c(coun$nDef, coun$nExc, coun$nOver, coun$nRed, coun$nMod)
        return (vDEORM)
}

# Calculate discrepancy ratios
# @return ratDisc Adjusted ratio average
```

```
ratDisc = function(mR,
                coun = counDisc(mR)) {

        ratDef = coun$nDef / coun$nConc
        ratExc = coun$nExc / coun$nDes
        ratOver = coun$nOver / coun$nDes
        ratRed = coun$nRed / coun$nConc

        ratMod = (ratDef + ratExc + ratOver + ratRed) / 2
        # 2 (complementary phenomena)

                # Reduce variable path ($)
                nConc = coun$nConc
                nDes = coun$nDes
                loc = coun$loc
        return (nConc, nDes,
                loc, coun,
                ratDef, ratExc, ratOver, ratRed, ratMod)
}

# ret: Returns the main result
retRatDisc = function(mR,
                coun = counDisc(mR)) {
        rat = ratDisc(mR, coun)
        vRatDEORM = c(rat$ratDef, rat$ratExc, rat$ratOver, rat$ratRed, rat$ratMod)
        return (vRatDEORM)
}

# Measure the united (cross) discrepancies.
# @version simple. 2004.01. (30 LOC) Counts the union and crosses.
uniDiscSimp = function(mR,
                loc = locDisc(mR)) {
        ## Variable declaration
        # Look up the cross
        nOne = 0
        nCros = 0
        mCros = matrix(0, loc$nConc, loc$nDes)

        ## Functionality
        # for each Red, check if is a cross RedOver
        for (i in 1:loc$nConc) {
                if (loc$vRed[i]) { # == 1) {
                        for (j in 1:loc$nDes) {
                                if (loc$vOver[j] && mR[i, j]) {       # == 1) {
                                        nCros = nCros + 1      # mR = 1 (no 2)
                                        mCros[i, j] = 1
                                } # if
                        }
                } # if
        }

        # For counting arc members of the union, count 1to1 arcs
        for (i in 1:loc$nConc) {
                for (j in 1:loc$nDes) {
                        if ( (mR[i, j] == 1) && (loc$vRed[i] == 0)
                        && (loc$vOver[j] == 0)) {
                                nOne = nOne + 1
                        }
                }
        }

        # Count arc members of the union
        nUni = sum(mR) - nOne

        # nUni # mR = 4 (NO 6)
        nMaxUni = loc$nConc * loc$nDes
        # SIMPLE
        ratCros = nCros / nMaxUni
        # CHAIN
        ratUni = nUni / nMaxUni
```

```
        return (mCros, nCros, ratCros, nUni, nMaxUni, ratUni)
}

# ret: Returns the main result
# No apply

# Removes the zeros from a vector, reducing the length of it
# A zero is considered as macheps > abs(x)
delZero = function(vWithZero, macheps = 1e-10) {
        len = length(vWithZero)
        j = 1
        vNoZero = array(0, 1)
        for (i in 1:len) {
                if (abs(vWithZero[i]) >= macheps) {
                        vNoZero[j] = vWithZero[i]
                        j = j + 1
                }
        }
        return (vNoZero)
}

# Order basic statistics
ordStat = function(vOrd) {
        vNoZero = delZero(vOrd)
        meanOrd = mean(vNoZero)
        minOrd = min(vNoZero)
        maxOrd = max(vNoZero)
        return (meanOrd, minOrd, maxOrd)
}


# Summary of metric discrepancies
# @version 2004.02.15 mean, min, max of orders. Ajusted Ratio
metrSummDisc = function(mR,
                rat = ratDisc(mR)) {

        vsDisc = c("Def", "Exc", "Over", "Red", "Mod")
        vsCol   =   c("MeanOrder",   "MinOrder",   "MaxOrder",   "Count",   "Max",   "Ratio",
"AdjRatio")

        ordOver = ordStat(rat$loc$vOrdOver)
        ordRed = ordStat(rat$loc$vOrdRed)

        vMeanOrd = c(0, 0, ordOver$meanOrd, ordRed$meanOrd, 0)
        vMinOrd = c(0, 0, ordOver$minOrd, ordRed$minOrd, 0)
        vMaxOrd = c(0, 0, ordOver$maxOrd, ordRed$maxOrd, 0)

        vCoun = c(rat$coun$nDef, rat$coun$nExc,
                rat$coun$nOver, rat$coun$nRed, rat$coun$nMod)
        vRat = c(rat$ratDef, rat$ratExc, rat$ratOver, rat$ratRed, 0)        # No apply
        vAdjRat = 1/2 * vRat
        vAdjRat[5] = rat$ratMod       # 5=Last element

        vMax = c(rat$nConc, rat$nDes, rat$nDes, rat$nConc, 0)       # Maximum possible

        mSumm = cbind(vMeanOrd, vMinOrd, vMaxOrd, vCoun, vMax, vRat, vAdjRat)
        rownames(mSumm) = vsDisc
        colnames(mSumm) = vsCol

        return (mSumm)
}

# ret: Returns the main result
# No apply

# Calculates the relative weight and
# the adjusted weight of each Discrepancy
# @param vWeig Absolute weights
# @param vIndRat Individual ratios: DEORUM
# @return if WeigMod == 0, then contribution is 1/5=0.2
weigDisc = function(vWeig, vIndRat) {
```

```
        vNormIndWeig = vWeig / sum(vWeig)
        vNormWeig = c(vNormIndWeig, 1)        # By definition

        # Adj: Adjusted by complementarity. Factor 2
        vAdjNormIndWeig = vNormIndWeig * 2
        vAdjNormWeig = c(vAdjNormIndWeig, 2)  # 2=By complementarity

        vAdjIndWeigDisc = vAdjNormIndWeig * vIndRat
        weightDisc = sum(vAdjIndWeigDisc)
        vWeigDisc = c(vAdjIndWeigDisc, weightDisc)

        # Relative contribution
        if (weightDisc > 0) {
                vRelCont = vWeigDisc / weightDisc
        } else {
                vRelCont = 0.2 # 1/5 discrepancies types
        }
        vRelCont[6] = 1               # By definition = 1, round purpose
        return (vNormWeig, vAdjNormWeig, vWeigDisc, vRelCont)
}

# ret: Returns the main result
retWeigDisc = function(vWeig, vIndRat) {
        weig = weigDisc(vWeig, vIndRat)
        return (weig$vWeigDisc)
}

# Summary of weighted metric discrepancies.
# @param vWeig
# Suggested 2004.02.11
# vWeig = c(20 Def, 10 Exc, 18 Over, 9 Red, 3 Uni)   DEORU
#
# Suggested
# vWeig = c(3 Red, 3 Over, 4 Def, 4 Exc, 2 Uni)      RODEU
# vWeig = c(3, 3, 4, 4, 2),
#
# Equivalent to metrSummDisc
# vWeig = c(1 Red, 1 Over, 1 Def, 1 Exc, 0 Uni) RODEU
# vWeig = c(1, 1, 1, 1)
# @version 2004.02.15 mean, min, max of orders. Ajusted Ratio
# @version union
weigMetrSummDisc = function(mR,
                vWeig = c(20, 10, 18, 9, 3),
                rat = ratDisc(mR)) {
        vsDisc = c("Def", "Exc", "Over", "Red", "Uni", "Mod")
        vsCol = c("MeanOrder", "MinOrder", "MaxOrder",
                "Count", "Max", "Ratio", "AdjRatio",
                "NormWeight", "AdjWeight", "WeightDisc", "Contribution")

        uni = uniDiscSimp(mR)

        ordOver = ordStat(rat$loc$vOrdOver)
        ordRed = ordStat(rat$loc$vOrdRed)

        vMeanOrd = c(0, 0, ordOver$meanOrd, ordRed$meanOrd, 0, 0)
        vMinOrd = c(0, 0, ordOver$minOrd, ordRed$minOrd, 0, 0)
        vMaxOrd = c(0, 0, ordOver$maxOrd, ordRed$maxOrd, 0, 0)

        vCoun = c(rat$coun$nDef, rat$coun$nExc, rat$coun$nOver, rat$coun$nRed,
                uni$nUni, rat$coun$nMod)
        vMax = c(rat$nConc, rat$nDes, rat$nDes, rat$nConc,
                uni$nMaxUni, 0)        # Maximum possible

        # Ind: Individual ratios
        vIndRat = c(rat$ratDef, rat$ratExc, rat$ratOver, rat$ratRed,
                uni$ratUni)
        vRat = c(vIndRat, 0)   # No weighted discrepancie ratio
        vAdjRat = 1/2 * vRat
        vAdjRat[6] = rat$ratMod       # 6=Last element

        weig = weigDisc(vWeig, vIndRat)
```

```
        mSumm = cbind(vMeanOrd, vMinOrd, vMaxOrd, vCoun, vMax, vRat, vAdjRat,
                weig$vNormWeig, weig$vAdjNormWeig, weig$vWeigDisc, weig$vRelCont)
        rownames(mSumm) = vsDisc
        colnames(mSumm) = vsCol

        return (mSumm)
}

# ret: Returns the main result
# No apply

# Mix Red and Over discrepancies
# @param vsOver Declared: array(sNoArc, rat$nDes)
# @param vsRed Declared:  array(sNoArc, rat$nConc)
joinOverRed = function(mR, vsOver, vsRed,
                useRowColNames = TRUE,
                sOne = "1To1", sNoArc = "-", sDiscSep = "",
                rat = ratDisc(mR)) {
        msOverRed = matrix(sNoArc, rat$nConc, rat$nDes)

        # Copy names from mR
        if (useRowColNames == TRUE) {
                rownames(msOverRed) = rownames(mR)
                colnames(msOverRed) = colnames(mR)
        }

        #sAux = "" # Trace
        for (i in 1:rat$nConc) {
         for (j in 1:rat$nDes) {
                if (mR[i, j]) {        # == 1) {
                        if (rat$loc$vRed[i]) {
                                if (rat$loc$vOver[j]) {
                                        msOverRed[i, j] = paste(
                                                vsOver[j], vsRed[i],
                                                sep = sDiscSep)
                                        #sAux = paste(sAux, "OR")
                                } else {
                                        msOverRed[i, j] = vsRed[i]
                                        #sAux = paste(sAux, "R")
                                }
                        } else {
                                if (rat$loc$vOver[j]) {
                                        msOverRed[i, j] = vsOver[j]
                                        #sAux = paste(sAux, "O")
                                } else {
                                        msOverRed[i, j] = sOne
                                        #sAux = paste(sAux, "1")
                                }
                        } #if vRed
                } # if mR
         }
         #sAux = paste(sAux, "/")
        }
        return (msOverRed)
}

# String representation of the discrepancies. Identify discrepancies.
# Local diagnosis RDOE
# @param sOne String for 1 to 1 relationship
# @param sNoArc String for showing when there isn't a link
# @param sDiscSep Separator of Over and Red
# @param vsDisc Names of discrepancies in order: Def, Exc, Over, Red
# @version 2004.01 (48 LOC separate???)
strDisc = function(mR,
                useRowColNames = TRUE,
                sOne = "1To1", sNoArc = "-", sDiscSep = "",
                vsDisc = c("D", "E", "O", "R"),
                rat = ratDisc(mR)) {

        vsDef = array(sNoArc, rat$nConc)      # aux col
```

```
        vsExc = array(sNoArc, rat$nDes)       # aux row
        vsOver = array(sNoArc, rat$nDes)      # aux row
        vsRed = array(sNoArc, rat$nConc)      # aux col

        # Locate redundancies and deficits
        # Red and Def are mutually excluding
        for (i in 1:rat$nConc) {
                if (rat$loc$vRed[i]) { # == 1) {
                        vsRed[i] = paste(vsDisc[4], rat$loc$vOrdRed[i],
                                sep = "") # [4] = Red
                } else if (rat$loc$vDef[i] == 1) {    # == 1) {
                        vsDef[i] = vsDisc[1]   # [1] = Def
                }
        }

        # Locate overloads and excess
        # Over and Exc are mutually excluding
        for (j in 1:rat$nDes) {
                if (rat$loc$vOver[j]) {       # == 1) {
                        vsOver[j] = paste(vsDisc[3], rat$loc$vOrdOver[j],
                                sep = "") # [3] = Over
                } else if (rat$loc$vExc[j]) { # == 1) {
                        vsExc[j] = vsDisc[2]   # [2] = Exc
                }
        }

        msOverRed = joinOverRed(mR, vsOver, vsRed, useRowColNames,
                sOne, sNoArc, sDiscSep, rat)
        return (rat, vsDef, vsExc, vsOver, vsRed, msOverRed)
}

# ret: Returns the main result
retStrDisc = function(mR,
                useRowColNames = TRUE,
                sOne = "1To1", sNoArc = "-", sDiscSep = "",
                vsDisc = c("D", "E", "O", "R"),
                rat = ratDisc(mR)) {
        str = strDisc(mR, useRowColNames, sOne, sNoArc, sDiscSep,
                vsDisc, rat)
        return (str$msOverRed)
}

# Joins msOverRed, vsDef on right, vsExc on bottom
# @param sCorner Right bottom corner separator
# @see strDisc for other parameters
# @throw Possible error if mR has no rownames or colnames
joinStrDisc = function(mR,
                useRowColNames = TRUE,
                sOne = "1To1", sNoArc = "-", sDiscSep = "",
                vsDisc = c("D", "E", "O", "R"),
                sDef = "Def", sExc = "Exc",
                sCorner = "",
                sDisc = strDisc(mR, useRowColNames,
                        sOne, sNoArc, sDiscSep,       vsDisc)) {

        msDisc = cbind(sDisc$msOverRed, sDisc$vsDef)
        msDisc = rbind(msDisc, c(sDisc$vsExc, sCorner))

        # Name rows and cols
        # Last name is Def or Exc
        if (useRowColNames == TRUE) {
                vsRow = c(rownames(sDisc$msOverRed), sExc)
                vsCol = c(colnames(sDisc$msOverRed), sDef)
                rownames(msDisc) = vsRow
                colnames(msDisc) = vsCol
        }

        return (msDisc)
}

# ret: Returns the main result
```

```
# No apply

#------------------------------------------------------------
#
# Test
#
mR    =    read.table("D:/Posada/TesisMetricas/AvancesMetricas/MetricInR/EjemIn/Ejem4.dat",
header = TRUE)
mR

# { Test
#   { They autocall
loc = locDisc(mR)
loc
coun = counDisc(mR)
coun
rat = ratDisc(mR)
rat
#   }

mMetr = metrSummDisc(mR)
mMetr
mWM = weigMetrSummDisc(mR)
mWM
mWMEqu = weigMetrSummDisc(mR, vWeig = c(4, 4, 4, 4, 0),)
mWMEqu

cd = uniDiscSimp(mR)
cd

#   { They autocall
sd = strDisc(mR)
sd
msDisc = joinStrDisc(mR)
msDisc
#   }

#   { ret
vCoun = retCounDisc(mR)
vCoun

vRat = retRatDisc(mR)
vRat

vWeig = c(1, 1, 1, 1, 0)
vIndRat = c(0.2, 0.4, 0.6, 0.8, 0.1)
vWeigDisc = retWeigDisc(vWeig, vIndRat)
vWeigDisc

vsSD = retStrDisc(mR)
vsSD
#   } ret
# } Test

#------------------------------------------------------------
#
# Main
#
# Input. Entrada
mR  =  read.table("D:/Posada/TesisMetricas/AvancesMetricas/MetricInR/EjemIn/Ejem4Corr.dat",
header = TRUE)
#mR   =   read.table("D:/Posada/TesisMetricas/AvancesMetricas/MetricInR/EjemIn/Ejem4.dat",
header = TRUE)
mR

# Output. Salida. Resumen de metricas
# options(digits = 4)
mWM = weigMetrSummDisc(mR)
mWM

# Output. Salida. Representacion textual
```

```
msDisc = joinStrDisc(mR)
msDisc

#END--------------------------------------------
```

Apéndice F. Tablas extras de los casos de estudio

F.1 Caso de matrices

Paquetes

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def				0	2	0%	0%	33%	67%	0%	0%
Exc				0	1	0%	0%	17%	33%	0%	0%
Over	2	2	2	1	1	100%	50%	30%	60%	60%	86%
Red	0	0	0	0	2	0%	0%	15%	30%	0%	0%
Uni				2	2	100%	50%	5%	10%	10%	14%
Mod				1		0%	50%	100%	200%	70%	100%

Clases

	Mean Order	Min Order	Max Order	Count	Max	Ratio	AdjRatio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def				0	13	0%	0%	33%	67%	0%	0%
Exc				2	9	22%	11%	17%	33%	7%	14%
Over	3	2	5	6	9	67%	33%	30%	60%	40%	75%
Red	4	3	5	2	13	15%	8%	15%	30%	5%	9%
Uni				18	117	15%	8%	5%	10%	2%	3%
Mod				10		0%	52%	100%	200%	54%	100%

Atributos

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def				0	20	0%	0%	33%	67%	0%	0%
Exc				6	25	24%	12%	17%	33%	8%	21%
Over	2.63	2	5	8	25	32%	16%	30%	60%	19%	50%
Red	2.71	2	5	7	20	35%	18%	15%	30%	11%	27%
Uni				27	500	5%	3%	5%	10%	1%	1%
Mod				21		0%	46%	100%	200%	38%	100%

F.2 Caso de escolar resumido

Métricas

Paquetes

Elaboración 1

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	1	0%	0%	20%
Exc	0	1	0%	0%	20%
Over	0	1	0%	0%	20%
Red	0	1	0%	0%	20%
Uni	0	1	0%	0%	20%
Mod	0	0	0%	0%	100%

Elaboración 2

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	2	0%	0%	0%
Exc	1	2	50%	17%	32%
Over	1	2	50%	30%	58%
Red	0	2	0%	0%	0%
Uni	2	4	50%	5%	10%
Mod	2	0	50%	52%	100%

Elaboración 3

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	4	0%	0%	0%
Exc	1	5	20%	7%	100%
Over	0	5	0%	0%	0%
Red	0	4	0%	0%	0%
Uni	0	20	0%	0%	0%
Mod	1	0	10%	7%	100%

Clases

Elaboración 1

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	3	0%	0%	0%
Exc	2	5	40%	13%	100%
Over	0	5	0%	0%	0%
Red	0	3	0%	0%	0%
Uni	0	15	0%	0%	0%
Mod	2	0	20%	13%	100%

Elaboración 2

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	7	0%	0%	0%
Exc	3	9	33%	11%	27%
Over	3	9	33%	20%	49%
Red	2	7	29%	9%	21%
Uni	8	63	13%	1%	3%
Mod	8	0	48%	41%	100%

Elaboración 3

	Count	Max	Ratio	WeightDisc	Contribution
Def	2	19	11%	7%	16%
Exc	3	14	21%	7%	17%
Over	5	14	36%	21%	50%
Red	4	19	21%	6%	15%
Uni	17	266	6%	1%	2%
Mod	14	0	44%	43%	100%

Atributos

Elaboración 1

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	8	0%	0%	0%
Exc	7	15	47%	16%	100%
Over	0	15	0%	0%	0%
Red	0	8	0%	0%	0%
Uni	0	120	0%	0%	0%
Mod	7	0	23%	16%	100%

Elaboración 2

	Count	Max	Ratio	WeightDisc	Contribution
Def	0	17	0%	0%	0%
Exc	11	30	37%	12%	51%
Over	3	30	10%	6%	25%
Red	3	17	18%	5%	22%
Uni	12	510	2%	0%	1%
Mod	17	0	32%	24%	100%

Elaboración 3

	Count	Max	Ratio	WeightDisc	Contribution
Def	4	53	8%	5%	14%
Exc	11	42	26%	9%	24%
Over	12	42	29%	17%	47%
Red	10	53	19%	6%	15%
Uni	44	2226	2%	0%	1%
Mod	37	0	41%	37%	100%

F.3 Caso de escolar completo

Paquetes

Elaboración 1

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	1	0%	0%	33%	67%	0%	20%
Exc	0	0	0	0	1	0%	0%	17%	33%	0%	20%
Over	0	0	0	0	1	0%	0%	30%	60%	0%	20%
Red	0	0	0	0	1	0%	0%	15%	30%	0%	20%
Uni	0	0	0	0	1	0%	0%	5%	10%	0%	20%
Mod	0	0	0	0	0	0%	0%	100%	200%	0%	100%

Elaboración 2

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	2	0%	0%	33%	67%	0%	0%
Exc	0	0	0	1	2	50%	25%	17%	33%	17%	32%
Over	2	2	2	1	2	50%	25%	30%	60%	30%	58%
Red	0	0	0	0	2	0%	0%	15%	30%	0%	0%
Uni	0	0	0	2	4	50%	25%	5%	10%	5%	10%
Mod	0	0	0	2	0	0%	50%	100%	200%	52%	100%

Elaboración 3

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	4	0%	0%	33%	67%	0%	0%
Exc	0	0	0	1	5	20%	10%	17%	33%	7%	100%
Over	0	0	0	0	5	0%	0%	30%	60%	0%	0%
Red	0	0	0	0	4	0%	0%	15%	30%	0%	0%
Uni	0	0	0	0	20	0%	0%	5%	10%	0%	0%
Mod	0	0	0	1	0	0%	10%	100%	200%	7%	100%

Clases

Elaboración 1

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	3	0%	0%	33%	67%	0%	0%
Exc	0	0	0	2	5	40%	20%	17%	33%	13%	100%
Over	0	0	0	0	5	0%	0%	30%	60%	0%	0%
Red	0	0	0	0	3	0%	0%	15%	30%	0%	0%
Uni	0	0	0	0	15	0%	0%	5%	10%	0%	0%
Mod	0	0	0	2	0	0%	20%	100%	200%	13%	100%

Elaboración 2

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	7	0%	0%	33%	67%	0%	0%
Exc	0	0	0	3	9	33%	17%	17%	33%	11%	27%
Over	2	2	2	3	9	33%	17%	30%	60%	20%	49%
Red	2	2	2	2	7	29%	14%	15%	30%	9%	21%
Uni	0	0	0	8	63	13%	6%	5%	10%	1%	3%
Mod	0	0	0	8	0	0%	48%	100%	200%	41%	100%

Elaboración 3

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	2	19	11%	5%	33%	67%	7%	16%
Exc	0	0	0	3	14	21%	11%	17%	33%	7%	17%
Over	3	2	5	5	14	36%	18%	30%	60%	21%	50%
Red	2	2	2	4	19	21%	11%	15%	30%	6%	15%
Uni	0	0	0	17	266	6%	3%	5%	10%	1%	2%
Mod	0	0	0	14	0	0%	44%	100%	200%	43%	100%

Atributos

Elaboración 1

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	8	0%	0%	33%	67%	0%	0%
Exc	0	0	0	7	15	47%	23%	17%	33%	16%	100%
Over	0	0	0	0	15	0%	0%	30%	60%	0%	0%
Red	0	0	0	0	8	0%	0%	15%	30%	0%	0%
Uni	0	0	0	0	120	0%	0%	5%	10%	0%	0%
Mod	0	0	0	7	0	0%	23%	100%	200%	16%	100%

Elaboración 2

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	0	17	0%	0%	33%	67%	0%	0%
Exc	0	0	0	11	30	37%	18%	17%	33%	12%	51%
Over	2	2	2	3	30	10%	5%	30%	60%	6%	25%
Red	2.67	2	4	3	17	18%	9%	15%	30%	5%	22%
Uni	0	0	0	12	510	2%	1%	5%	10%	0%	1%
Mod	0	0	0	17	0	0%	32%	100%	200%	24%	100%

Elaboración 3

	Mean Order	Min Order	Max Order	Count	Max	Ratio	Adj Ratio	Norm Weight	Adj Weight	Weight Disc	Contribution
Def	0	0	0	4	53	8%	4%	33%	67%	5%	14%
Exc	0	0	0	11	42	26%	13%	17%	33%	9%	24%
Over	3.33	2	5	12	42	29%	14%	30%	60%	17%	47%
Red	2	2	2	10	53	19%	9%	15%	30%	6%	15%
Uni	0	0	0	44	2226	2%	1%	5%	10%	0%	1%
Mod	0	0	0	37	0	0%	41%	100%	200%	37%	100%

Apéndice G. Artículo: Measuring the Matching Between the Analyst and Designer Work: Characterization of the Representational Gap

Measuring the Matching Between the Analyst and Designer Work: Characterization ofthe Representational Gap

Omar Posada Villarreal
Centro de Investigación en Matemáticas (CIMAT)
Guanajuato, Gto, México
posada@cimat.mx

Cuauhtémoc Lemus Olalde
Centro de Investigación en Matemáticas (CIMAT)
Guanajuato, Gto, México
clemola@cimat.mx

Luis F. Fernández Martínez
Universidad Autónoma de Ciudad Juárez (UACJ).
Ciudad Juárez, Chih, México
lfernand@uacj.mx

Abstract

The need for more and better software applications demands improvement in software processes and products. As a result, there is an increasing demand for software metrics and better development approaches. This research focuses on characterizing (i.e. locating, identifying, and measuring) the representational gap, which is the interval between the mental model of the problem (e.g. domain model in the Unified Process) and the software representation of its solution (e.g. design model). These models were chosen because it is cheaper to identify defects in the early stages. The ontological discrepancies from an improved Bunge-Wand-Weber model, were used for characterizing this gap, and were represented as a bipartite digraph matrix. Also, the efficiency of the modelers was measured. Colloquially, this research detects how much the analyst work matches the designer work. Two cases of study were tested with an implementation of the proposed algorithm.

1. Introduction

The need of more and better software applications demands improvement in software processes (SPI) and products. As a result, there is an increasing demand of software metrics and better development approaches (as the leading object orientation) [2].

The key contribution of this paper is to prove how to evaluate the matching between the analyst and designer work. Its goal is to support the early detection of defects. Therefore, this research focuses on characterizing (i.e. locating, identifying, and measuring) the *representational gap*, which is "the gap (i.e. interval) between our mental model of the domain (e.g. domain model) and its representation in software (e.g. design model)" [9]. An example of this gap is shown in Figure 1. These models were chosen because identifying defections in the early stages is cheaper (i.e. reduces resources), since it prevents defect propagation in the current and later stages [7]. It is estimated that the time and cost of letting a defect pass to the next stage is three times more.

[9] thinks that it is "difficult to quantify" this gap, and that most of the software engineers know that a narrow gap is useful. On one hand, there is a huge gap if an application is directly programmed in binary code. On the other hand, there is a minimal gap when the matching between the domain vocabulary and the software vocabulary is one-to-one.

The ontological discrepancies from an improved Bunge-Wand-Weber model, were used for characterizing this gap, and were represented as a bipartite digraph matrix. This characterization lets one measure the efficiency of the modelers.

This document is organized as follows. The similar works, and main concepts are reviewed in section 2. The selection of models, and model elements is justified in section 3. The gap characterization is formed by the discrepancy localization (section 4), identification (section 5), and measuring (section 6). Two cases of study were tested with an implementation of the proposed algorithm in section 8. Some concluding remarks, and future work are presented in section 9 and section 10, respectively.

2. State of the art

2.1. Software metrics

Even the great impact of early stage metrics [11], there are less than for later stages (disciplines in the Unified Process terminology) [1]. The closest works [6], [8] to the present, are counting metrics, which do not consider any representational gap structure.

For differentiate some basic concepts, it will be listed their definition [4]:

- *Metric* is the function that assigns a number or symbol to an entity for characterize one or more attributes.
- *Measure* is the value of the metric.
- *Measurement* is the process where the metric is calculated through the attributes of the entities from the real world, applying clear, and defined rules.

This work applies the concepts of the representational theory of measurement, that are introduced in [4], and summarized in [12].

1.1. Unified Process

The Unified Process (UP) was chosen because it is a widely, mature, and standard methodology for object oriented software development, that strongly suggests the use of a *de facto* standard, the Unified Modeling Language (UML) [11].

Even the domain, and design model use the class diagram, they represent different entities: the domain model represents concepts from the real world, and the design model, software objects. It should be noted that, by definition, the domain model does not have methods.

1.2. Ontological discrepancies

The Bunge-Wand-Weber (BWW) model, inspired in the system theory, has been applied in the metrics suite for the object oriented development of Chidamber and Kemer [2], and in at least, ten works mentioned in [11].

The ontological discrepancies (from here denoted as discrepancies) of the BWW model may undermine the ontological clarity of modeling constructs and languages. These discrepancies are shown in Figure 2 as part of the improved model. Following, they are identified in [11]:

- *Construct deficit*. An ontological concept is not represented in the modeling constructs. Usually, it is a problematic situation (i.e. defect prone).
- *Construct excess*. A modeling construct does not represent any ontological concept. There are two types:
  - *Not problem-domain oriented*. Even the construct is not part of the domain; it could be useful, or necessary to the system development. It is not necessarily a problem.
  - *Genuine*. If the constructs are clearly intended to represent the problem domain but there is no corresponding concept, is genuine construct excess. This is a problematic situation.
- *Construct overload*. Several ontological concepts are represented by a modeling construct. Usually, it is a problematic situation (i.e. defect prone).
- *Construct redundancy*. An ontological concept is represented by several modeling constructs. There are two types:
  - *Subtype*. It might be useful when the modeling constructs represent disjunctive subtypes of the same ontological concept. It is not necessarily a problem.
  - *Genuine*. It indicates weakness in the definition, and in the structure of the modeling constructs. It is a problematic situation, because each group of constructs should be merged, if they are semantically similar, or split, if semantically different.

Therefore, unless the modeler, or the verifier justifies the discrepancy, it is a problematic situation.

2. Models and model elements selection

The domain, and design model were selected because in the UP methodology, they represent the mental model of the domain, and the software representation, respectively, of the representational gap [9]. Moreover, they are part of the early disciplines (i.e. stages), and both use the class diagram for their static representation.

The selected model elements are classes, packages, and attributes, in order of importance (supported by the cases of study). The selection criteria include the commonality of use [14], independent existence (i.e. it can exist without other elements), and the hierarchical composition that allows one chooses a level of detail. In this context, it is called hierarchical composition, the fact that a package could be composed by inner packages, or classes; and a class, by attributes. This is supported by the UML metamodel [10].

3. Locating discrepancies

A generalization of the Larman's representational gap [9] is proposed: instead of the "mental model of the domain", the term of *general model* (e.g. domain model) is suggested. In the same way, the *particular model* (e.g. design model) substitutes "its representation in software".

The systematic matching between the *general*, and *particular model elements* (e.g. conceptual classes, and design classes, respectively) is called location of discrepancies, and is done by the verifier.

The most suitable dependency is the refinement ("refine" stereotype), defined by [14] as "a relationship between two versions of a concept in different development stages, or in different abstraction levels. […] In principle, there is a matching from the less finished concept to the more finished concept". Then, the less finished concept corresponds to the general model

element, and the more finished concept, to the particular model element. In Figure 1, classes that participate in the matching between the domain, and design model, are located through refinements.

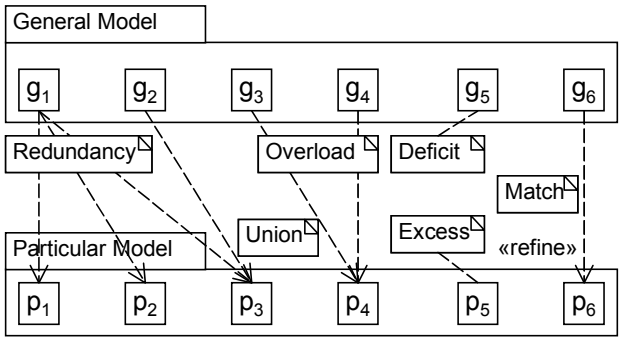


Figure 1. Representational gap between classes with the discrepancies types from the improved BWW model (class diagram)

The matching is registered into a bipartite digraph matrix, further called *refinement matrix*. The source vertices partition represent the set of general model elements (matrix rows), while the target vertices partition, the set of the particular model elements (matrix columns). The arcs represent the refinements, directed from the general to the particular model elements. This representation allows a formal definition of the problem, and a comprehensive theory [5].

Let G be the set of general model elements (e.g. conceptual classes); P, the set of particular model elements (e.g. design classes); n_X, the cardinality of the set X; the refinement matrix is $R \in \{0,1\}^{n_G \times n_P}$, where:

$$r_{i,j} = \begin{cases} 1, & g_i \text{ is refined in } p_j \\ 0, & \text{otherwise} \end{cases}$$

1. Identifying discrepancies

Some extensions to the BWW model were performed in order to support the characterization of the representational gap. These improvements are related with the discrepancy types, including their formal definition, that they are complimentary, the perception of the similarity between them, and the perception of the union discrepancy.

1.1. Formal definition of the type, and order of the discrepancies

The identification detects the type, and order of discrepancies from the refinement matrix.

The discrepancy order is the degree of the vertices (i.e. quantity of refinements in a model element); for the general model elements is the row sum, $r_{i\bullet} = \sum_{j=1}^{n_P} r_{i,j}$, and for the particular model elements is the column sum, $r_{\bullet j} = \sum_{i=1}^{n_G} r_{i,j}$, with $i = \{1,...,n_G\}$, and $j = \{1,...,n_P\}$. This order just applies for redundancy, and overload.

Four binary vectors e (*def*, *exc*, *over*, *red*, *uni*, and *1to1* states for deficit, excess, overload, redundancy, union, and one-to-one refinement) identify the discrepancy type:

$$e_{def,i} = \begin{cases} 1, & r_{i\bullet} < 1 \\ 0, & \text{otherwise} \end{cases}; \; e_{exc,j} = \begin{cases} 1, & r_{\bullet j} < 1 \\ 0, & \text{otherwise} \end{cases}$$

$$e_{over,j} = \begin{cases} 1, & r_{\bullet j} > 1 \\ 0, & \text{otherwise} \end{cases}; \; e_{red,i} = \begin{cases} 1, & r_{i\bullet} > 1 \\ 0, & \text{otherwise} \end{cases}.$$

And two vectors o identify the discrepancy order:

$$o_{over,j} = \begin{cases} r_{\bullet j}, & r_{\bullet j} > 1 \\ 0, & \text{otherwise} \end{cases}; \; o_{red,i} = \begin{cases} r_{i\bullet}, & r_{i\bullet} > 1 \\ 0, & \text{otherwise} \end{cases}$$

1.2. Discrepancy matrix

Table 1. Type, and order in the discrepancy matrix

R	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	Def
g[1]	R3	R3	O2R3	-	-	-	-
g[2]	-	-	O2	-	-	-	-
g[3]	-	-	-	O2	-	-	-
g[4]	-	-	-	O2	-	-	-
g[5]	-	-	-	-	-	-	D
g[6]	-	-	-	-	-	1to1	-
Exc	-	-	-	-	E	-	

The discrepancy matrix is composed by $(n_G+1)\times(n_P+1)$ label (i.e. string) elements. The identification of Figure 1 is shown in Table 1. The overload, redundancy, and one-to-one refinements are formed by the concatenation (represented by the plus sign) of the type, and order where an arc exists. The discrepancy union is the join in a model element of an overload, and a redundancy (e.g. "O2R3" in Table 1):

$$s_{i,j} = \begin{cases} ("O"+o_{over,j}) \\ +("R"+o_{red,i}), & e_{red,i} = 1 \vee e_{over,j} = 1, r_{i,j} = 1 \\ "1to1", & e_{red,i} = e_{over,j} = 0, r_{i,j} = 1 \\ \varnothing, & \text{otherwise} \end{cases}$$

While the deficit, and excess refinements are composed by:

$$s_{i,n_P+1} = \begin{cases} \text{"D"}, & e_{def} = 1 \\ \varnothing, & \text{otherwise} \end{cases}$$

$$s_{n_G+1,j} = \begin{cases} \text{"E"}, & e_{exc} = 1 \\ \varnothing, & \text{otherwise} \end{cases}$$

1.1. Discrepancy categories

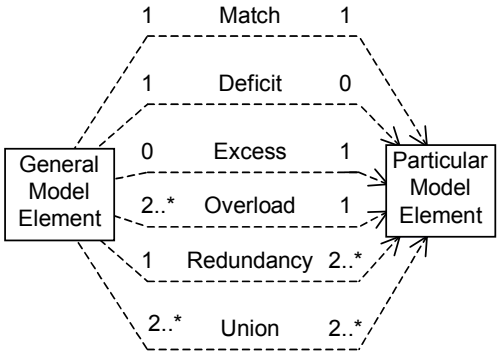


Figure 2. Gap relationships between the general, and particular model elements

Inspired by the BWW ontological discrepancies, the *refinement discrepancies* between the general model element, and the particular model element, define the representational gap. The model element match (i.e. one-to-one), and the proposed discrepancy types are shown in Figure 2 (note that the union discrepancy is not considered in the BWW model, see 2.3).

Several discrepancy characteristics were found. Discrepancies are complimentary; this means that if one adds (or deletes) a refinement, some discrepancies are formed (or removed), assuming that the number of model elements remains fixed.

Table 2. Discrepancy categories from the improved BWW model

Discrepancy types		**Is there a representation in the counterpart model?**	
		Yes	No
More elements in the:	***Discrepancy category***	*Represented*	*No represented*
Domain model	*Abundant in the general*	Overload	Deficit
Design model	*Abundant in the particular*	Redundancy	Excess

A discrepancy categorization is proposed in Table 2, due to the formal definition of the discrepancies, and the description of the ontological discrepancies from the BWW model [11]. The discrepancy categories are: represented, no represented, abundant in the general (usually problematic), and abundant in the particular (not necessarily problematic).

2. Measuring discrepancies

After studying, and experimenting with several measuring methods, the *Weighted Count* method was chosen because it is intuitive, simple, flexible, and it lets one separately measure each discrepancy type. The other methods were the Square Chi Test of Independence [3], [13], [15], Simple Count, Flow [5], and certain heuristics.

Briefly, the Weighted Count method obtains the discrepancy quantity by type, and order; then divide them by the maximum possible number of discrepancies; after that it ponders the importance of each discrepancy type with a given weight vector.

2.1. Weight assignation

For expressing that the discrepancies are complimentary, and to normalize the weighted metrics between [0, 1], it must fulfill (1),

$$w_{def} + w_{exc} = w_{over} + w_{red} + w_{uni} = 1, \quad (1)$$

where $W = \{w_k\}$, with $k = \{def, exc, over, red, uni\}$.

The verifier assigns the weights, and may consider the application requirements, domain, experience, etc. Some rules are suggested:

- If one wants to add in the metric that discrepancies with abundance in the general (i.e. deficit, overload), are more problematic than those with abundance in the particular (i.e. excess, redundancy), as affirmed by [11], then it must fulfill: $w_{def} > w_{exc}$, $w_{over} > w_{red}$. Particularly, it is suggested (2) with $\alpha = 2$:

$$w_{def} = \alpha \cdot w_{exc}, \; w_{over} = \alpha \cdot w_{red} \quad (2)$$

- If one wants to add in the metric that the union discrepancy acts as a discriminant (penalization due to high order discrepancies), then its weight should be the lightest (4). Particularly, it is also suggested that it represents a low proportion (e.g. $\rho_{uni} = 10\%$) as shown in (5):

$$w_{uni} < \min(w_k) \quad (4)$$

$$\frac{w_{uni}}{w_{over} + w_{red} + w_{uni}} = \rho_{uni} \approx 0.1 \quad (5)$$

The weight vector used here, which fulfill (1), (2), (3), (4), and (5) is:

$$W = (w_{def}, w_{exc}, w_{over}, w_{red}, w_{uni}) = (2/3, 1/3, 0.6, 0.3, 0.1) \quad (6)$$

1.1. Proposed metrics

Table 3. Metric descriptions

Column	Description	Scale	Direct, or indirect
Type	Type of the discrepancy	Nominal	Indirect
Minimum, and maximum order*	Statistical information about the set of orders	Interval	Direct
Mean order*		Interval	Indirect
Quantity	Number of discrepancies in the model	Absolute	Direct
Maximum	Greatest possible number of discrepancies fixing the number of model elements	Ratio	Indirect
Relative	Fraction from dividing the quantity by the maximum	Ratio	Indirect
Discrepancy weight	Product of the relative metrics by the weight vector	Ratio	Indirect
Contribution*	Fraction of the model metric	Ratio	Indirect

Table 4. Gap metrics example

Type	Qty.	Maximum	Relative	Discrepancy weight
Def	1	6	17%	11%
Exc	1	6	17%	6%
Over	2	6	33%	20%
Red	1	6	17%	5%
Uni	6	36	17%	2%
Model	**5**		**42%**	**43%**

Table 3 summarizes the proposed metrics. The columns marked with an asterisk are not shown in the next tables because they are informative, or secondary. The measurement scale, and the direct/indirect measurement are explained in [4]. Table 4 shows the gap metrics of Figure 1. Two logical adjusts were needed: the model quantity excludes the union quantity from the sum (because the union is derived from the *represented discrepancies*), and the relative model metric (because discrepancies are complimentary, it is half of the sum) excludes the union relative metric.

1.2. Efficiency of the modelers

Table 5. Variable description for the efficiency of the modelers metric

Variable	Description	Table type
I	Initial metrics set	Input
J	Justified (corrected) metrics set	Input
E	Eliminated metrics set	Input
ε_{abs}	Absolute error	Output
ε	Relative error	Output
Effi	Efficiency of the modelers	Output

Initial discrepancies are composed by justified discrepancies, and those to eliminate; this is represented by $I = J + E$. Because the discrepancies to eliminate are not desired, they constitute the absolute error, $\varepsilon_{abs} = E = I - J$. The relative error is $\varepsilon = \frac{\varepsilon_{abs}}{I} = 1 - \frac{J}{I}$, defined as $\varepsilon = 0$, when $I = 0$ (avoidance of the zero division), because it is expected that $J \le I$. Then, the efficiency of the modelers is $Effi = 1 - \varepsilon$. The description of these variables is given in Table 5.

Table 6. Metrics of the model

Model	Quantity	Ratio	Discrepancy weight
Initial	5	42%	43%
Justified	3	20%	17%
Eliminated	2	22%	27%
Relative error	**40%**	**53%**	**62%**
Efficiency	**60%**	**47%**	**38%**

The proposed metrics from comparing the initial, and the corrected (i.e. justified) model of Figure 1 are shown in Table 6. These metrics express the narrowing in the representational gap from 43% to 17% after eliminating the 40% of the discrepancies, which represented the 62% of the initial metric of the discrepancy weight. The

efficiency of the modelers was 60%. Therefore, they are not very efficient (40% of error).

The sequence of required activities is shown in Figure 3. The analyst defines the domain model that is processed by the designer to conceive a specific solution in a design model. Instead of giving the design model to the programmer (with hidden defects from the analysis, and design), both models are reviewed by the verifier with the gap characterization. Then the modelers are feedback with the discrepancy matrix, and the initial gap metric tables. After they correct their models, another characterization is done to compare their products measuring the efficiency of the modelers for management purposes.

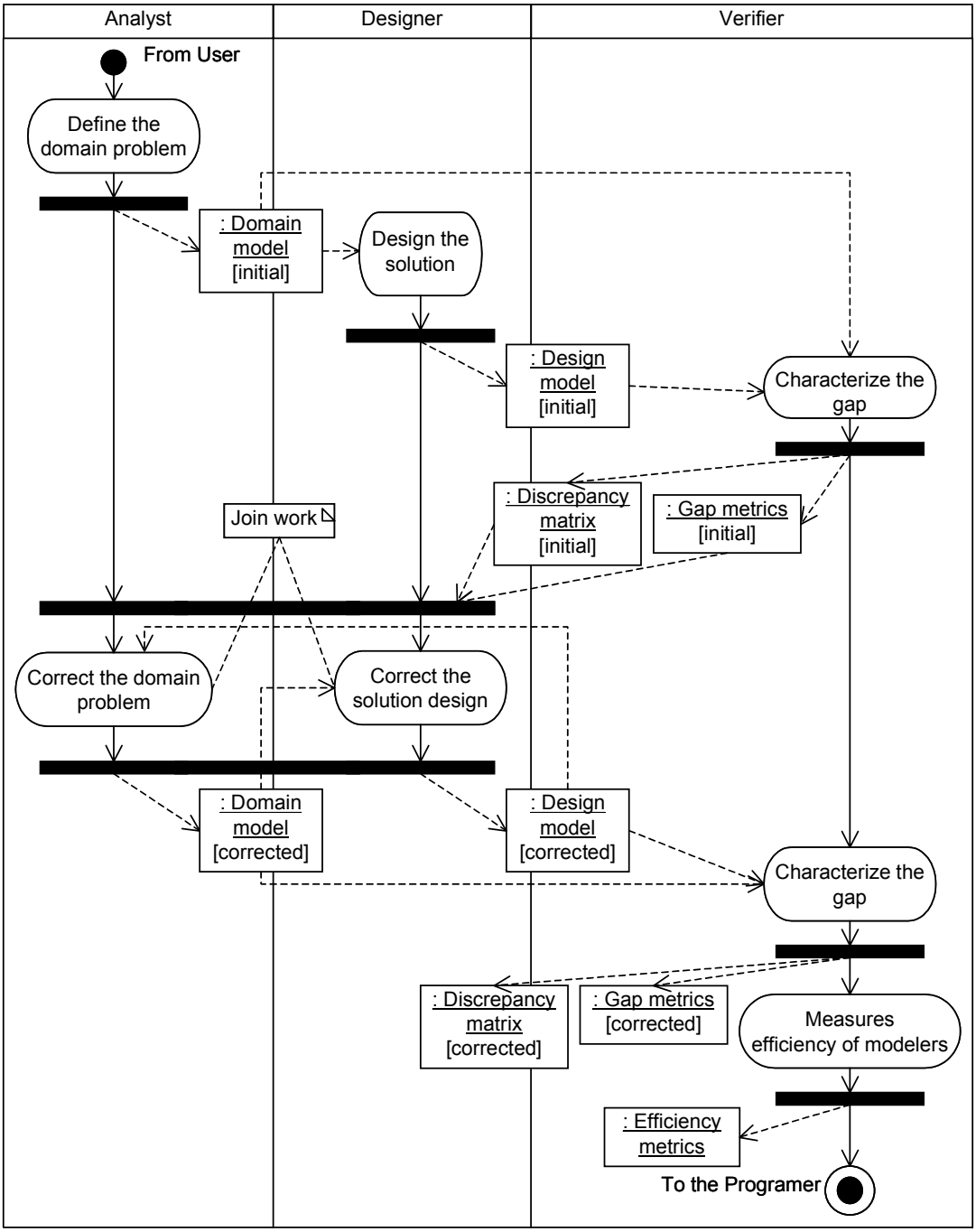


Figure 3. Proposed activities, and products (objects) to characterize the gap, and measure the efficiency of the modelers (activity diagram)

1. Cases of study

For space consideration, it has been shown the characterization for a small but illustrative example that characterizes the gap between classes. While the cases of study characterize it between the selected model elements: classes, packages, and attributes. These cases used the weights of (6) for the domain, and design model that belong to the domain modeling, and design disciplines, respectively. Several lessons were learned for each model element (even not listed here for space).

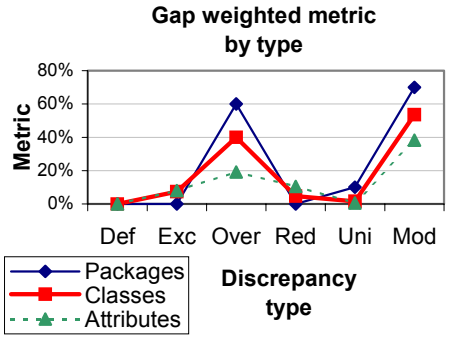


Figure 4. Weighted metric for the gap by model element, and discrepancy type of the first case of study (Linear Algebra)

The goal of the first case of study was to compare the domain model with the design model extracted from source code with a reverse engineering tool. This code was part of an academic project, a Linear Algebra library, written in Java. This project was not altered, or designed in any CASE tool.

Figure 4 shows certain similarity between the weighted metrics of the selected model elements. Particularly, the overload discrepancy has the highest contribution to the metric of the model.

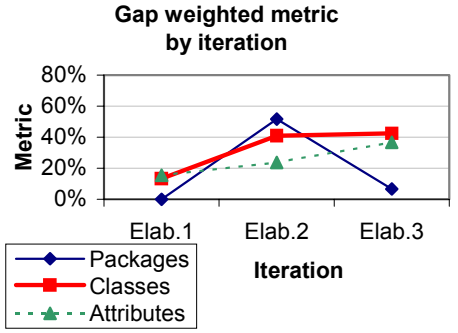


Figure 5. Weighted metric for the gap by iteration of the second case of study (Scholar)

The purpose of the second case of study was to study the behavior through three UP iterations of a scholar application. For each iteration, it was documented the use cases, domain and design model, and observations about the matching between the selected model elements, and the system development.

In Figure 5, there is an increasing trend as iterations pass, except for the weighted metric for the packages in Elaboration 3. Even the system was growing in these iterations, the gap between classes was stabilized. There is a minimal gap between packages in Elaboration 1 because there was just one package in each model.

2. Proposed representational gap definition

After studying, and characterizing the representational gap in two cases of study, a generalized, and more formal definition than [9] is proposed: *"The representational gap is the distance between a general (e.g. domain model) and a particular (e.g. design model) model of the (software) system development stages (i.e. disciplines), that considers the set of refinement discrepancies obtained through a bipartite digraph, where the vertices represent the model elements of the same type (e.g. classes, packages, attributes), and the arcs, the refinements between the general, and particular model elements"*.

3. Conclusions

This paper summarizes a thesis research work that focused on characterizing the representational gap between the software development stages (i.e. disciplines). The refinement discrepancies (i.e. ontological discrepancies) of an improved BWW model throughout representation of a bipartite digraph were the theorical basis of an implemented algorithm that characterized the gap. The goal is resource reduction, and better quality, through defect identification in the early stages for preventing the defect propagation of other elements of the current, and following stages.

Particularly, the representational gap was located, identified, and measured, between the classes, packages, and attributes (i.e. selected model elements) from the domain, and design model (i.e. selected models) that belong to the UP domain modeling, and design disciplines, respectively. Also, the efficiency of the modelers was measured, applying the error formula to the initial and corrected gap metrics table. An historic record of these corrections provides insight on how much the analyst work matches the designer work in a formal, and quantifiable way.

Even [9] states the phenomena of the representational gap, and considers it to be “difficult to quantify”; he does not formally define, or characterize it. Therefore, a formal definition for characterizing the representational gap is proposed. Oppositely to Larman [9] that seeks to narrow the gap, here it is considered that it is better to understand, and control it in a systematic way. This characterization properly takes into account this consideration. Everything is done for preventing, and correcting defect in the early stages of system development.

1. Future work

Because it is an initial understanding on the representational gap, further work it is clearly obvious. The current research lines include the prototype development in a CASE tool for UML; the application between other pair of disciplines (e.g. design, and implementation model), and other model elements as associations, heritage relationships, methods, etc.; study of the behavior of the metrics evolution; and the design of proper patterns for the discrepancy control.

2. Acknowledgment

Helpful comments were received on earlier drafts from G. Padilla. This research was supported by the CONACYT, and CONCYTEG grant.

3. References

[1] Abreu, F.B. et al, "W20 ECOOP Workshop on Quantitative Approaches in OO Software Engineering", 1999.
[2] Chidamber, S.R. and Kemerer, C.F., "A Metrics Suite for Object Oriented Design", 1994.
[3] Cochran, W.G., "The Square Chi Test of Independence of Fit", Annals of Mathematical Statistics, 1952.
[4] Fenton, N.E. and Pfleeger, S.L., Software Metrics: A rigourous & practical approach, ITP Thompson, 1997.
[5] Gross, J. and Yellen, J., Graph Theory and its applications, CRC Press, 1998.
[6] Hogan, J., "An Analysis of OO Software Metrics", 1997.
[7] Humphrey, W.S., A discipline for software engineering, Addison-Wesley, 1995.
[8] Kim, H. and Boldyreff, C., "Developing Software Metrics Applicable to UML Models", 2002.
[9] Larman, C., Applying UML and patterns, 2001.
[10] Object Management Group, OMG Unified Modeling Language Specification, 2003.
[11] Opdahl, A.L. and Henderson-Sellers, B., "Ontological evaluation of the UML using the Bunge-Wand-Weber model", Springer Software and Systems Modelling (SoSyM) 1(1), 2002.
[12] Purao, S. and Vaishnavi, V., "Product metrics for Object-Oriented Systems", ACM Computing Surveys, 2003.
[13] Rice, J.A., Mathematical Statistics and Data Analysis, Duxbury Press, 1995.
[14] Rumbaugh, J., Jacobson, I. and Booch, G., The Unified Modeling Language reference manual, Addison-Wesley, 1999.
[15] Sheskin, D.J., Handbook of parametric and nonparametric statistical procedures, Chapman & Hall, 2003.

Bibliografía

Referencias citadas en el documento

[Abreu99]	Abreu, F.B. et al. "W20 ECOOP Workshop on Quantitative Approaches in OO Software Engineering". 1999
[Behzad81]	Behzad, M., Chartrand, G. and Lesniak-Foster, L. Graphs and digraphs. Wadsworth International Group. 1981.
[Bogart83]	Bogart, K.P. Introductory combinatorics. Pitman. 1983.
[Booch99]	Booch, G., Rumbaugh, J. and Jacobson, I. The Unified Modeling Language user guide. Addison Wesley. 1999.
[Chidamber94]	Chidamber, S.R. and Kemerer, C.F. "A Metrics Suite for Object Oriented Design". 1994.
[Cochran52]	Cochran, W.G. "The Square Chi Test of Independence of Fit". Annals of Mathematical Statistics. 1952.
[Dumke95]	Dumke, R.R. and Foltin, E. "Evaluated Object-Oriented Software Development". 1995.
[Fenton97]	Fenton, N.E. and Pfleeger, S.L. Software Metrics: A rigourous & practical approach. ITP Thompson. 1997.
[Gross98]	Gross, J. and Yellen, J. Graph Theory and its applications. CRC Press. 1998.
[Hogan97]	Hogan, J. "An Analysis of OO Software Metrics". 1997.
[Howe03]	Howe, D. "The Free On-line Dictionary of Computing". 2003.
[Humphrey95]	Humphrey, W.S. A discipline for software engineering. Addison-Wesley. 1995.
[Kim02]	Kim, H. and Boldyreff, C. "Developing Software Metrics Applicable to UML Models". 2002.
[Krueger92]	Krueger, C.W. "Software reuse". ACM Computing Surveys. 24(2) pp. 131-183. 1992.
[Larman01]	Larman, C. Applying UML and patterns. 2001.
[Mundinteractivos04]	Mundinteractivos. "Diccionarios". 2004.
[OMG03a]	Object Management Group. OMG Unified Modeling Language Specification. 2003.
[Opdahl02]	Opdahl, A.L. and Henderson-Sellers, B. "Ontological evaluation of the UML using the Bunge-Wand-Weber model". Software and Systems Modelling (SoSyM) 1(1). Springer. 2002.
[Pamplona02]	Universidad de Pamplona. "Eficiencia, eficacia y efectividad". Universidad de Pamplona. Control Interno. Boletín Informativo Nº 3 - Octubre de 2002. 2002.
[Pressman01]	Pressman, R.S. Software Engineering: A Practitioner´s Approach. 2001.
[Purao03]	Purao, S. and Vaishnavi, V. "Product metrics for Object-Oriented Systems". ACM Computing Surveys. Vol. 35, No. 2, June 2003, pp. 191-221. 2003.
[Rice95]	Rice, J.A. Mathematical Statistics and Data Analysis. Duxbury Press. 1995.
[Rumbaugh99]	Rumbaugh, J., Jacobson, I. and Booch, G. The Unified Modeling Language reference manual. Addison-Wesley. 1999.
[Sheskin03]	Sheskin, D.J. Handbook of parametric and nonparametric statistical procedures. Chapman & Hall. 2003.
[Siviy01]	Siviy, J. "Six Sigma". Software Engineering Institute. 2001.
[Sommerville96]	Sommerville, I. Software engineering. International Computer Sciences Series. 1996.
[Tayntor03]	Tayntor, B.C. Six Sigma Software Development. CRC Press LLC. 2003.

* 27 documentos

Bibliografía consultada pero no citada

[Abreu94] Abreu, F.B. and Carapuça, R. "Candidate Metrics for Object-Oriented Software within a Taxonomy Framework". 1994.

[Abreu96] Abreu, F.B., Esteves, R. and Goulão, M. "The Design of Eiffel Programs: Quantitative Evaluation Using the MOOD Metrics". 1996.

[Bala99] Bala, P. "Applying some object oriented metrics in an university software engineering lab". 1999.

[Basili95] Basili, V.R., Briand, L. and Melo, W.L. "A validation of object-oriented design metrics as quality indicators". 1995.

[Boehm01] Boehm, B. and Basili, V.R. "Software Defect Reduction Top 10 List". 2001.

[Carter93] Carter, B.D. et al. "A Pilot Software Maintainability Study Of A Scientific Program Library". 1993.

[Fairley85] Fairley, R.E. "Software engineering concepts". McGraw-Hill. 1985.

[Fenton99] Fenton, N.E. "A Critique of Software Defect Prediction Models". 1999.

[France01] France, R. and Bieman, J.M. "Multi-View Software Evolution: A UML-based Framework for Evolving Object-Oriented Software". Proceedings International Conference on Software Maintenance (ICSM 2001). 2001.

[Frappier94] Frappier, M., Matwin, S. and Mili, A. "Software Metrics for Predicting Maintainability Software Metrics Study: Technical Memorandum 2". 1994.

[Guizzardi02] Guizzardi, G., Herre, H. and Wagner, G. "Towards Ontological Foundations for UML Conceptual Models". In Proceedings of 1st Int. Conf. on Ontologies, Databases and Applications of Semantics (ODBASE 2002), Springer. 2002.

[Kan95] Kan, S.H. Metrics and models in software quality engineering. 1995.

[Li93] Li, W. and Henry, S.M. "Object-Oriented Metrics Which Predict Maintainability". 1993.

[Menard03] Menard, R. "Domain Modeling: Leveraging the heart of RUP for straight through processing". 2003.

[Monden01] Monden, A., Sato, S. and Matsumoto, K. "Capturing Industrial Experiences of Software Maintenance Using Product Metrics". 2001.

[Muller97] Muller, P.A. Modelado de objetos con UML. Enrolles. 1997.

[OMG02] Object Management Group. "Software Process Engineering Metamodel Specification". Object Management Group, Inc. 2002.

[OMG03b] Object Management Group. "UML 2.0 Superstructure Specification". 2003.

[Pritchett IV01] Pritchett IV, W.W. "An Object-Oriented Metrics Suite for Ada 95". 2001.

[Putnam02] Putnam, D. "How much software can be developed in a year?". 2002.

[Rational] Rational Software Corporation. "Rational ProjectConsole Database Sizing Estimation".

[Rational02] Rational Software Corporation and École Polytechnique de Montréal. "Guidelines: Metrics". 2002.

[Rational03] Rational Software Corporation. "Rational RequisitePro Metrics FAQs". 2003.

[Royce00] Royce, W. "Next-Generation Software Economics". 2000.

[Rubinstein01] Rubinstein, D. "Rational Adds ProjectConsole to latest Suite". 2001.

[Rufai03] Rufai, R.A. "New structural similarity metrics for UML models". 2003.

[Sedigh-Ali] Sedigh-Ali, S. and Ghafoor, A. "Metrics-Guided Quality Management for Component-Based Software Systems".

[SMLab] SMLab. "SMLab's Bibliography. Object Oriented Design Measures".

[SMLab] SMLab. "Software Metrics Classification".

[Stavrinoudis99] Stavrinoudis, D., Xenos, M. and Christodoulakis, D. "Relation Between Software Metrics And Maintainability". 1999.

[Street__] Street, S.W.I. "Categorical Data and Contingency Tables".

[Verkamo01] Verkamo, A.I. et al. "Measuring design diagrams for product quality evaluation". 2001.

[Wand93] Wand, Y. and Weber, R. "On the ontological expressiveness of information systems analysis and design grammars". Journal of Information Systems. 1993.

[White98] White, S.A. and Lemus Olalde, C. "Architectural Reuse in Software Development". 1998.

[Wiegers97] Wiegers, K.E. "Software Metrics: Ten Traps to Avoid". 1997.

[Wu] Wu, H.C. "Practical Metrics using Rational ProjectConsole".

[Wyrick01] Wyrick, P. and Ishigaki, D. "Unified Software Project Management with Rational ProjectConsole". 2001.

[Xenos00] Xenos, M. et al. "Object-Oriented Metrics A Survey". 2000.

[Zuse03] Zuse, H. "Ten Theses What can Practitioners learn from Measurement Theory?". 2003.

* 39 documentos