SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Secretaría Académica, de Investigación e Innovación
Dirección de Posgrado, Investigación e Innovación

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Centro Nacional de Investigación y Desarrollo Tecnológico

Subdirección Académica

Departamento de Ciencias Computacionales

TESIS DE MAESTRÍA EN CIENCIAS

Metodología de Validación de Consistencia de Patrones de Análisis

presentada por
Ing. José Domingo Juárez Hernández

como requisito para la obtención del grado de
Maestro en Ciencias de la Computación

Director de tesis
Dr. Moisés González García

Co-Director de tesis
Dr. René Santaolaya Salgado

Cuernavaca, Morelos, México. Julio de 2016.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 22 de junio del 2016
OFICIO No. DCC/159/2016

Asunto: Aceptación de documento de tesis

C. DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. José Domingo Juárez Hernández**, con número de control M14CE012, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado **"Metodología de validación de consistencia de patrones de análisis"** y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Moisés González García
Doctor en Ciencias en la Especialidad de Ingeniería Eléctrica
7501724

CO-DIRECTOR DE TESIS

Dr. René Santaolaya Salgado
Doctor en Ciencias de la Computación
4454821

REVISOR 1

Dra. Olivia Graciela Fragoso Díaz
Doctora en Ciencias en Ciencias de la Computación
7420199

REVISOR 2

Dr. Joaquín Pérez Ortega
Doctor en Ciencias Computacionales
4795984

REVISOR 3

Dra. Alicia Martínez Rebollar
Doctora en Informática
7399055

C.p. Lic. Guadalupe Garrido Rivera - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

AMR/lmz

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 23 de junio de 2016
OFICIO No. SAC/236/2016

Asunto: Autorización de impresión de tesis

ING. JOSÉ DOMINGO JUÁREZ HERNÁNDEZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Metodología de validación de consistencia de patrones de análisis"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

ATENTAMENTE
"CONOCIMIENTO Y TECNOLOGÍA AL SERVICIO DE MÉXICO"

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

SEP TecNM
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

C.p. Lic. Guadalupe Garrido Rivera.- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

Dedicatoria

A mis padres:

Antonio Juárez Marroquín y América Hernández López

Por el invaluable esfuerzo y sacrificio para brindarme la educación valores e ideales en mi persona, además de motivar mi formación profesional, agradezco su incondicional apoyo en todas las fases de mi vida y por ser partícipes de mis anhelos y proyectos.

Agradecimientos

Gracias al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado en el desarrollo de esta Tesis.

Al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) por haberme permitido realizar mis estudios de maestría en ciencias computacionales.

A mis directores de tesis: Dr. Moisés González García y Dr. René Santaolaya Salgado por haberme guiado durante el desarrollo de este trabajo de investigación.

A mis asesores y revisores la Dra. Alicia Martínez Rebollar, Dra. Olivia Graciela Fragoso Díaz, Dr. Joaquín Pérez Ortega a lo largo de este trabajo de investigación.

A mis amigos en la maestría a Pedro, Sandro, Joel, Vania, Roberto, Jorge, Salvador, Rita, Félix, Bismark, Alida.

GRACIAS

Resumen

La comunidad de ingeniería de software dedica esfuerzos importantes a proponer soluciones a problemas recurrentes en todos los niveles de desarrollo de software. En consecuencia, ha surgido la noción de patrones de software, entre los cuales se encuentran los patrones de análisis. Estos patrones constituyen un tópico de interés por parte de los ingenieros de software, independientemente de su sub-área de investigación. Su uso como herramienta durante la fase de análisis en el desarrollo de un producto de software, ayuda a entender el problema mirando más allá de los requisitos "superficiales". Sin embargo, hoy en día los patrones de análisis se especifican mediante plantillas, las cuales incluyen descripciones textuales y visuales usando lenguaje natural y diagramas no estandarizados, que dificultan su manejo.

Por esta razón en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) se desarrolló la "Metodología para la Construcción de Ontologías de Patrones de Análisis" (MECOPA), la cual detalla los procesos necesarios que deben realizarse con los patrones de análisis para refinar la información del modelo de dominio y representarlo mediante una ontología. Pero MECOPA no abarca actualmente forma la validación de consistencia de los patrones de análisis. Por tal motivo, en este trabajo de investigación se desarrolló la "Metodología de validación de consistencia de patrones de análisis" (MEVASE) la cual se incluye como una nueva fase en MECOPA, permitiendo validar la descripción de patrones de análisis resultantes.

En vista que hay una gran variedad de patrones de análisis y las ventajas de usar versiones consistentes de los patrones de análisis (en estructura y relaciones entre conceptos), el resultado de esta investigación impulsará su uso en el modelado de negocios y permitirá aprovechar su potencial completamente.

Abstract

The software engineering community devotes significant efforts to propose solutions to recurring problems at all levels of software development. Accordingly, there has emerged the notion of software patterns, among which are analysis patterns. These patterns are a topic of interest among software engineers, regardless of the research area. Its use as a tool for the analysis phase in the development of a software product helps to understand the problem under study beyond the "surface" requirements. However, nowadays analysis patterns are specified through templates, which include visual and textual descriptions using natural language and not standardized diagrams that make difficult its handling.

For this reason at the National Center for Research and Technological Development (CENIDET) it was developed the "Methodology for ontology Building of analysis patterns" (MECOPA) which details the processes needed which must be performed with analysis patterns to refine the information domain model and represent it by an ontology. But currently MECOPA it not included validating of consistency the analysis patterns. Therefore, in this research was developed the "Methodology validation of consistency the analysis patterns" (MEVASE) which is included as a new phase in MECOPA, allowing validate the description of analysis patterns resulting.

Given that there are a variety of patterns of analysis and the advantages to use consistent versions of analysis patterns (in structure and relationships between concepts), the result of this research will promote its use in business modeling and will exploit its potential completely.

Contenido

Página

Lista de Figuras

Lista de Tablas

Capítulo 1. Introducción

En este capítulo se describe el problema a abordar en esta tesis. Mostrando también objetivo, alcances, limitaciones, justificación, beneficios y la organización general de este documento.

1.1 Introducción

El desarrollo de software es un proceso que consume tiempo y costo, pero no garantiza la calidad del producto. Actualmente, los ingenieros de software se han dado cuenta de que una parte considerable de recursos se utilizan en la solución de problemas que son similares o idénticos a otros que ya fueron resueltos en proyectos anteriores. Por esta razón la comunidad de ingeniería de software dedica esfuerzos importantes a proponer soluciones a problemas recurrentes en todos los niveles de desarrollo de software. En consecuencia, ha surgido la noción de patrones de software [1].

Un patrón es una idea que ha sido útil en un contexto y probablemente lo sea en otros. Es un modo de proveer información en forma de una declaración de problema, algunas restricciones, o una presentación de una solución ampliamente aceptada al problema. También puede incluir una discusión de las consecuencias de esa solución. Los patrones ayudan a disminuir la complejidad del software en varias fases en el ciclo de vida. Proponen ideas en diferentes niveles de abstracción, así como soluciones obtenidas por experiencia y las hacen accesible para los desarrolladores, analistas e ingenieros menos expertos o principiantes.

Las etapas de análisis y diseño son fundamentales en el desarrollo de software. En la fase de análisis se estudia y especifica el dominio del problema dentro del contexto de objetivos y metas preestablecidas en el negocio, buscando comprender los detalles y complejidades concernientes al problema para ir hacia un modelo mental de lo que se pretende alcanzar. A menudo encontramos que muchos aspectos de un problema en un dominio han aparecido en otros proyectos. Abstraer el modelo conceptual y aplicarlos al problema en cuestión es el desafío. De esto tratan los patrones de análisis.

Actualmente los patrones de análisis se especifican mediante plantillas, las cuales incluyen descripciones textuales y visuales usando lenguaje natural y diagramas no estandarizados, que dificultan su manejo. Por esta razón no son incluidos en la fase del modelado del negocio. Sin embargo, en [3] se propone una plantilla de patrones de análisis con el propósito de unificar otros enfoques existentes. A la fecha parece que ha dado resultado ya que está emergiendo como una de las más utilizadas. Aunque no incluye la semántica en la descripción de los patrones.

El propósito de esta investigación es llevar a cabo la descripción y validación de patrones de análisis, utilizando una formalización adecuada y ontologías. Que la representación formal elegida se pueda incluir como parte de la ontología de los patrones de análisis. Para realizarlo se propone la descripción de los patrones de análisis mediante la plantillas estable [3] y la construcción de las ontologías, que incluyan la semántica de los patrones de análisis, esto para facilitar la creación de patrones compuestos en trabajos futuros.

1.2 Descripción del problema

Una limitación en los patrones de análisis actuales es la falta de información semántica en su descripción, es decir, hay elementos definidos de forma ambigua y las relaciones entre ellos no están bien definidas. Esta deficiencia en la definición de la semántica dificulta la validación de patrones de análisis.

1.3 Objetivos

1.3.1 Objetivo general

- Crear y validar patrones de análisis utilizando una plantilla estable para describirlos, nomenclatura formal y ontologías en el proceso de validación.

1.3.2 Objetivos específicos

- Identificar una notación formal adecuada, para validación de patrones de análisis y para su inclusión en algún lenguaje para ontologías.
- Desarrollar la ontología para un patrón de análisis descrito, incluyendo nomenclatura formal.
- Proponer un método de validación de patrones de análisis.
- Validar uso de ontologías en la validación de patrones.

1.4 Justificación

En la fase de análisis se elaboran modelos conceptuales para comprender y comunicar los conocimientos básicos del problema de estudio. Los patrones que representan los modelos conceptuales son llamados patrones análisis. Los patrones de análisis pueden ser usados para entender un problema similar y relacionado. Los problemas de la vida real pueden requerir la integración de varios patrones análisis [4].

Actualmente no existe herramienta que permitan realizar la validación en patrones de análisis. Por tal motivo, el realizar la validación de patrones de análisis, es un área de oportunidad para mejorar la descripción del modelado de problemas en diferentes contextos. En vista que hay una gran variedad de patrones de análisis que podría ser validado, el resultado de esta investigación impulsará su uso en el modelado de negocios y permitirá aprovechar su potencial completamente.

1.5 Beneficios

Los beneficios que se obtendrán al realizar esta investigación serán:

- Proveer al diseñador de ontologías, una metodología para la construcción de ontologías de patrones de análisis.
- Construir ontologías para patrones de análisis, así como su representación computacional.
- Un conjunto de reglas que permitan conocer la manera correcta de combinar patrones de análisis.
- La validación de reglas de combinación de un patrón compuesto.
- Un trabajo que pueda fungir como base para proyectos de investigación futuros.

1.6 Organización del documento de tesis

La estructura restante de este documento se compone de cinco capítulos más, los cuales se describen a continuación.

Capítulo 2. Marco teórico. En este capítulo se presenta para una mejor compresión de la lectura conceptos empleados en esta tesis como lo son: patrones de análisis, modelo de estabilidad de software, metodología para la construcción de ontologías (METHONTOLOGY), entre otros.

Capítulo 3. Estado del Arte. En este capítulo se presentan los resultados del proceso de búsqueda de información relevante con respecto a trabajos de investigación relacionados que sirven como punto de referencia y comparación con el trabajo propuesto.

Capítulo 4. Metodología de Validación de Consistencia de Patrones de Análisis. En este capítulo se hace una adaptación de la Metodología para la Construcción de Ontologías de Patrones de Análisis (MECOPA) para la creación de ontologías de patrones de análisis. Se explica el proceso de desarrollo y adaptación de la "Metodología de Validación de Consistencia de Patrones de Análisis (MEVASE)", la cual permite mejorar la descripción en patrones de análisis desarrollados a través de MECOPA.

Capítulo 5. Construcción y Validación de la Ontología del Patrón de Análisis de Cuenta. En este capítulo se presenta la documentación y los pasos que se llevaron a cabo para la construcción de la ontología del patrón de análisis para cuenta a través de la Metodología para la Construcción de Ontologías de Patrones de Análisis (MECOPA) la cual incluye la nueva fase de validación para la creación de ontologías de patrones de análisis.

Capítulo 6. Conclusiones y Trabajos Futuros. En este capítulo se presentan las conclusiones obtenidas a través del proceso de investigación desarrollado para llevar a cabo este trabajo de tesis y los que se podrían considerar como trabajos futuros.

Capítulo 2. Marco Teórico

En este capítulo se presenta para una mejor compresión de la lectura conceptos empleados en esta tesis como lo son: patrones de análisis, modelo de estabilidad de software, metodología para la construcción de ontologías (METHONTOLOGY), entre otros. Así como los lenguajes, GRL y UCM, para la especificación de los requerimientos utilizados en esta tesis.

2.1 Patrones de análisis

Martin Fowler introdujo este término para describir soluciones relacionadas con problemas que aparecen durante el análisis de requisitos y las fases de modelado conceptual de los datos. Para él "un patrón es una idea que fue útil en un contexto práctico y probablemente será útil en otros" [6]. La semántica de los patrones de análisis describe aspectos específicos de algún dominio de aplicación en concreto, permitiendo definir un conjunto de modelos conceptuales que juntos describen un lenguaje del dominio [6].

En la ingeniería de software, los patrones se usan para permitir la reusabilidad de soluciones exitosas de problemas recurrentes. El propósito de los patrones se centra en incorporar calidad y productividad en las fases de desarrollo del software y brindar al desarrollador un punto de partida. La especificación de los patrones de análisis se puede realizar por medio de narración libre, plantillas o modelos conceptuales.

2.2 Plantilla de patrones de análisis

Como un intento de unificar las plantillas de patrones de análisis existente, en [3] se propone una plantilla que combina las características comunes de las ya existentes y añade nuevas características que no habían sido consideradas en plantillas actuales. La plantilla consta de 18 apartados, descritos a continuación:

1. Nombre: Identificador del patrón.
2. También conocido como: Otros nombres que hacen referencia a este patrón.
3. Historial: Registro cronológico de las versiones anteriores.
4. Ajustes estructurales: Definición de extensiones u omisiones de campos de la plantilla.
5. Problema: Breve descripción del problema que se resuelve a través del patrón.
6. Motivación: Descripción de las fuerzas involucradas y una situación representativa que motive el uso del patrón.
7. Contexto: Descripción detallada del contexto en el que el problema y la solución se repita y para el cual la solución es deseable.
8. Aplicabilidad: Descripción de las condiciones bajo las cuales el patrón puede ser aplicado.

9. Requerimientos: Listado de requerimientos funcionales, no funcionales, dependencias y contribuciones, identificación de conflictos y guía de soluciones, prioridades y participantes.
10. Modelado: Uso de modelos para representar procesos expresados en un nivel alto de abstracción.
11. Contexto resultante: Configuración del sistema después de aplicar el patrón.
12. Consecuencias: Ventajas y desventajas del uso del patrón.
13. Trampas anti-patrones: Las trampas más comunes que pueden originarse al aplicar el patrón.
14. Ejemplos: Uno o más ejemplos de aplicaciones que ilustren el uso del patrón.
15. Patrones relacionados: Lista de patrones similares que describan problemas o soluciones similares.
16. Patrones de diseño: Patrones de diseño o de arquitectura que pueden utilizarse para un mayor refinamiento.
17. Directrices de diseño: Consejos sobre la aplicabilidad del patrón, sin llegar a detalles muy específicos.
18. Usos conocidos: Ocurrencias conocidas de la aplicación del patrón en sistemas existentes. Se deben incluir al menos tres sistemas diferentes.

2.3 Plantilla de patrones de análisis estable

Es una plantilla de patrones de análisis considerada como una modificación a la plantilla de [3] en la cual se agrega el modelo de estabilidad expuesto en [7], la modificación consiste básicamente en añadir aspectos dinámicos y estructurales. Esta plantilla es actualmente es desarrollada por Rubí Celia Martínez Jiménez, alumna de CENIDET.

2.4 Modelo de Estabilidad de Software

Introducido en [7], presenta tres niveles conceptuales clasificados de acuerdo a su naturaleza:

- Temas de negocios perdurables (EBTs por el inglés enduring business themes): son las clases que presentan los conceptos básicos y perdurables d

- sistema. Por lo tanto, son extremadamente estables y forman el núcleo del modelo de estabilidad de software (SSM por el inglés software stability model). Describen los principales objetivos del sistema.
- Objetos de negocios (BOs por el inglés business objects): son las clases que mapean las EBTs del sistema en objetos más concretos. Los BOs son tangibles y externamente estables, pero son internamente adaptables.
- Objetos industriales (IOs por el inglés industrial objects): son las clases que mapean las BOs del sistema en objetos físicos.

2.5 Ontología

Es una especificación explícita de una conceptualización. Es una descripción formal de conceptos que pertenecen a un dominio, así como las relaciones existentes entre ellos [8].

2.6 Methontology

METHONTOLOGY [9] ha sido desarrollada por el Grupo de Ingeniería Ontológica de la Universidad Politécnica de Madrid. Esta metodología permite crear ontologías en el nivel de conocimientos, y tiene sus raíces en las actividades identificadas por el proceso de desarrollo de software propuesto por la organización IEEE [10] y en otras metodologías de ingeniería de conocimientos.

METHONTOLOGY proporciona guías sobre cómo llevar a cabo el desarrollo de la ontología a través de las actividades de especificación, conceptualización, formalización, implementación y mantenimiento.

2.7 Protégé

Protégé es una herramienta para el desarrollo de Ontologías y Sistemas basados en el conocimiento creado en la Universidad de Stanford, las aplicaciones desarrolladas con Protégé son empleadas en resolución de problemas y toma de decisiones en dominios particulares. La herramienta emplea una interfaz de usuario que facilita la creación de una estructura de frames con clases, slots e instancias de una forma integrada. Las ontologías de

Protégé pueden ser exportadas en una gran variedad de formatos incluyendo RDF(S), OWL y XML Schema.

Al igual que Eclipse, Protégé es un framework para el que otros proyectos sugieren plugins. La aplicación está escrita en Java y usa fuertemente Swing para crear su compleja interfaz. Protégé permite modelar ontologías de dos formas diferentes: 1) Protégé-Frames: Permite modelar ontologías de acuerdo al Open Knowledge Base Connectivity protocol [OKBC]; 2) Protégé-OWL: Permite construir ontologías siguiendo el Web Ontology Lenguaje [OWL].

2.8 Enterprise Architect

Enterprise Architect es una plataforma de alto desempeño para el modelado, visualización y diseño basada en el estándar UML 2.4.1. Ofrece trazabilidad completa desde mapas mentales, pasando por los requerimientos, hasta el diseño y la distribución del software con el nivel de eficiencia, robustez, herramientas de colaboración y seguridad requerida para sacar adelante proyectos altamente demandantes de cualquier tamaño.

Dentro de este capítulo se incluyeron conceptos que permiten sustentar la investigación de este trabajo de tesis. Se incluye información acerca de los patrones de análisis debido a que la investigación de esta tesis está enfocada principalmente al estudio y uso de los mismos. De igual forma se presentó la plantilla para patrones de análisis propuesta en [3], la cual combina las características comunes de las ya existentes y añade nuevas características que no habían sido consideradas en plantillas actuales para mejorar la descripción de un patrón de análisis. Otros conceptos importantes mencionados dentro de este capítulo son ontologías y el modelo de estabilidad de software, los cuales combinados nos dan la pauta para representar a los patrones de análisis como conocimiento.

En el capítulo siguiente se presentan algunos trabajos que utilizan patrones de análisis que sirven de comparativa con respecto a esta investigación.

Capítulo 3. Estado del Arte

En este capítulo se presentan los resultados del proceso de búsqueda de información relevante con respecto a trabajos de investigación relacionados que sirven como punto de referencia y comparación con el trabajo propuesto.

3.1 Antecedentes

De investigaciones que se han realizado en el departamento de Ciencias Computacionales en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), en el área de patrones de análisis y de ontologías se pueden mencionar los siguientes:

3.1.1 Construcción de una ontología mediante métodos semiautomáticos

En esta investigación [11] se proporciona una herramienta para ayudar al diseñador de ontologías en tareas de lectura, organización de textos y búsquedas de relaciones semánticas a partir de un corpus de documentos de texto.

3.1.2 Estudio de factibilidad para la formulación de reglas de combinación de patrones de diseño en arquitecturas orientadas a objetos

En esta investigación [12] se desarrolló las reglas formales basadas en gramáticas "Símbolo - Relación" para verificar la correctitud estructural de los patrones de diseño Observer, Composite e Iterator, así como las reglas de verificación de correctitud de la combinación de los patrones de diseño Composite-Observer, Composite-Iteraror, y Observe-Iteraror. Estas reglas fueron tomadas e implementadas en la herramienta Sistema Verificador de Patrones (SiVerPat).

3.1.3 Generación de ontologías en OWL a partir de diagramas de clases de UML mediante MDA (arquitectura dirigida por modelos)

En esta investigación [13] se hace uso de la Arquitectura Dirigida por Modelos (MDA por el inglés model driven architecture), y realiza transformaciones de metamodelos mediante el lenguaje de transformación ATL (Atlas Transformation Language) aunque en esta investigación no se llegaron a definir reglas de transformación para un lenguaje de transformación, se desarrollaron métodos que posteriormente podrán evolucionar a reglas de transformación.

3.1.4 Creación de una ontología para el dominio de e-gobierno en México

En esta investigación [14] se realizó una adaptación de la metodología Methontology para construir ontologías adecuándolas a necesidades específicas y se crea una ontología genérica del gobierno de México en sus tres niveles: Federal, Estatal y Municipal. Resultados

preliminares de la tesis doctoral “Definición del conjunto de operaciones unitarias, para realizar transformaciones entre modelos UML, apegadas a la MDA”, elaborado por el M.C. Juan Moisés Villamil Brito. Se han desarrollado tres reportes de avance, que contienen información útil.

3.1.5 Ambiente de modelado de arquitecturas de software conducido por reglas de combinación de patrones de diseño

En esta investigación [15] se desarrolló la herramienta SiVePat (Sistema Verificador de Patrones). Ésta, herramienta consiste de un Ambiente de Modelado Orientado a Objetos que genera y verifica automáticamente la estructura básica de tres patrones de diseño Observer, Composite e Iterator. Además, se genera y verifica automáticamente la combinación entre los patrones de diseño Observer-Composite, Composite-Iterator y Observer-Iterator.

3.1.6 Reglas para la combinación de patrones de diseño

En esta investigación [16] se continúa desarrollando teóricamente las reglas de combinación de patrones de diseño. Aquí se agregan las reglas de verificación estructural de los patrones de diseño Strategy, Composite, Template method y Factory method. Además, se agregaron las reglas de combinación de los patrones de diseño Strategy-Composite, Strategy-Factory method, Strategy-Template method, Composite-Template method, Composite-Factory method y Template method-Factory method.

3.1.7 Prototipo para la representación visual de patrones de análisis usando UML

Resultados de la investigación de la propuesta tesis de maestría de Rubí Celia Martínez Jiménez acerca de “**Prototipo para la representación visual de patrones de análisis usando UML**”, a graduarse antes de febrero del 2015 [17].

3.1.8 Almacenamiento y uso de patrones de análisis apegados a su semántica

Resultados de la investigación de la propuesta tesis de maestría de Ramiro Mar López Ramírez acerca de “**Almacenamiento y uso de patrones de análisis apegados a su semántica**”, a graduarse en 2015 [2].

Los trabajos de investigación anteriores mencionan uno de los problemas actuales que presentan los patrones de análisis, la cual consiste en la falta de semántica en su descripción. Varias tesis abordan la importancia de utilizar ontologías para solucionar ese problema [14]. Además, se explican metodologías para la construcción de ontologías. Los trabajos de tesis que influye en esta investigación es: a) La de Ramiro Mar López Ramírez la cual ha obtenido buenos resultados al construir ontologías para patrones de análisis; b) La de Rubí Celia Martínez Jiménez que actualmente está en desarrollo en la cual se propone una ontología de plantilla de patrones de análisis, la cual será utilizada para describir los patrones de análisis que se utilizarán en los casos de estudios de la presente investigación.

En la Tabla 3.1 se presenta una comparativa de los trabajos relacionados señalados en este apartado. A continuación, se describen los símbolos y columnas de la tabla mencionada.

- **Símbolos empleados:**

  - El símbolo "✗" en la intersección de una fila con una columna, significa que la característica es falsa.
  - El símbolo "✔" en la intersección de una fila con una columna, significa la característica es válida.
  - El símbolo "-" en la intersección de una fila con una columna, significa que no se tiene referencia de la característica en la documentación del trabajo relacionado.
  - OWL: Lenguaje de Ontologías para la Web, tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML.
  - OWL DL: Incluye todas las construcciones del lenguaje OWL, pero pueden ser utilizadas solamente bajo ciertas restricciones.
  - PAL: Lenguaje para programación con patrones.
  - LePUS: Lenguaje extendido de especificación de patrones que soporta su combinación.
  - ATL: Lenguaje de transformación de modelos que permite especificar cómo uno o más modelos objetivo pueden ser producidos desde un conjunto de modelos fuente.
  - MECOPA: Metodología para la Construcción de Ontologías de Patrones de Análisis.
  - OPPAE: Ontología de Plantilla para Patrones de Análisis Estables.

- **Columnas:**

  - Plantilla de patrones (1ra columna): Utilización de alguna plantilla para describir el patrón de estudio.
  - Modelado de procesos (2da columna): Contiene diversas herramientas de modelado de proceso de negocios.
  - Ontología (3ra columna): Utilización de ontología para resolución del problema de estudio.
  - Herramienta (4ta columna): Herramientas de apoyo para el desarrollo del trabajo de investigación.
  - Metodología o Lenguaje (5ta columna): Contiene los distingos métodos y lenguajes para la formalización y creación de ontologías.
  - Perspectiva del negocio (6ta columna): Define la perspectiva empleada para definir el negocio.
  - Producto resultado (7ma columna): Contiene los productos resultantes del trabajo de investigación, los cuales pueden ser, herramientas, modelos, componentes, arquitecturas, entre otros.

Tabla 3.1. Comparación de antecedentes relacionados

Antecedentes	Plantilla de patrón	Modelado de procesos	Ontología	Herramientas / Atributo	Metodología o lenguaje	Perspectiva del negocio	Producto resultado
[11]	✗	CU, Actividad, Secuencia	✓	Prompt, Protégé,	OWL DL	Modelo de valor	Herramienta - Métodos semiautomáticos
[12]	✓	Flujo, Relación, Clases	✗	Enterprise Architect, ModelMaker	PAL, LePUS	Patrones de diseño	Reglas de validación
[13]	✗	Clase, Actividad	✓	Eclipse EMF	ATL, OWL	Modelo de valor	Reglas de transformación
[14]	✗	Estado, Clase	✓	Protégé,	OWL	Modelo de valor	Herramienta
[15]	✓	Clases	✗	Enterprise Architect, ModelMaker	PAL, LePUS	Patrones de diseño	Herramienta - Reglas de combinación
[16]	✓	Clase	✗	Enterprise Architect, ModelMaker	-	Patrones de diseño	Validación de reglas de combinación
[17]	✓	GLR, UCM, Clases, CU	✓	Protégé	OWL	Patrones de análisis	Propuesta OPPAE
[2]	✓	GLR, UCM, Clases, CU	✓	Protégé	OWL	Patrones de análisis	Propuesta Método MECOPA
Propuesta	✓	GLR, UCM, Clases, CU	✓	Protégé,, Enterprise Architect	OWL DL	Patrones de análisis	Métodos de Validación y reglas de combinación

3.2 Trabajos relacionados

En este apartado se analizan aspectos relevantes que proponen diversos autores con respecto a sus resultados de investigación en temas de patrones de análisis que tienen una relación con la presente investigación.

3.2.1 Hierarchical patterns: a way to organize (analysis) patterns

En este trabajo de investigación [18] se jerarquizan patrones de análisis como un medio para su categorización y combinación. La jerarquía de patrones de análisis propuesta se divide en tres niveles: a) Patrón de Análisis General, b) Patrón de Análisis Especializado, c) Patrón de Dominio. La clasificación de niveles en los patrones de análisis permite lograr generalidad y abstracción adecuada. A nivel más general se puede obtener un framework para un nivel más especializado, el cual se utiliza cuando el nivel especializado no se ha creado todavía. Además, el nivel especializado proporciona una guía específica a un analista durante la construcción de un modelo conceptual de un sistema del mundo real.

3.2.2 On the integration of stable analysis patterns with traditional patterns

Este trabajo de investigación [4] se propone un enfoque interactivo que consta de 4 fases con las cuales integran patrones estables y patrones tradicionales para obtener un patrón compuesto, o en general, un nuevo modelo de análisis. Este enfoque se comprobó con dos casos de estudio:

- Patrón alquiler de recursos con capacidad de negociación. Se obtiene de la integración del “patrón de negociación (estable)” con el “Patrón de Alquiler de Recursos (tradicional)”.

- Patrón pedido y envió con confianza. Se obtiene de la integración del “patrón de pedido y el envío (tradicional)” con el “patrón confianza (estable)”.

Este enfoque evita duplicar esfuerzos en el desarrollo de patrones de análisis que abordan un mismo problema. Además, el enfoque permite utilizar prácticamente cualquier patrón de análisis disponibles, por lo tanto, se extiende el espacio reutilización de patrones de análisis.

3.2.3 The umbrella pattern language: towards a pattern language for analysis patterns integration

En este trabajo de investigación [19] se propone la creación de un framework el cual permitirá integrar patrones de análisis. Esté, se enfocará en ver la forma adecuada para llevar a cabo la integración de patrones de análisis estables con patrones tradicionales. Además, se tiene como objetivo a largo plazo el desarrollo de un modelo de integración de tecnologías de análisis con diferentes estructuras.

3.2.4 Towards a framework for analysis patterns evaluation

En este trabajo de investigación [20] se propone un framework para evaluar patrones de análisis. Esté incorpora cualidades fundamentales que, en conjunto, forman patrones de análisis efectivos. Además, tiene como objetivo principal proporcionar una estructura que incorpore las principales cualidades que forman un "buen" patrón de análisis. De esta manera, un desarrollador hará uso de este framework, junto a su experiencia y la intuición, para seleccionar el patrón de análisis adecuado para su problema. El framework enfatiza, que debe cumplir un patrón de análisis para poder ser capaz de contribuir a la estabilidad del sistema que lo utiliza. El framework de evaluación propuesto tiene las siguientes propiedades esenciales:

- Sin Tiempo: En otras palabras, estable. La estabilidad influye en la reusabilidad de un modelo.
- Generalidad: Significa qué los patrones de análisis que modelan un problema específico debería ser fácil de utilizar para modelar el mismo problema cada vez que aparezca.
- Usabilidad y reusabilidad: Los patrones de análisis deben ser represados de manera clara que permita hacer su reusó fácil.
- Presentación de los aspectos básicos: Los patrones de análisis deben presentar aspectos básicos del problema.
- Adaptabilidad: Esta propiedad está ligada a la propiedad de estabilidad. Los modelos deben ser fácilmente adaptados y modificados.
- Integrabilidad: En la mayoría de los casos, los patrones son integrados con otros patrones y/o el resto del sistema para formar una parte de modelos más grades.

3.2.5 Improving analysis patterns reuse: an ontological approach

En este trabajo de investigación [21] se propone una visión de mejora en la reusabilidad de patrones de análisis por el uso de una ontología. El autor propone un proceso para la construcción de modelos de análisis que consta de cuatro enfoques principales:

- Extracción del conocimiento: En esta fase una colección de patrones existentes y otras fuentes de conocimiento son analizados cuidadosamente, por ejemplo, modelos de dominio y modelos relacionados para otros proyectos.
- Desarrollo de ontología: De la primera fase el "Extracción del conocimiento" el resultado obtenido (conocimiento extraído) se introduce en esta fase donde se lleva a cabo el desarrollo de una ontología que captura el conocimiento extraído.
- Reúso del conocimiento: En esta fase se lleva a cabo la construcción de modelos de análisis a partir de un conocimiento activo que se obtiene de la ontología resultante de la segunda fase en donde se codifica el conocimiento que se materializa en los patrones de análisis más la experiencia de la organización donde se aplica esta ontología.
- Argumentación del conocimiento: En esta fase se lleva a cabo un proceso de retroalimentación del conocimiento para mejorar la argumentación y la ontología actual.

3.2.6 A systematic analysis patterns specification

En este trabajo de investigación [3] se propone una plantilla para la especificación de patrones de análisis. Se utiliza un proceso sistemático que consta de 18 pasos utilizando una plantilla con la cual se obtiene información más detallada sobre el patrón de estudio. El objetivo central de este trabajo de investigación es la normalizar y unificar la representación de patrones de análisis, y por lo tanto su claridad e integridad.

3.2.7 Enterprise ontology

En este trabajo de investigación [22] se presenta un ejemplo de composición por ensambles en el cual se describe la importancia de identificar componentes atómicos y subconjuntos, para llevar a cabo un montaje de componentes. Se presenta como ejemplo un diagrama de árbol en el cual utiliza el proceso para fabricar una bicicleta. En este ejemplo se utiliza el “El patrón básico de Transacción” el cual consiste en dirigir la conversación cara a cara entre dos sujetos, uno en el papel de inicializador y el otro en el papel de ejecutor. Para la solución de

ensamblaje se utiliza “la estructura de cierre de una transacción” para llevar a cabo de manera exitosa la integración de componentes.

3.2.8 The role of analysis patterns in systems analysis

En este trabajo de investigación [23] se propone un método de análisis de sistemas con patrones de análisis, el cual se basa en el paradigma de reusó. De acuerdo al paradigma empleado se plantean dos procesos: el primero es el desarrollo para reusó en el cual se producen los bienes reutilizables que se almacenan en un repositorio, es en este proceso donde entran en juego los patrones de análisis; mientras que el segundo, es el desarrollo con reusó en el cual se desarrollan aplicaciones con componentes reutilizables. En el caso de estudio planteado se implementan cuatro patrones de análisis, sin embargo, no se hace alusión al uso de alguna plantilla de patrones de análisis ni se realizan transformaciones de ningún tipo.

3.6.9 Composing analysis patterns to build complex models: flight reservation

En este trabajo de investigación [24] se propone el concepto de patrón de análisis semántico (SAP por el inglés semantic analysis pattern), el cual es un patrón que describe un pequeño conjunto coherente de casos de usos que juntos describen una aplicación genérica básica. El objetivo de un SAPs es crear patrones de análisis complejos (modelos orientados a objetos), el autor presenta un caso de estudio utilizando una aplicación de viaje el cual es un “sistema de reserva de vuelos”, en este ejemplo se observa que al realizar la composición de algunos patrones simples es posible construir estas aplicaciones como un SAP de un modo sistemático. Los patrones atómicos o componentes corresponden a funciones específicas del sistema y pueden ser tanto nuevos patrones o modelos existentes, quizás especializados para la aplicación. Debido a la forma en que se construye toda la aplicación, el modelo resultante es también flexible y reutilizable. El sistema compuesto final, también podría ser la base para un framework. Es posible visualizar la estructura de un sistema de gran tamaño como un conjunto de componentes unidos, donde cada uno se basa en un conjunto diferente de aspectos funcionales.

En la Tabla 3.2 se presenta una comparativa de los trabajos relacionados señalados en este apartado. A continuación, se describen los símbolos y columnas de la tabla mencionada.

- **Símbolos empleados:**

  - El símbolo "✗" en la intersección de una fila con una columna, significa que la característica es falsa.
  - El símbolo " ✔ " en la intersección de una fila con una columna, significa que la característica es válida.
  - El símbolo "-" en la intersección de una fila con una columna, significa que no se tiene referencia de la característica en la documentación del trabajo relacionado.

- **Columnas:**

  - Plantilla de patrones (1ra columna): Utilización de alguna plantilla para describir el patrón de estudio.
  - Ontología (2da columna): Utilización de ontología para resolución del problema de estudio.
  - Patrones utilizados (3ra columna): Tipos de patrones utilizados para los trabajos de estudio.
  - Utiliza combinación de patrones (4ta columna): Realiza de la integración de dos o más patrones de estudio.
  - Mantiene integridad de los patrones (5ta columna): Se mantiene la estructura básica del patrón de estudio.
  - Perspectiva del negocio (6ta columna): Define la perspectiva empleada para definir el negocio.
  - Modelado de procesos (7ma columna): Contiene las diversas herramientas de modelado de proceso de negocios.

Tabla 3.2. Comparación de trabajos relacionados

Trabajos relacionados	Plantilla de patrones	Ontología	Patrones utilizados	Utiliza combinación de patrones	Mantiene la integridad de los patrones	Perspectiva del negocio	Modelado de procesos
[18]	✗	✗	Partes, Objetos, Rendicion de cuentas, Operaciones	✓	✓	Patron de Analisis	Componentes
[4]	✗	✗	Medicion, envio, negociacion	✓	✓	Patron de Analisis	Clases
[19]	✗	✗	Renta de recursos, negociacion.	✓	✓	Patrones de Analisis	Clases
[5]	✗	✗	Cuenta	✗	✓	Patrones de Analisis	Clases
[21]	✗	✓	Cuenta	✗	✓	Patron de Analisis	Clases
[3]	✗	✗	Admisiones	✗	✓	Patron de analisis	Clase, Estado, Secuencia
[22]	✗	✗	Transacción	✗	✓	Patron de Analisis	Actividad
[23]	✗	✗	reserva y el uso de entidades reutilizables, observaciones y mediciones , pedido y el envío de un producto, inventarios	✓	✓	Patron de analisis	
[24]	✗	✗	eservacion de	✓	✓	Patron de analisis	Actividad, Secuencia, Clases
Propuesta	✓	✓	Cuenta, Renta de recursos, negociacion.	✓	✓	Patron de Analisis	GRL, UCM, Clases, CU

Dentro de este capítulo se realizó una comparativa de los principales antecedentes y trabajos relacionados previos a este trabajo, se explicaron las similitudes y las diferencias con la presente investigación, además de servir como punto de referencia para ver la importancia de esta aportación con respecto de las demás en la sociedad. De acuerdo a los trabajos presentados podemos afirmar que no existen formas de realizar una composición y validación de Patrones de análisis que apliquen reglas formales para su mejor descripción, lo que nos lleva a proponer una metodología de validación para patrones de análisis (ver Capítulo 4).

Capítulo 4. Metodología de Validación de Consistencia de Patrones de Análisis

En este capítulo se hace una adaptación de la Metodología para la Construcción de Ontologías de Patrones de Análisis (MECOPA) para la creación de ontologías de patrones de análisis. Se explica el proceso de desarrollo y adaptación de la "Metodología de Validación de Consistencia de Patrones de Análisis (MEVASE)", la cual permite mejorar la descripción en patrones de análisis desarrollados a través de MECOPA.

4.1. Introducción

El termino patrones de análisis, actualmente, constituye un tópico de interés por parte de la comunidad de ingeniería de software, independientemente del ámbito de investigación. Su uso como herramienta durante la fase de análisis en el desarrollo de un producto de software, ayuda a entender el problema mirando más allá de los requisitos "superficiales".
Actualmente la metodología MECOPA [2] detalla procesos necesarios por los que deben pasar los patrones de análisis para hacer un refinamiento del modelo de dominio que se está representado, es decir, entender en detalle el negocio y sus reglas, así como representarlo mediante una ontología.

La metodología propuesta para este trabajo de investigación se anexa como una nueva fase al proceso propuesto en MECOPA, permitiendo validar la descripción de patrones de análisis desarrollados a través de esta metodología.

4.2. Complemento de MECOPA con la Metodología de Validación de Consistencia de Patrones de Análisis en MECOPA.

La metodología MECOPA está compuesta de ocho fases, las cuales las cuales constituyen un proceso para crear la ontología para un patrón de análisis (verFigura 4.1).

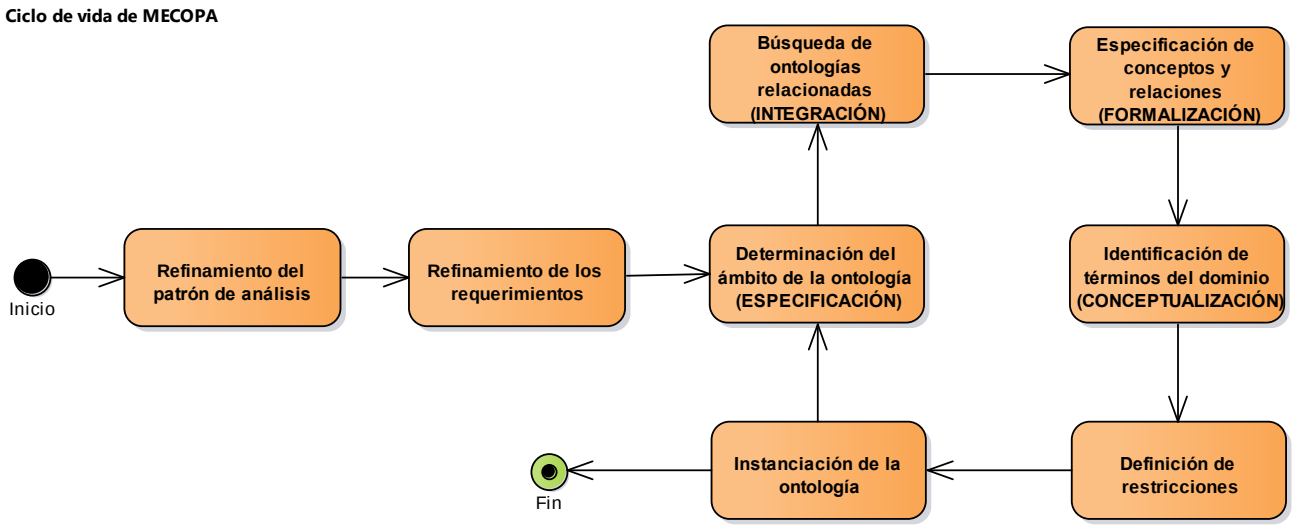


Figura 4.1. Ciclo de vida de la metodología MECOPA

Para incluir una fase de validación en MECOPA, se llevó a cabo el desarrollo y adaptación de la "**Metodología de Validación de Consistencia de Patrones de Análisis (MEVASE)**", la cual permite mejorar la descripción en patrones de análisis desarrollados incluyendo su semántica. Estas mejoras en MECOPA se logran mediante una retroalimentación en las fases de:

- ✓ Identificación de términos del dominio (Conceptualización)
- ✓ Especificación de conceptos y relaciones (Formalización)
- ✓ Definición de restricciones

Se realiza un refinamiento de la información obtenida, mejorando la descripción de tablas de relaciones binarias de: a) Conceptos, b) Roles, y c) Propiedades, para que represente mejor el modelo de domino del problema que describe un patrón de análisis (ver Figura 4.2).

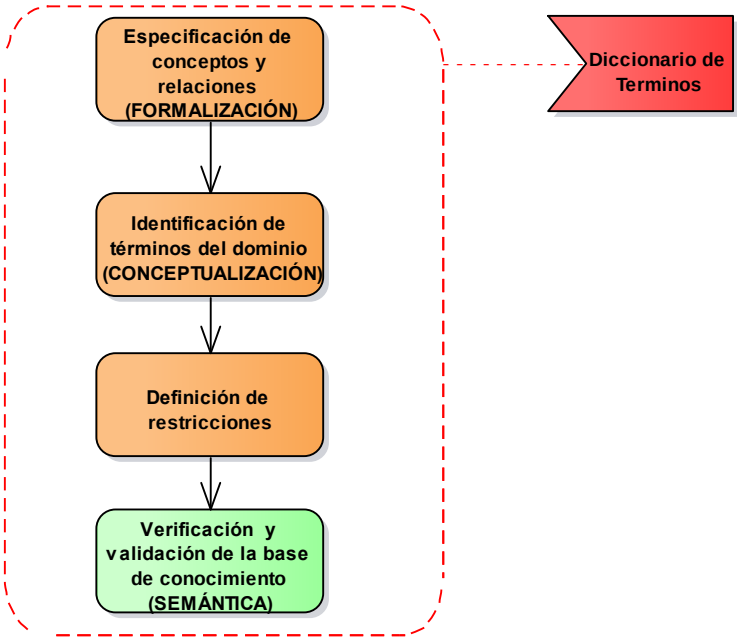


Figura 4.2. Refinamiento de información en fases de MECOPA

La información representada en las tablas de relaciones binarias se valida en la fase "Verificación y Validación de la base de conocimiento (SEMANTICA)" que constituye la metodología MEVASE, propuesta en este trabajo de investigación (ver Figura 4.3).

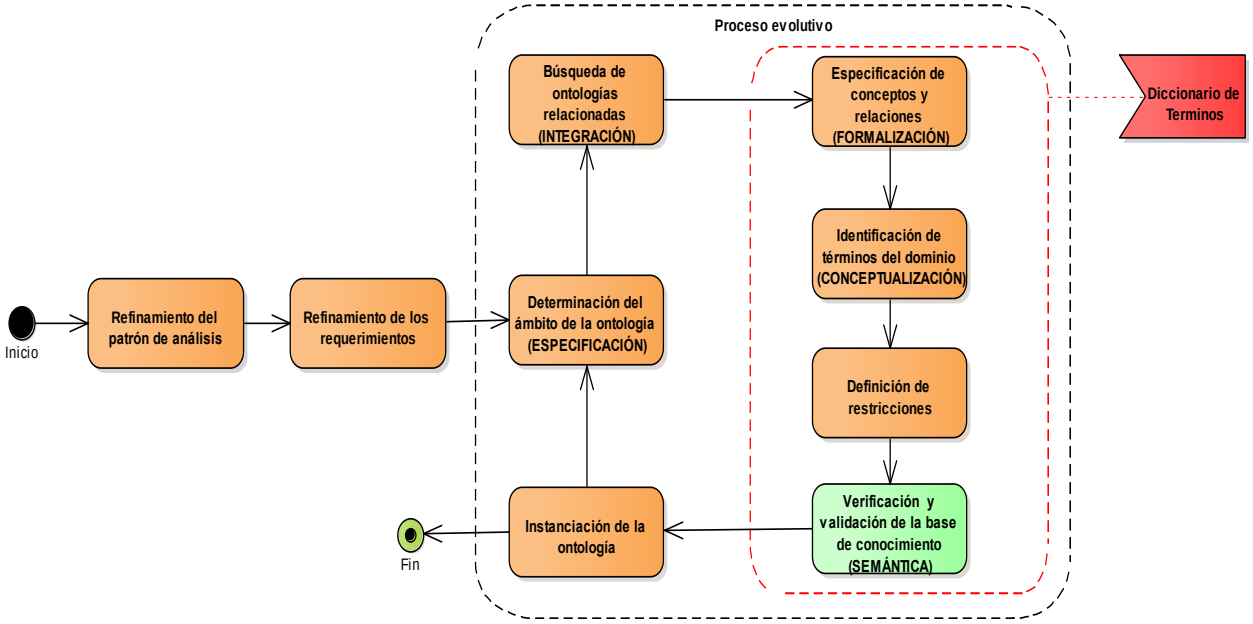


Figura 4.3. Implantación de la metodología MEVASE en MECOPA

La metodología MEVASE permite la retroalimentación de información a las fases originales de MECOPA, con lo cual analistas de sistemas pueden: 1) Verificar la consistencia de un patrón de análisis, y 2) Validar la ontología de un patrón de análisis durante su ciclo de desarrollo.

4.3 Fases que componen al ciclo de vida de la metodología MEVASE

La metodología MEVASE permite la formalización de patrones de análisis a través de lógica matemática, es decir, utiliza como lenguaje formal la "Lógica Descriptiva". La Lógica Descriptiva (DL por el inglés description logics) es una familia de formalismos basados en lógica, para la Representación de Conocimiento. Usando la DL y el proceso de la metodología MEVASE se puede verificar y validar las fases de "Identificación de términos del dominio (Conceptualización), Especificación de conceptos y relaciones (Formalización), Definición de restricciones" (ver Figura 4.3).

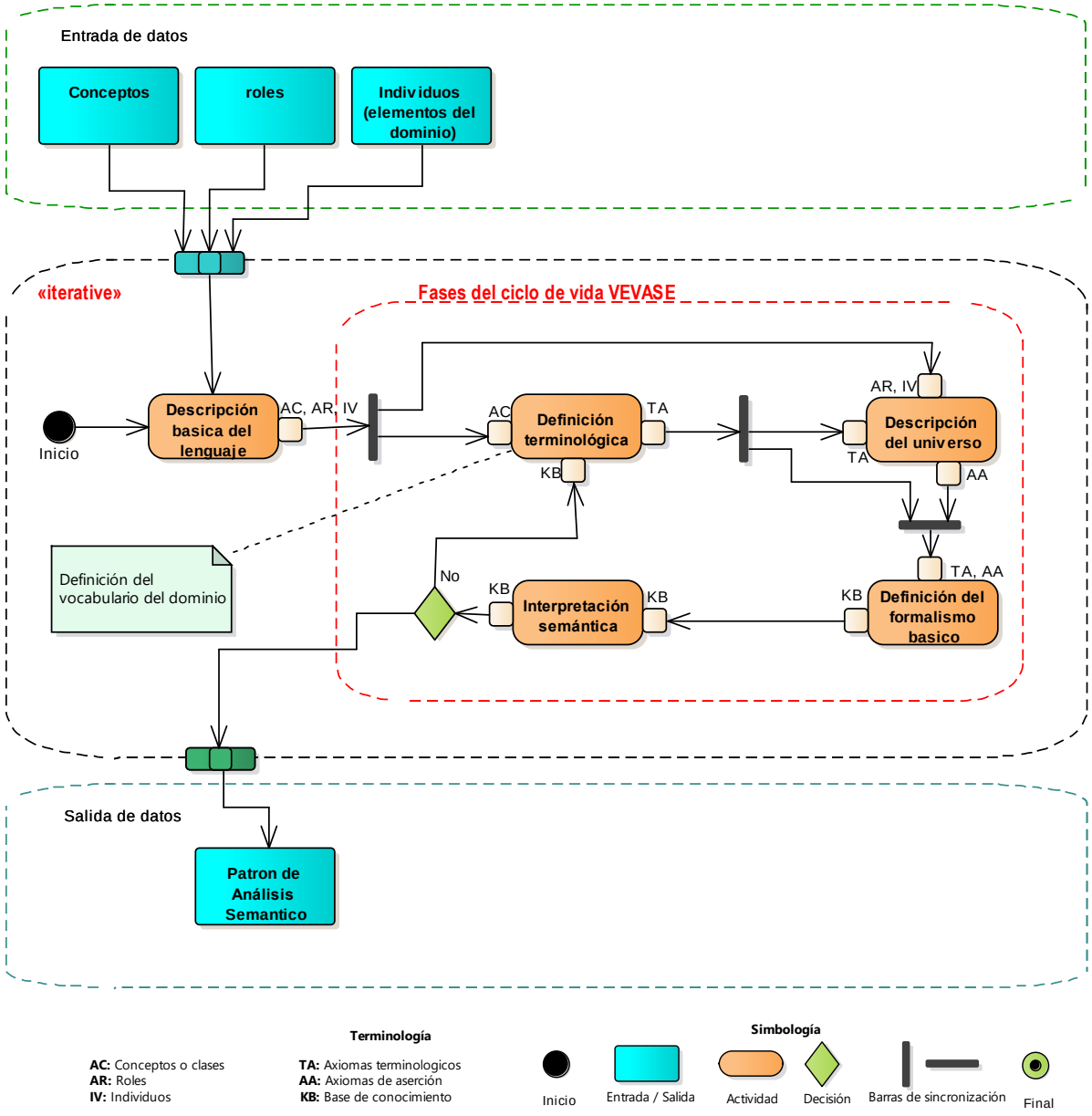


Figura 4.4. Proceso de la metodología MEVASE

La Figura 4.4, representa la estructura general del ciclo de vida de la metodología MEVASE, propuesta para este proyecto de investigación. En la Figura 4.5, se detallan cada una de las fases de MEVASE.

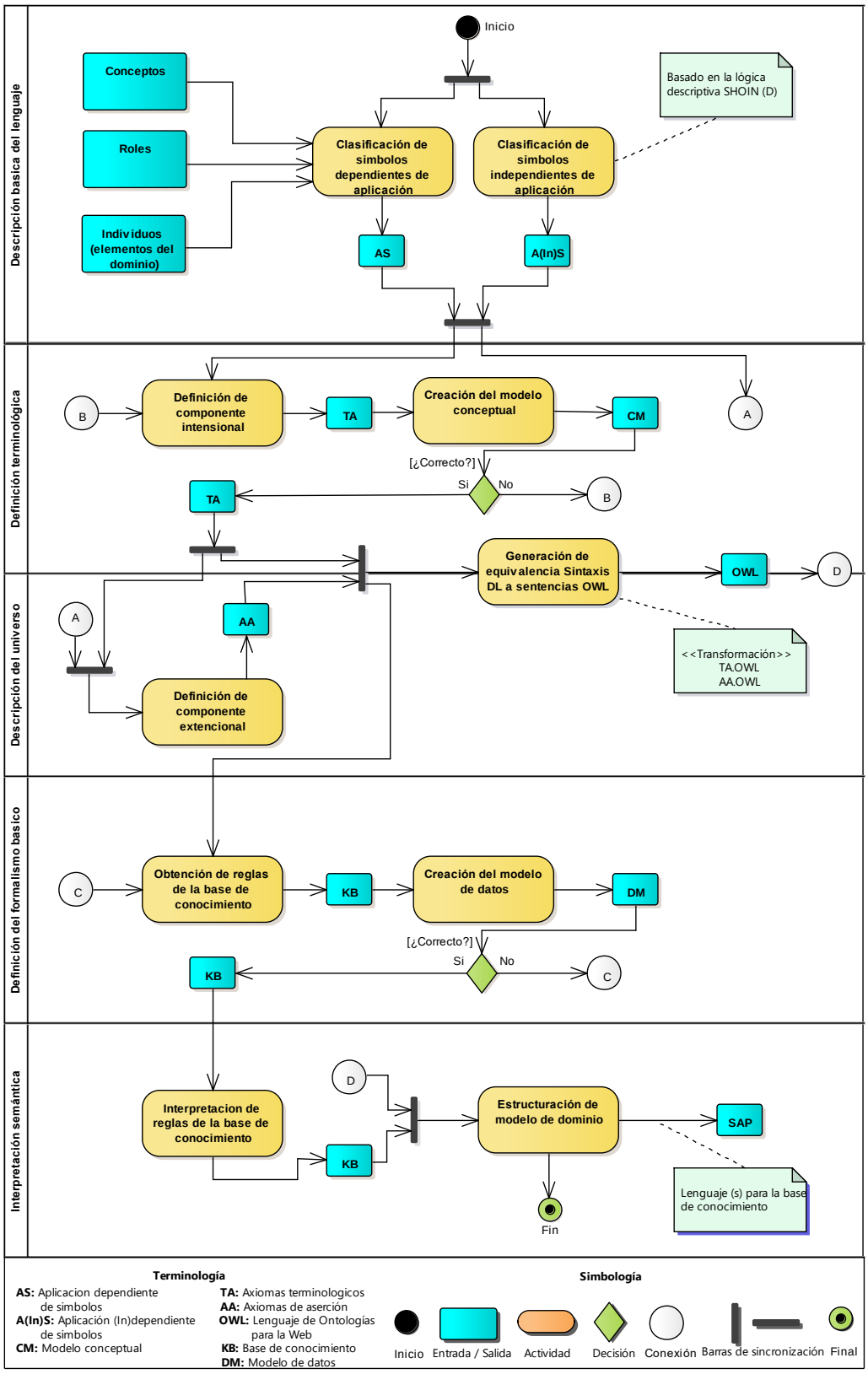


Figura 4.5. Actividades para cada fase de la metodología MEVASE

A continuación, se explica en que consiste cada una de las fases que componen el proceso de MEVASE en el cual se trataran ejemplos relacionados con la representación y validación formal del patrón de análisis de cuenta (AAP por el inglés account analysis pattern), el cual es el ejemplo de estudio del presente trabajo de investigación.

4.3.1 Descripción básica del lenguaje

En esta fase, utilizando una notación matemática de la lógica descriptiva, se adquieren los elementos a utilizar para definir la "representación del conocimiento" en un lenguaje basado en: a) Símbolos dependientes de aplicación, b) Símbolos independientes de aplicación.

4.3.1.1 Clasificación de símbolos dependientes de aplicación

En esta actividad se identifica el conjunto de símbolos dependientes de aplicación (AS por el inglés application dependent symbols), los cuales son objetos y relaciones del mundo real determinados por el dominio de estudio que el patrón de análisis representa. El conjunto AS se divide en tres subconjuntos disjuntos: 1) Conjunto de conceptos atómicos (AC por el inglés atomic concepts) representando clases de objetos; 2) Conjunto de roles atómicos (AR por el inglés atomic roles) representa relaciones binarias sobre objetos; 3) Conjunto de individuos (IV por el inglés individuals) representa objetos individuales de la aplicación de dominio.

En la **Tabla** 4.1 se presenta un ejemplo de símbolos dependientes de aplicación, que clasifica los elementos que pertenecen a cada uno de sus subconjuntos AC, AR, IV.

.

Tabla 4.1. Ejemplo de símbolos dependientes de aplicación

$$
\begin{aligned}
AC &= \{Cuenta, Cliente\} \\
AR &= \{esAbiertaPor\} \\
IV &= \{cta, cte\} \\
AS &= AC \cup AR \cup IV \\
&= \{ Cuenta, Cliente, esAbiertaPor, cta, cte\}
\end{aligned}
$$

4.3.1.2 Símbolos (In) dependientes de aplicación

En esta actividad se adquieren los símbolos dependientes de aplicación que forman un conjunto de conectores definidos por el tipo de notación lógica descriptiva (DL). Estos conectores se utilizan para construir conceptos complejos a partir de los atómicos.
Un ejemplo de los elementos básicos que pertenecen a los símbolos (In) dependiente de aplicación, usando notación de la lógica descriptiva básica (ver Anexo B) se presenta en la Tabla 4.2.

Tabla 4.2. Ejemplo de símbolos (In) dependientes de aplicación

$A(In)S = \{ALC\}$
$= \{ A$ (*concepto atómico*)
⊤ (*concepto universal*)
⊥ (concepto vacío)
¬ (*negación o complemento*)
⊓ (*conjunción o intersección*)
⊔ (*disyunción o unión*)
∀ (*cuantificador universal o restricción de valor*)
∃ (*Cuantificador existencial*) }

Mientras los símbolos independientes de aplicación los cuales son los elementos variantes de un dominio pueden elegirse arbitrariamente (ver Tabla 3), el conjunto de conectores es de la lógica descriptiva. La expresividad de la lógica descriptiva se determina por los constructores disponibles, para un concepto específico y roles.

4.3.2 Definición terminológica

En esta fase se define el componente intencional (IC por el inglés intentional component) [25].

4.3.2.1 Componentes intencionales

El componente intencional es un conjunto finito de axiomas terminológicos (TA por el inglés terminológica axioms) los cuales permiten representar el nivel de conocimiento del esquema (estructura) sobre el dominio de estudio. La descripción de conceptos está determinada a través de dos posibles tipos de axiomas terminológicas: 1) Implicaciones conceptuales, y 2) Equivalencias conceptuales. Estos axiomas permiten definir las relaciones generales entre

conceptos (o clases), que se mantienen en todo momento, para todos los posibles objetos del domino de conocimiento.

4.3.2.1.1 Implicaciones conceptuales

El axioma de implicación conceptual $(C \sqsubseteq D)$ expresa que todas las instancias de descripción del concepto C son también instancias de descripción del concepto D.

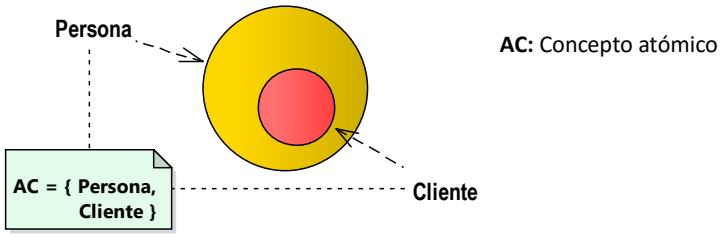


Figura 4.6. Implicación conceptual

El siguiente ejemplo $\text{Cliente} \sqsubseteq \text{Persona}$, indica que todas las instancias (elementos) del concepto cliente son también instancias (elementos) del concepto Persona, en la Figura 4.6 muestra una representación visual de este axioma.

4.3.2.1.2 Equivalencias conceptuales.

El axioma de equivalencia conceptual $(C \doteq D)$ expresa que el conjunto de instancias de descripción del concepto C es equivalente al conjunto de instancias de descripción del concepto D. El siguiente ejemplo $\text{ClienteBiblioteca} \doteq \text{Cliente} \sqcap \exists 1 \text{realiza}.\text{Transacción}$, define el concepto de "ClienteBiblioteca", como un "Cliente" que "realiza" alguna "Transacción", en la Figura 4.7 se muestra una representación visual de este axioma.

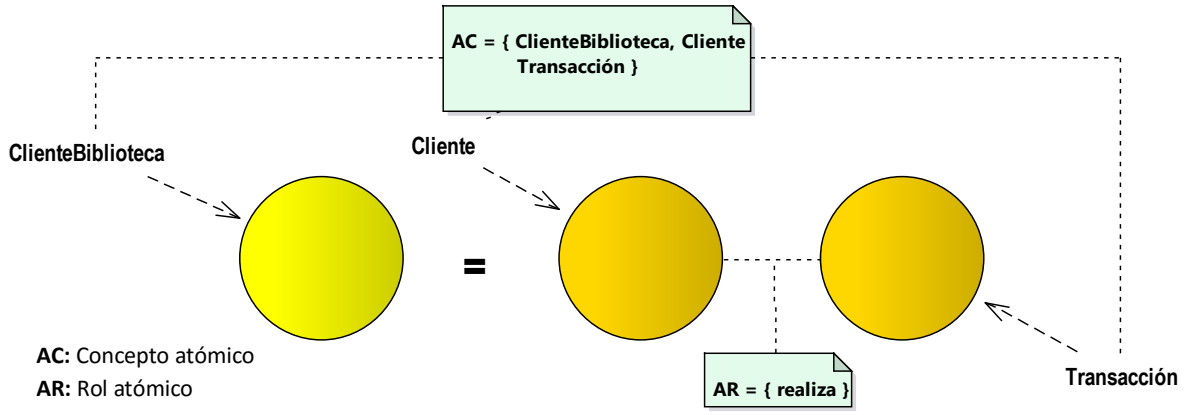


Figura 4.7. Equivalencia conceptual

ClienteBiblioteca es un concepto que representa la clase de objetos que son "clientes de una biblioteca". Cliente es un concepto que representa la clase de "clientes". El concepto Transaccion representa el conjunto de "transacciones" posibles para un cliente. El rol realiza representa la relación binaria entre las instancias de los conceptos transacciones y clientes.

Por lo tanto, el conjunto de axiomas terminológicos generados en esta fase es considerado como una representación basada en la lógica de un diagrama entidad-relación o un diagrama de clase UML. Esta representación es la estructura estática del dominio de discurso.

4.3.2.1.3 Creación del modelado conceptual

En esta actividad de acuerdo a los axiomas obtenidos en el componente intencional se demuestran la relación de subsunción (considerar algo parte de un concepto más amplio), formando una jerarquía, de conceptos. Establecida mediante un TA (axioma terminológico) el cual determina un orden parcial y por lo tanto es posible definir un grafo acíclico que permite validar la estructura para el patrón de estudio.

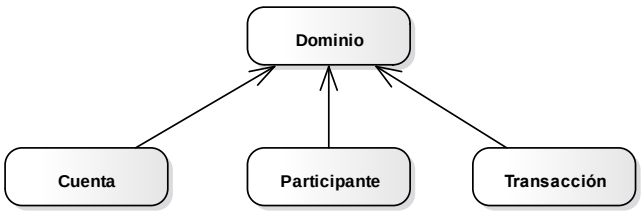


Figura 4.8. TBox (Grafo aciclico) para el domino de estudio

En Figura 4.8 se muestra un ejemplo de un grafo dirigido, el cual no debe tener recursividad en su estructura para el dominio de estudio que está representando.

4.3.3 Descripción del universo

En esta fase se define el componente extensional (EC por el inglés extensional component) [25] en base a símbolos (in) dependientes de aplicación.

4.3.3.1 Definición de componente extensional

El componente extensional es un conjunto finito de axiomas de aserción (AA por el inglés abox assertion) el cual contiene instancias particulares con nombre según el dominio de interés [26]. Los AA realizan dos tipos de afirmaciones: a) Afirmaciones sobre conceptos, b) Afirmaciones sobre relaciones, las cuales permiten validar el lenguaje del domino definido en los TA (axioma terminológico).

4.3.3.1.1 Afirmaciones sobre conceptos

Las afirmaciones sobre conceptos indican cuando un objeto (individuo) de la vida real es elemento de un concepto primitivo (ver sección 4.3.2.1.1 Implicaciones conceptuales) o un concepto definido (ver sección 4.3.2.1.2 Equivalencias conceptuales.) (Ver Figura 4.9).

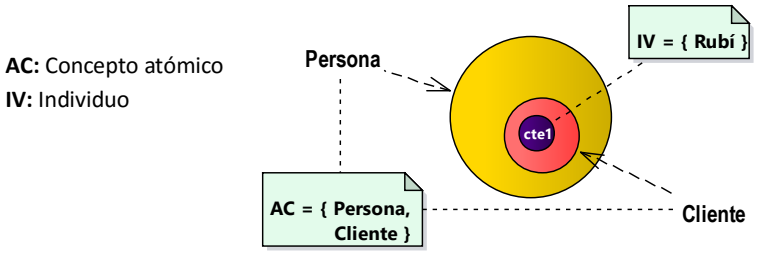


Figura 4.9. Afirmaciones sobre conceptos

El siguiente ejemplo $Persona \sqcap Cliente\ (Rubí)$, establece que el individuo Rubí es un elemento del concepto cliente, y es una persona. Teniendo en cuenta la definición anterior de cliente, se puede derivar de esta afirmación que Rubí es una instancia del concepto cliente.

4.3.3.1.2 Afirmaciones sobre roles

Las afirmaciones sobre roles indican la existencia de relaciones binarias entre individuos (ver Figura 4.10).

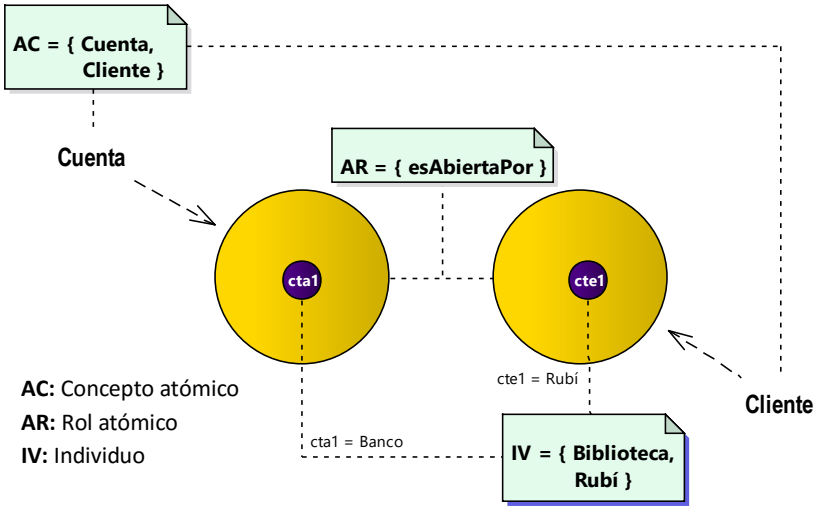


Figura 4.10. Afirmaciones sobre roles

Del mismo modo, $esAbiertaPor\ (Biblioteca, Rubí)$ especifica que la cuenta Biblioteca fue abierta por el cliente Rubí.

4.3.3.1.3 Generación de equivalencias Sintaxis DL a Sentencias OWL

En esta actividad se realiza la equivalencia de la sintaxis DL de las expresiones definidas en las secciones 4.3.2 Definición terminológica y 4.3.3 Descripción del universo, hacia sentencias OWL que declaran sobre: clases, individuos, propiedades; aserciones sobre: clases, individuos, propiedades, entre otros (ver Figura 4.11).

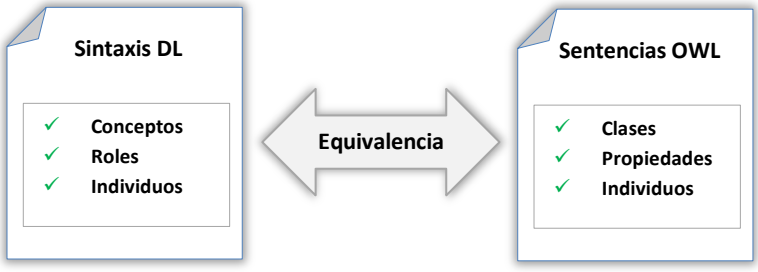


Figura 4.11. Equivalencias Sintaxis DL a Sentencias OWL

Declaración de clase

La equivalencia o transformación de clases que son parte del universo en una sintaxis DL a sentencias OWL, se realiza de la siguiente manera:

Sintaxis DL	
Cuenta ⊑ T	(1)
Sentencia OWL	
Declaration(Class(:Cuenta))	

Aserción Subclase

Las aserciones en subclases declaran que todos los individuos que pertenecen a una clase pertenecen también a otra clase (una clase es subsumida de otra clase). La equivalencia de la sintaxis DL a sentencias OWL se realiza de la siguiente manera:

Sintaxis DL	
Cliente ⊑ Persona	(2)
Sentencia OWL	
SubClassOf(:Cliente :Persona)	

Restricciones y axiomas de propiedades de objetos

Las restricciones y axiomas de propiedades afectan directamente a la relación binaria entre objetos. La equivalencia de la sintaxis DL a sentencias OWL se realiza de la siguiente manera:

Sintaxis DL	
Cuenta ⊓ ∀1esAbiertaPor. Cliente	(3)
Sentencia OWL	
SubClassOf(:Cuenta ObjectAllValuesFrom(:esAbiertaPor :Cliente))	

4.3.4 Definición del formalismo básico

La fase definición del formalismo básico establece una "base de conocimiento (KB por el inglés knowledge base)", el cual es una colección de información, incluida en componentes: a) intencional, b) extensional para un dominio de estudio.

4.3.4.1 Obtención de reglas de la base de conocimiento

En esta actividad se obtienen las reglas (axiomas DL) creados en las fases: 1) "Definición terminológica", y 2) "Descripción del Universo", permitiendo definir la base de conocimiento Del dominio de estudio. La base de conocimiento KB debe contener información especificada en un lenguaje declarativo tal como reglas lógicas. Un ejemplo de un KB se presenta en la Tabla 4.3.

Tabla 4.3. Ejemplo de la base de conocimiento para un patrón de análisis

KB	**={**	
	Cuenta ⊑ T	**(1)**
	Participante ⊑ T	**(2)**
	Transacción ⊑ T	**(3)**
	Persona ⊑ Participante	**(5)**
	Cliente ⊑ Persona	**(6)**
	Cliente ⊓ ∃1realiza. Transacción	**(7)**
	Persona ⊓ Cliente (Rubí)	**(8)**
	esAbiertaPor (Biblioteca, Rubí)	**(9)**
}		

T: Dominio

4.3.4.2 Creación del modelado de datos

En esta actividad se crea un diagrama visual permitiendo validar la estructura definida de acuerdo a las reglas definidas en la base de conocimiento (ver sección 4.3.4.1 Obtención de reglas de la base de conocimiento) a través de la evidencia lógica basado en la experiencia (ver Figura 4.12).

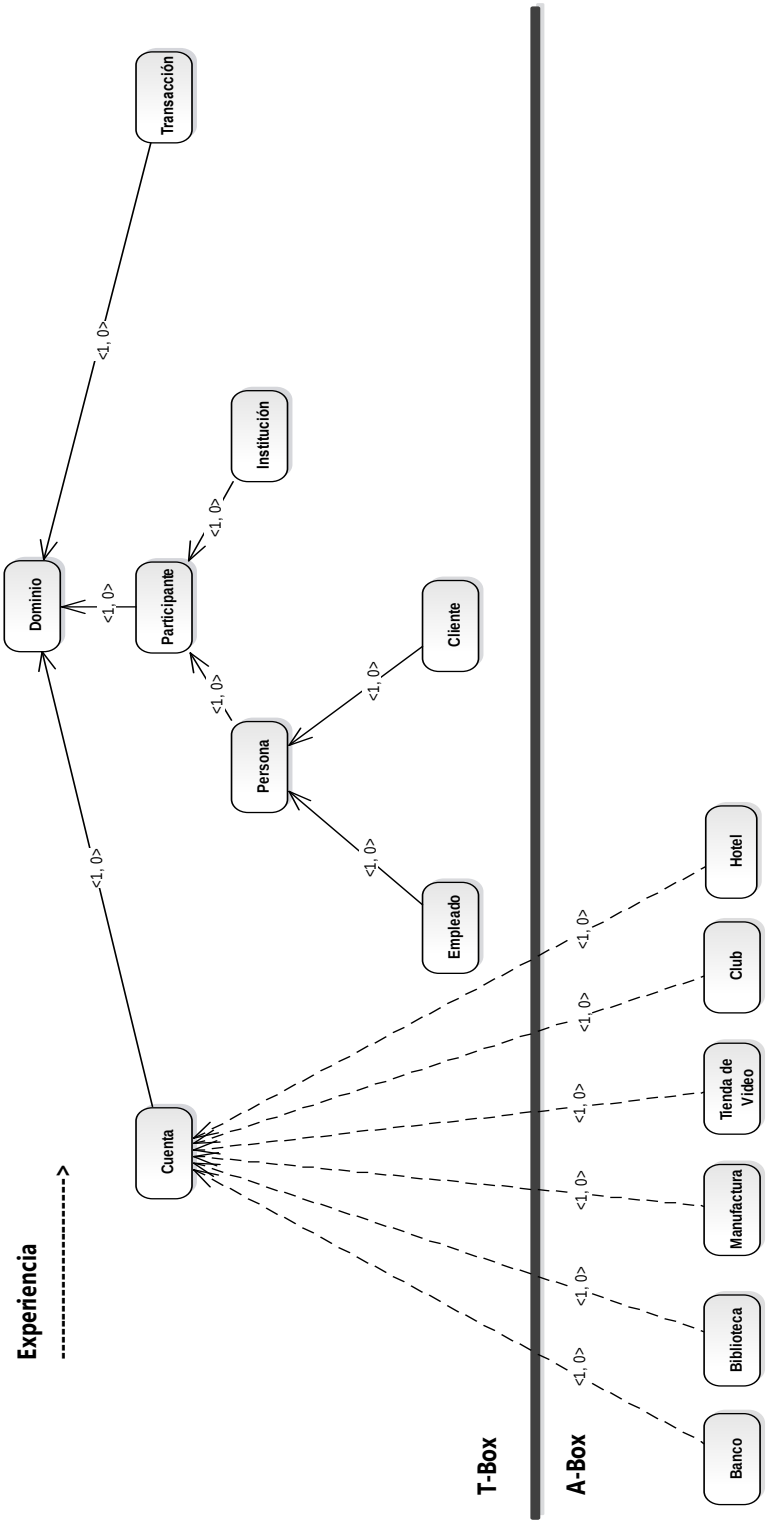


Figura 4.12. Modelado de datos

4.3.5 Interpretación semántica

En esta fase se realiza una correspondencia entre los objetos y relaciones del mundo real con los objetos definidos en la base de conocimiento (ver sección 4.3.4.1 Obtención de reglas de la base de conocimiento). La interpretación (*I* por el inglés interpretation)(ver Anexo B: Definición 1.19) se da por un par (Δ^I, $.^I$) donde: Δ^I es un conjunto no vacío de objetos llamado *dominio de interpretación* y $.^I$ es una función asignada a cada concepto atómico, cada rol atómico, y cada individuo representado en la base de conocimiento (ver Figura 4.12).

4.3.5.1 Interpretación de reglas de la base de conocimiento

En esta actividad la descripción de conceptos DL, definidos en la base de conocimiento (ver sección 4.3.4.1 Obtención de reglas de la base de conocimiento) son interpretados con respecto a una posibilidad infinita de un conjunto de objetos del dominio de interpretación

y una asociación de conceptos atómicos, roles atómicos, e individuos para objetos del dominio de interpretación. Un ejemplo de (I) se presenta en la Tabla 4.4.

Tabla 4.4 Ejemplo de interpretación de reglas de la base de conocimiento

Sea $\mathrm{AC} = \{\mathrm{Cuenta}, \mathrm{Cliente}\}, \mathrm{AR} = \{\mathrm{esAbiertaPor}\},$ y $\mathrm{IV} = \{\mathrm{bib}, \mathrm{rub}\}.$

Sea I = (Δ^I, $.^I$) donde

Δ^{I}	=	{Banco, Biblioteca, Manufactura, Club, Rubí, Ramiro, José, Guadalupe}
$\mathrm{Cuenta}^{\mathrm{I}}$	=	{Banco, Biblioteca, Manufactura, Club}
$\mathrm{Cliente}^{\mathrm{I}}$	=	{Ramiro, Rubí Guadalupe}
$\mathrm{esAbiertaPor}^{\mathrm{I}}$	=	{(Biblioteca, Rubí)}
$\mathrm{bib}^{\mathrm{I}}$	=	Biblioteca
$\mathrm{rub}^{\mathrm{I}}$	=	Rubí

Entonces I es una (posible) Interpretación DL con respecto a KB (ver Figura 4.13).

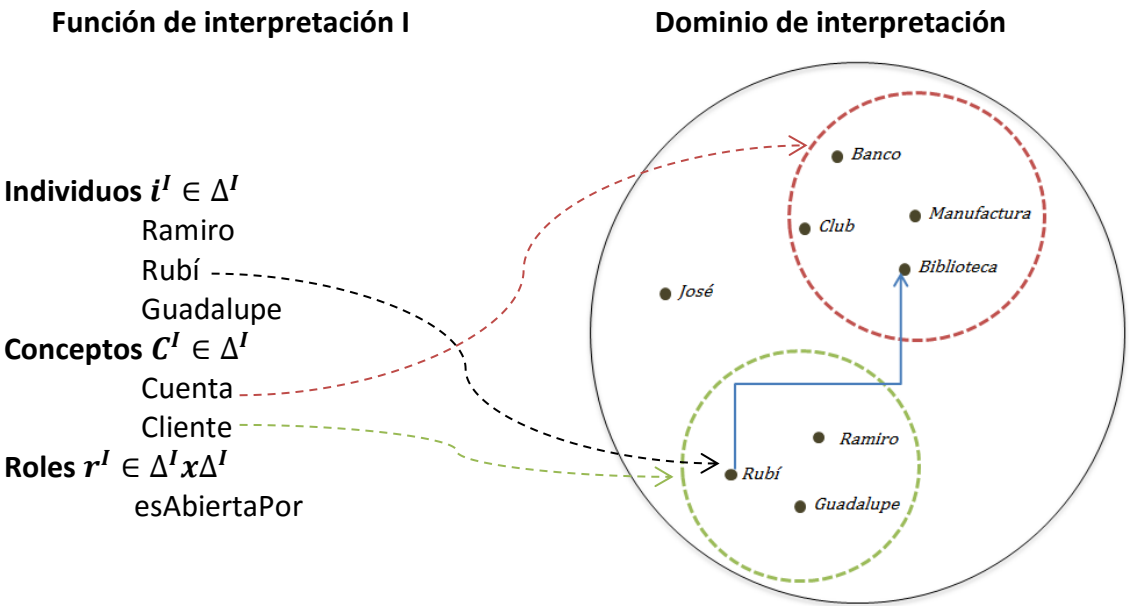


Figura 4.13. Función y dominio de interpretación para el dominio de estudio

Con el ejemplo presentado en Tabla 4.4 se comprueba si una base de conocimiento es consistente para un dominio de estudio, es decir, si existe un modelo de interpretación que satisface los axiomas definidos para la base de conocimiento (ver Anexo B, Definición 1.28).

4.3.5.2 Estructuración del modelo dominio.

En esta actividad se lleva a cabo la estructuración semántica del modelo de estudio permitiendo visualizar un diagrama definido de acuerdo a las reglas determinadas en las fases anteriores, (ver Figura 4.14).

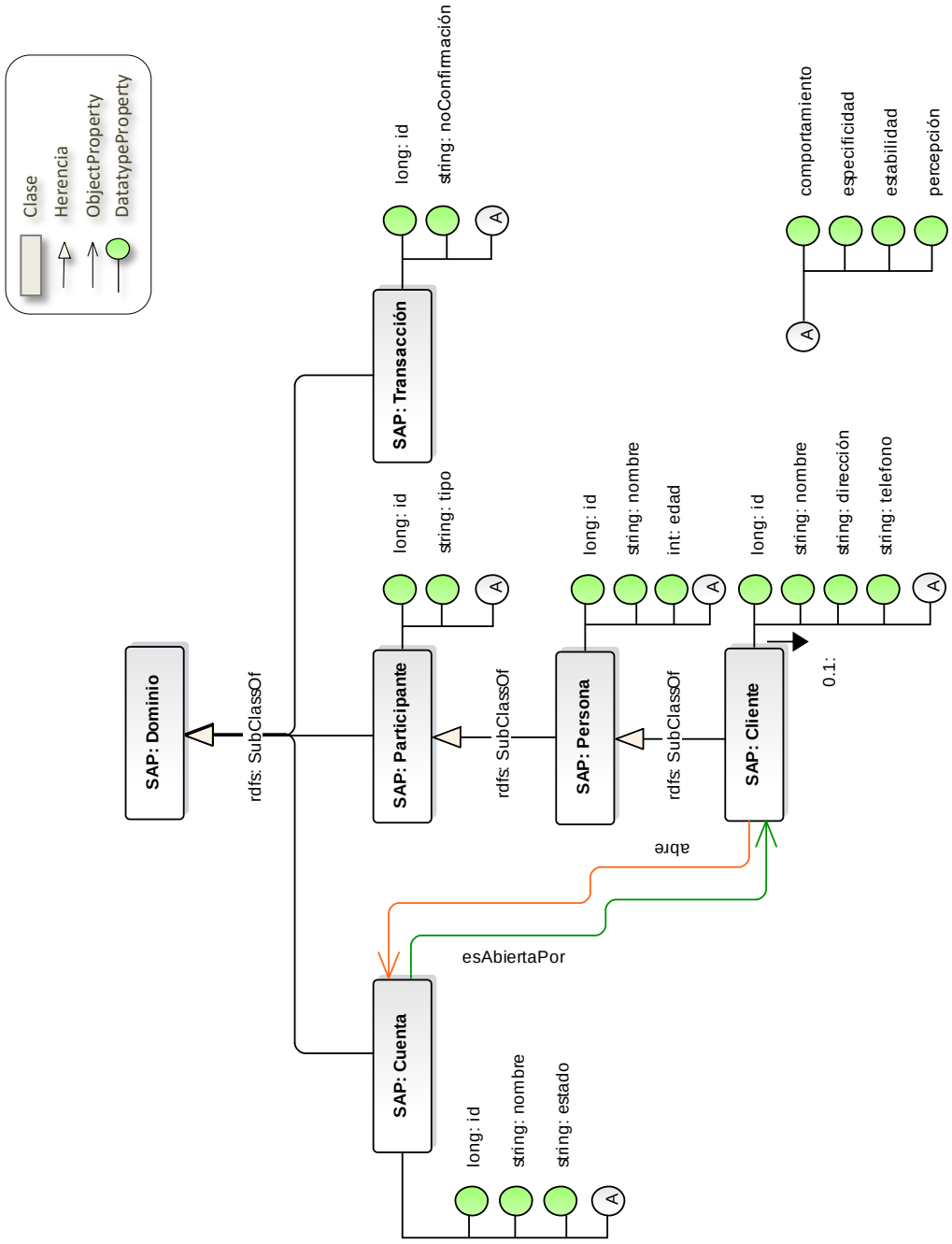


Figura 4.14. Estructuración semántica del modelo de estudio

En este capítulo se desarrolló el Metodología de Validación de Consistencia de Patrones de Análisis (MEVASE), la cual permite la formalización de patrones de análisis a través de lógica matemática, es decir, utiliza como lenguaje formal la “Lógica Descriptiva”. En el Capítulo 5 se muestra un ejemplo de la validación de un Patrón de análisis siguiendo la metodología propuesta.

Capítulo 5. Construcción y Validación de la Ontología del Patrón de Análisis de Cuenta

En este capítulo se presenta la documentación y los pasos que se llevaron a cabo para la construcción de la ontología del patrón de análisis para cuenta a través de la Metodología para la Construcción de Ontologías de Patrones de Análisis (MECOPA) la cual incluye la nueva fase de validación para la creación de ontologías de patrones de análisis.

5.1 Construcción de la ontología

En esta sección se encuentra detallado el proceso de creación de la ontología del patrón de análisis de cuenta, a través del seguimiento de la metodología MECOPA [2]. El orden en el que se llevan a cabo las fases que constituyen el ciclo de vida de una ontología, según la metodología MECOPA son las siguientes: 1) Refinamiento del patrón de análisis; 2) Refinamiento de los requerimientos; 3) Determinación del ámbito de la ontología (Especificación); 4) Búsqueda de ontologías relacionadas (Integración); 5) Identificación de términos del dominio (Conceptualización); 6) Especificación de conceptos y relaciones (Formalización); 7) Definición de restricciones; 8) Verificación y validación de la base de conocimiento (Semántica); 9) Instanciación de la ontología. A continuación se presenta la documentación de la ontología OPPAE [17] correspondiente a cada fase de MECOPA:

5.1.1 Refinamiento del patrón de análisis

Primera fase de MECOPA. En esta fase se describió el patrón de análisis de cuenta, asignando valores a cada una de las propiedades de las clases de la ontología OPPAE. En la Figura 5.1 se muestra la interfaz de la herramienta Protégé, donde se desarrolló la OPPAE, en la pestaña de “Individuos” y se muestran los valores que fueron asignados a cada una de las propiedades de la clase Nombre (por mencionar un ejemplo), los valores de las propiedades de las demás clases se pueden observar seleccionando cada uno de los elementos que se encuentran en la ventana “Individuos” (Instances , ver Anexo A).

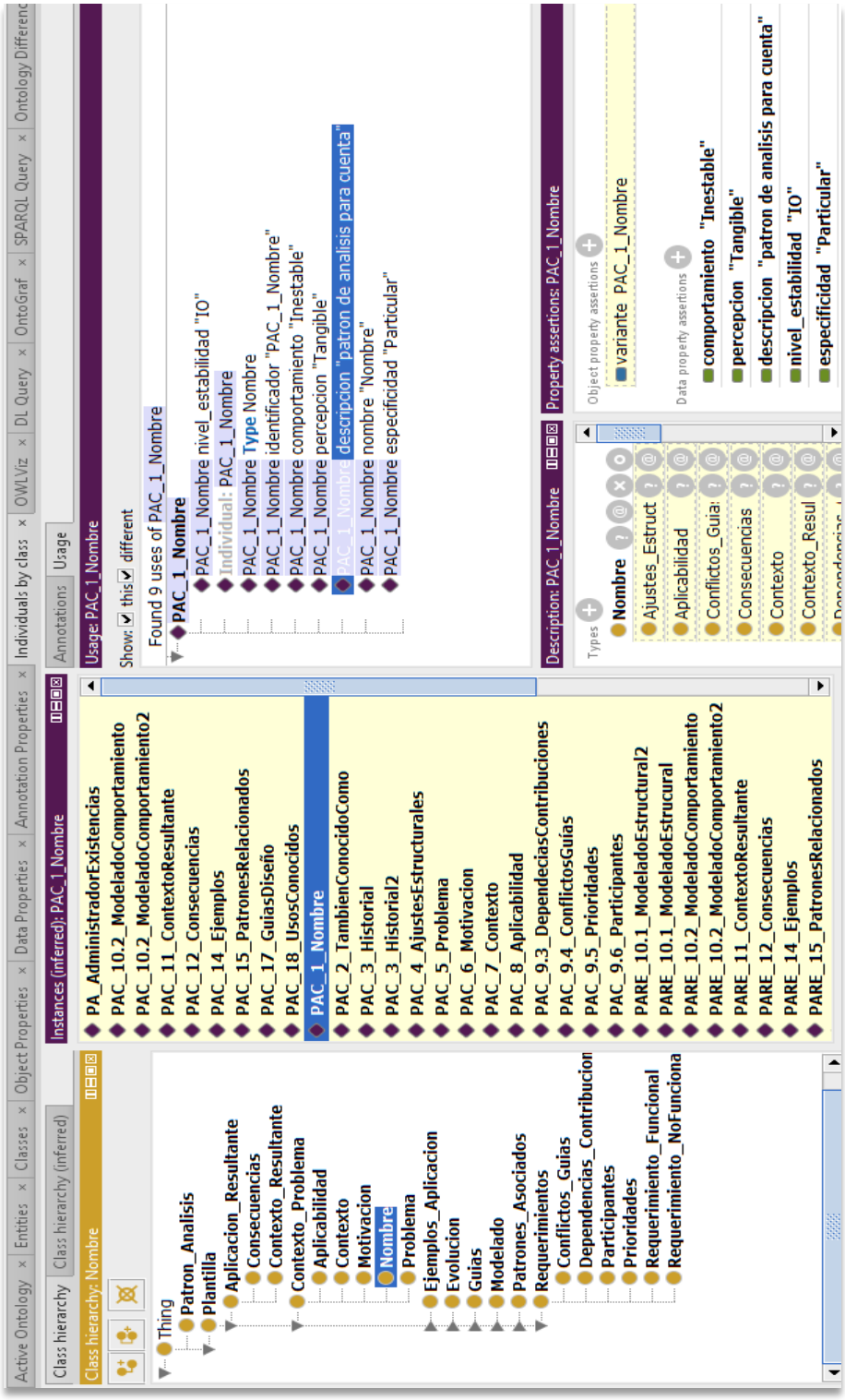


Figura 5.1. Individuos de clases de OPPAE en Protégé

5.1.2 Refinamiento de los requerimientos

Para llevar a cabo esta fase, nos dirijimos al apartado número 9 de la descripción del patrón de cuenta, donde se encuentran los requerimientos de forma textual. Primeramente se hizo una separación de las acciones más importantes que se deben llevar a cabo para que se cumpla el requerimiento (ejemplo: "Abrir una cuenta"), mediante una representación jeráquica (ver sección 5.1.2.2.1 Representación jerárquica) este proceso se hace para cada requerimiento. Posteriormente, se procedió a la elaboración de los diagramas GRL de cada uno de los requerimientos, en la sección 5.1.2.1.2 Diagrama GRL se muestra un ejemplo.

Luego se hizo una lista de los posibles componentes (equipos, actores, objetos, etc.) que intervienen en el cumplimiento de cada requerimiento, así como de las posibles responsabilidades o tareas de cada componente (ver sección 5.1.2.1.3 Tabla de responsabilidades). Culminando con el diseño de los diagramas UCM de cada uno de los requerimientos (ver sección 5.1.2.1.4 Diagrama UCM), a continuación, se muestran los resultados del refinamiento de requerimientos:

5.1.2.1 Requerimiento 1. Hacer una reservación

En la Figura 5.2 y Figura 5.3 se presentan los diagramas GRL y UCM respectivamente, que describen las metas y el comportamiento del requerimiento: abrir una cuenta:

5.1.2.1.1 Representación jerárquica

1. Abrir una cuenta

> **1.1 Registrar información:** Actividad que realiza el empleado para registrar la información del cliente.
>
> **1.2 Verificar crédito:** Proceso que efectúa un empleado, en el cual se puede saber si un cliente: tiene buen crédito, tiene mal crédito, o no tienen historial de crédito. El resultado de llevar a cabo este proceso influye en el éxito de la creación de la cuenta.
>
> **1.3 Crear cuenta:** Actividad que realiza el empleado para activar la cuenta de un cliente.

1.4 Proporcionar tarjeta: Proceso que efectúa un empleado, el cual consiste en otorgar una Tarjeta a un cliente. Esta, la utiliza el cliente como medio para realizar ciertas operaciones autorizadas.

5.1.2.1.2 Diagrama GRL

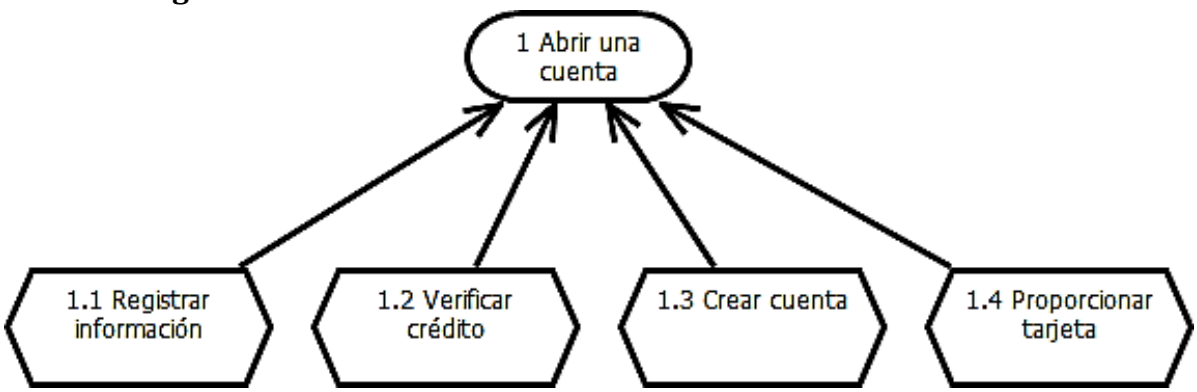


Figura 5.2. Diagrama GRL del requerimiento 1. Abrir una cuenta

5.1.2.1.3 Tabla de responsabilidades

Tabla 5.1. Actores objetos y responsabilidades del requerimiento 1

Actores	Responsabilidades
Cliente	Abrir cuenta
Empleado	Agregar cliente, Verificar Crédito, Regresar número de cuenta
Objetos	
Cuenta	Crear cuenta, Regresar número de cuenta
Sucursal	Agregar cuenta, Crear tarjeta

5.1.2.1.4 Diagrama UCM

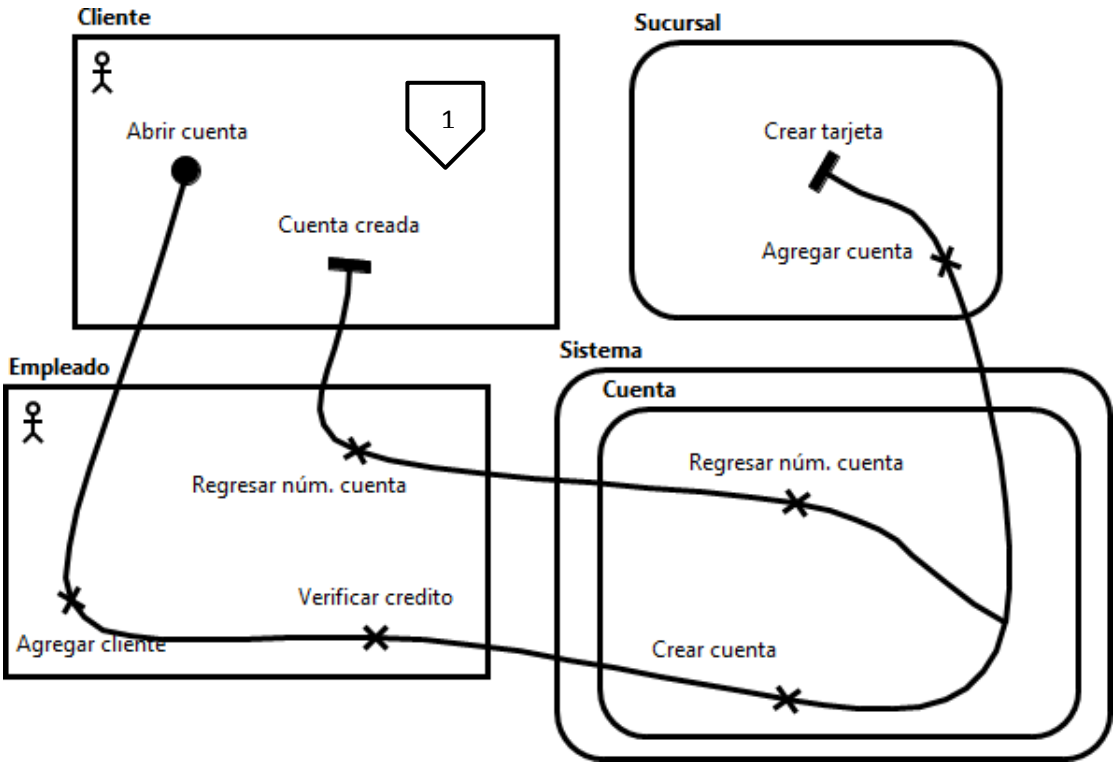


Figura 5.3. Diagrama UCM del requerimiento 1. Abrir una cuenta

5.1.2.2 Requerimiento 2. Realizar una transacción

Los siguientes diagramas GRL y UCM describen las metas y el comportamiento del requerimiento usar para realizar una transacción.

5.1.2.2.1 Representación jerárquica

2. Realizar una transacción

2.1 Usar tarjeta: El cliente utiliza su tarjeta emitida por la sucursal donde previamente fue creada su cuenta.

2.1.1 Validar tarjeta: El cliente valida su tarjeta con su NIP (Número de Identificación Personal) otorgada por la sucursal.

2.2 Tipo cuenta: Cuenta personal utilizada por el cliente en la cual se verá reflejada las diferentes operaciones realizadas para la cuenta elegida

2.3 Sucursal: Sucursal a cargo de la cuenta en la cual se determinará el éxito de la operación realizada por el cliente.

2.3.1 Aspectos históricos: Historial de transacciones realizadas en un determinado periodo de tiempo.

5.1.2.2.2 Diagrama GRL

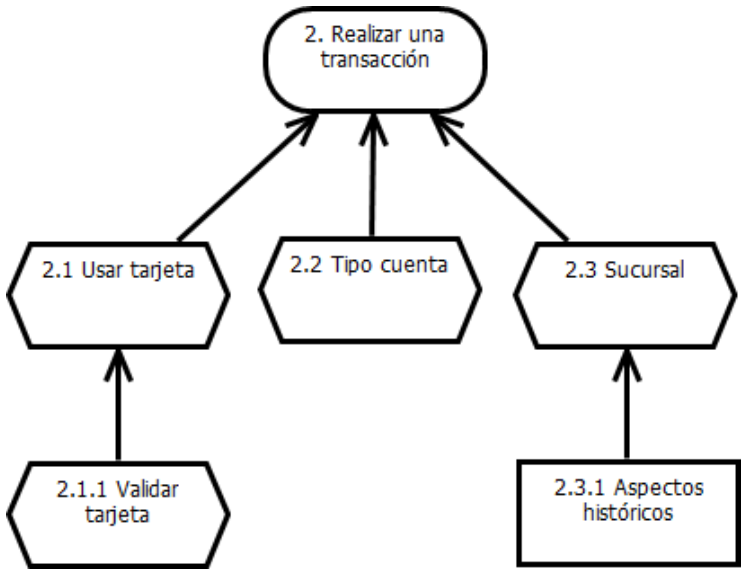


Figura 5.4. Diagrama GRL del requerimiento 2. Realizar una transacción

5.1.2.2.3 Tabla de responsabilidades

Tabla 5.2. Actores objetos y responsabilidades del requerimiento 2

Actores	Responsabilidades
Cliente	Realizar transacción, Usar tarjeta
Empleado	Validar tarjeta, Confirmar, Retornar error
Objetos	
Cuenta	Elegir cuenta, Ejecutar transacción, Confirmación, Retornar error
Transacción	Exitosa, Rechazada, Guardar historial

5.1.2.2.4 Diagrama UCM

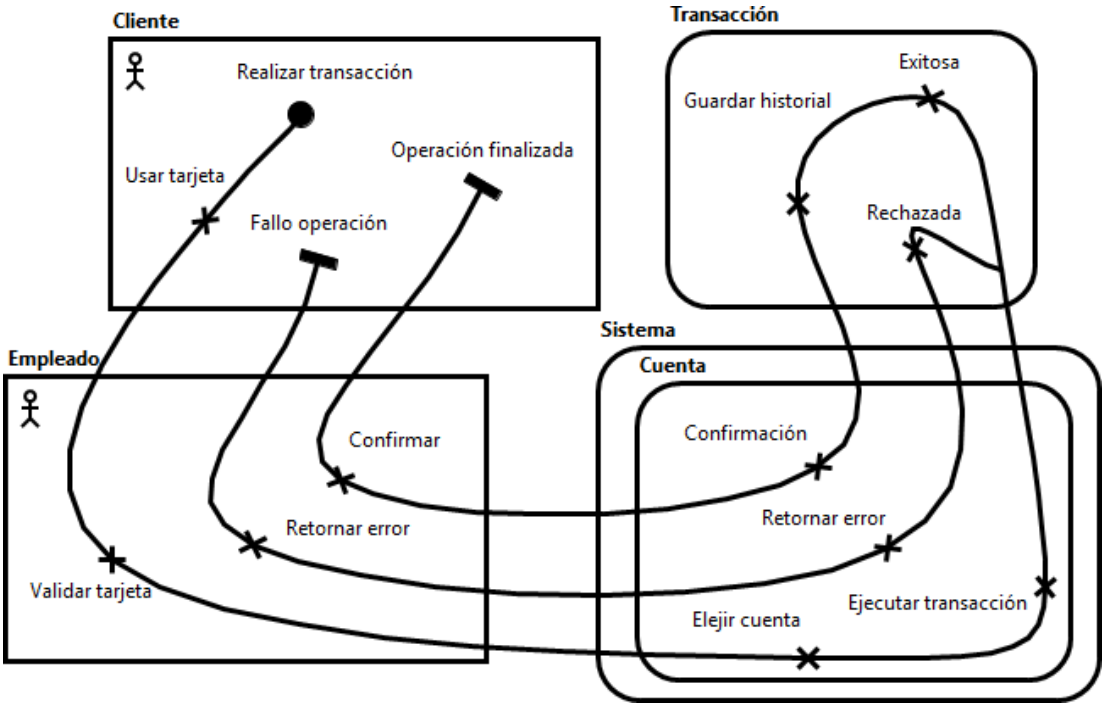


Figura 5.5. Diagrama UCM del requerimiento 2. Realizar una transacción

5.1.2.3 Requerimiento 4. Cerrar una cuenta

Los siguientes diagramas GRL y UCM describen las metas y el comportamiento del requerimiento cancelar una reservación.

5.1.2.3.1 Representación jerárquica

4. Cerrar una cuenta

4.1 Cerrar cuenta existente: Proceso para cancelar una cuenta previamente solicitada por un cliente.

4.1.1 Verificar cuenta: Verificar que no existen compromisos pendientes con la sucursal. Mientras haya compromisos pendientes de liquidar, la cuenta no se podrá dar de baja.

4.2 Aplicar políticas: Medidas que se aplican según sea el caso de la sucursal para determinar los efectos de la cancelación.

4.3 Archivo histórico: La información de la cuenta cancelada es transferido a un archivo histórico para tener un respaldo de los diferentes tipos de transacciones realizadas mientras la cuenta estuvo activa.

5.1.2.3.2 Diagrama GRL

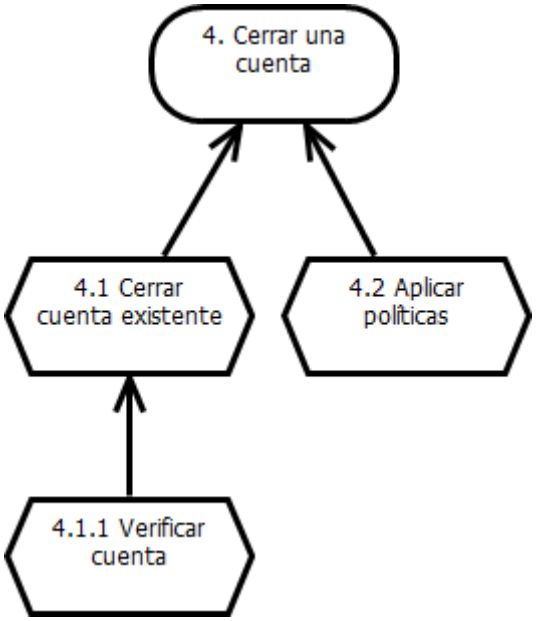


Figura 5.6. Diagrama GRL del requerimiento 4. Cerrar una cuenta

5.1.2.3.3 Tabla de responsabilidades

Tabla 5.3. Actores objetos y responsabilidades del requerimiento 4. Cerrar una cuenta

Actores	Responsabilidades
Cliente	Solicitar cancelación, Confirmar
Empleado	Verificar cuenta, Aplicar políticas, Cancelar
Objetos	
Cuenta	Archivo histórico
Sucursal	Cuenta cancelada

5.1.2.3.4 Diagrama UCM

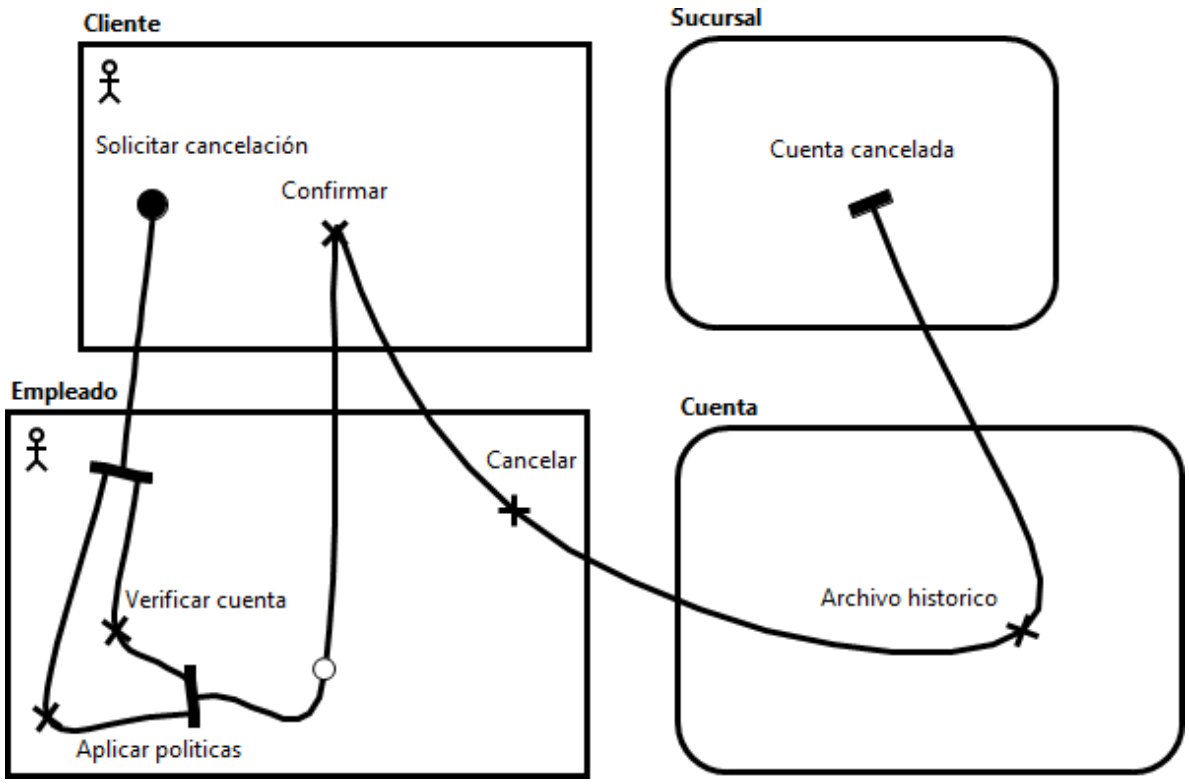


Figura 5.7. Diagrama UCM del requerimiento 4. Cerrar una cuenta

Tanto para el diseño de los diagramas GRL como para los diagramas UCM se utilizó el ambiente de desarrollo Eclipse con el plugin **jUCMNav v5.5.0**.

5.1.2.4 Requerimientos funcionales

Tabla 5.4. Tabla de dependencias y prioridades

Requerimiento	Dependencias	Prioridad
1. Abrir una cuenta		Alta
2. Realizar una transacción	R1	Media
3. Cerrar una cuenta	R1	Baja

5.1.3 Determinación del ámbito de la ontología (Especificación)

Este apartado corresponde a la tercera fase de MECOPA. Aquí se definen los usuarios, el dominio, el propósito y el tipo de ontología, así como el cuestionario de competencias:

5.1.3.1 Usuarios

Los usuarios finales previstos para la OPAC son:

- Responsables de análisis del dominio de cuentas.
- Desarrolladores de aplicaciones para patrones de análisis.
- Investigadores y expertos del dominio, para retroalimentación y mejora continua.

5.1.3.2 Dominio

La OPAC se desarrolló en el dominio de cuenta.

5.1.3.3 Propósito

El propósito fundamental de la **OPAC** es representar, organizar, formalizar y estandarizar el conocimiento relacionado con el dominio del patrón de análisis para cuenta.

5.1.3.4 Tipo de ontología

La OPAC es una ontología de dominio.

5.1.3.5 Cuestionarios de competencias

A continuación, se muestran los requerimientos que se deberán satisfacer usando la OPAC, la cual será capaz de contestar el siguiente cuestionario:

1. ¿Qué se necesita para crear una cuenta?
2. ¿Quiénes pueden ser clientes?
3. ¿Qué actores están involucrados en una transacción?
4. ¿Qué datos se necesitan para hacer una transacción?
5. ¿Dónde se registra (consulta) una transacción?

6. ¿Qué se requiere para cancelar una cuenta?
7. ¿Cuál es la diferencia una cuenta de banco y una cuenta de biblioteca?
8. ¿Cuáles son los estados de una cuenta?

5.1.4 Búsqueda de ontologías relacionadas (Integración)

A continuación, se muestra un análisis de las ontologías que proponen otros autores en temas relacionados a transacciones en dominio de contabilidad.

Ontología de Transacciones Contables

En [27] se propone la Ontología de Transacciones Contables la cual fue desarrollada a través de un proceso interactivo, en colaboración con la ICT y expertos en contabilidad. Esta ontología tiene como objetivo ayudar el comprender los tipos de transacciones que se pueden realizar en el dominio de contabilidad. Es importante mencionar que el autor define una transacción contable como un evento de negocio que tiene un impacto monetario sobre los estados financieros de una empresa. Las clases que se definen para esta ontología son las siguientes (ver Figura 5.8):

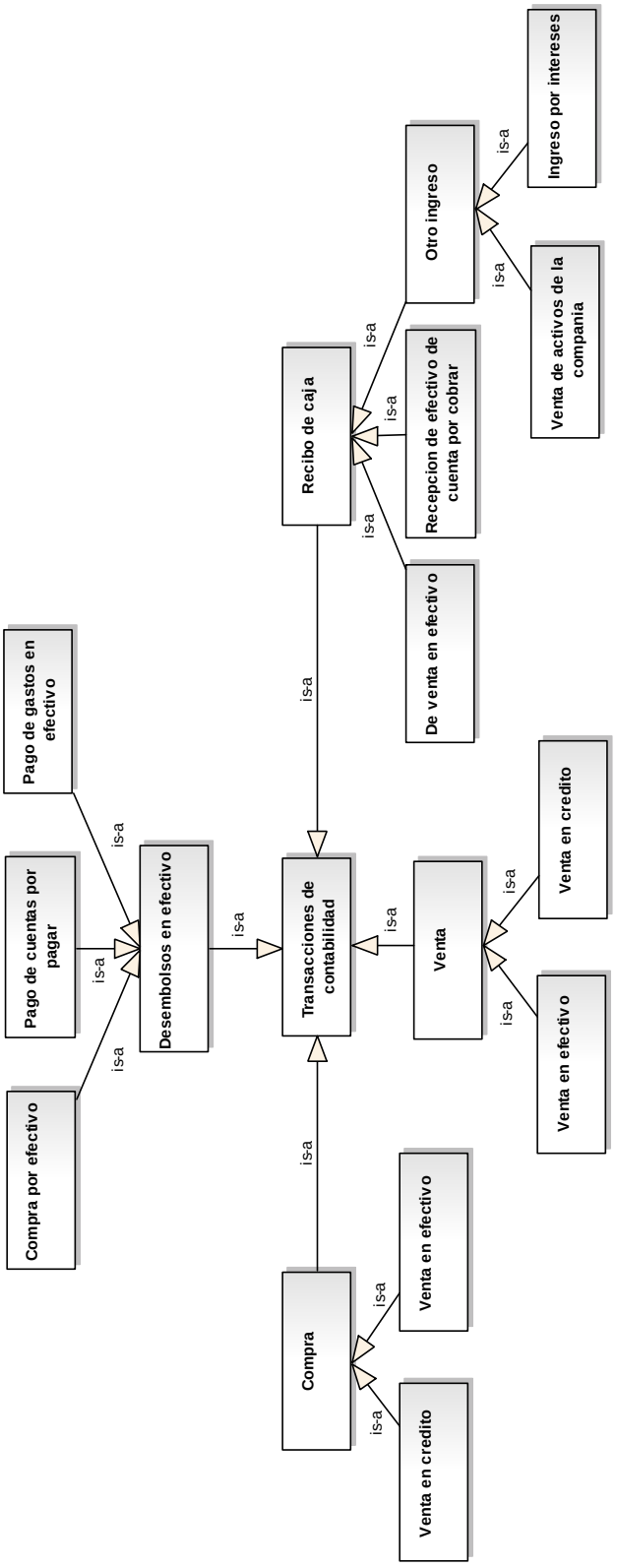


Figura 5.8. Clases de la Ontología de Transacciones Contable.

Tabla 5.5. Tabla comparativa

Ontología	Metodología	Dominio
Ontología de Transacciones Contables	No definido por el autor	Contabilidad
Ontología de Patrón de Análisis de Cuenta	MECOPA	Contabilidad

En la Tabla 5.5 se hace una comparativa de las ontologías enfocadas a los tipos de transacciones en una cuenta. En la Tabla 5.6 se muestran los elementos (o clases) comunes con la ontología OPAC (Ontología de Patrón de Análisis de Cuenta). Tanto la ontología 1 como la ontología número 2 de la Tabla 5.6 se dedican a la administración de una cuenta, sin embargo, el número 1 tiene un menor alcance, ya que incorpora más actividades para a la administración de cuentas, y la ontología número 2 es más general, no se enfoca en un tipo de cuenta en específico, sino que habla de una manera global.

Tabla 5.6. Comparativa de Ontologías

No.	Ontología	Clases comunes a OPAC	SSM	Entidad
2	Ontología de Transacciones Contables	Cuenta, Transacción	No	Asiento de avión
3	Ontología de Patrón de Análisis de Cuenta	Cuenta, Transacción	Si	Recurso general

5.1.5 Identificación de términos del dominio (Conceptualización)

5.1.5.1 Clases

Una clase es un término que se utiliza para definir una categoría que incluye un conjunto de objetos (individuos) con características particulares, o incluso una categoría que englobe a más categorías. De acuerdo al ejemplo utilizado en este capítulo, una clase podría ser Cliente. Cliente es una categoría que puede englobar más categorías como alumnos y profesores, de las cuales, a su vez puede haber alumnos de computación, alumnos de mecánica, y de igual forma profesores.

Las clases identificadas se presentan a continuación:

1. Cliente
2. Cuenta
3. Empleado
4. Institución
5. Transacción
6. Participante
7. Persona

5.1.5.2 Propiedades

Una propiedad básicamente es una característica que posee un individuo y que lo hace distinguirse de los demás. Aunque en un sentido estricto es una relación entre un individuo de una clase y un esquema XML.

Las propiedades identificadas para cada clase de la sección anterior son las siguientes:

1. Cliente. id cliente, nombre, dirección, teléfono, comportamiento, especificidad, nivel de estabilidad, percepción
2. Cuenta. id cuenta, nombre, estado de cuenta, comportamiento, especificidad, nivel de estabilidad, percepción
3. Empleado. id empleado , nombre, comportamiento, especificidad, nivel de estabilidad, percepción
4. Institución: id institución, nombre, giro, comportamiento, especificidad, nivel de estabilidad, percepción
5. Transacción. id transacción, Fecha de Realización, no de confirmación, comportamiento, especificidad, nivel de estabilidad, percepción
6. Participante: id participante, tipo, comportamiento, especificidad, nivel de estabilidad, percepción
7. Persona: id persona, nombre, edad, comportamiento, especificidad, nivel de estabilidad, percepción

5.1.6 Especificación de conceptos y relaciones (Formalización)

5.1.6.1 Conceptos

En el dominio de las ontologías, un concepto es sinónimo de clase, en este apartado se lleva a cabo la especificación de las clases encontradas en la sección 5.1.5.1 Clases:

- Cuenta
- Participante
    a. Institución
    b. Persona
        i. Cliente
        ii. Empleado
- Transacción

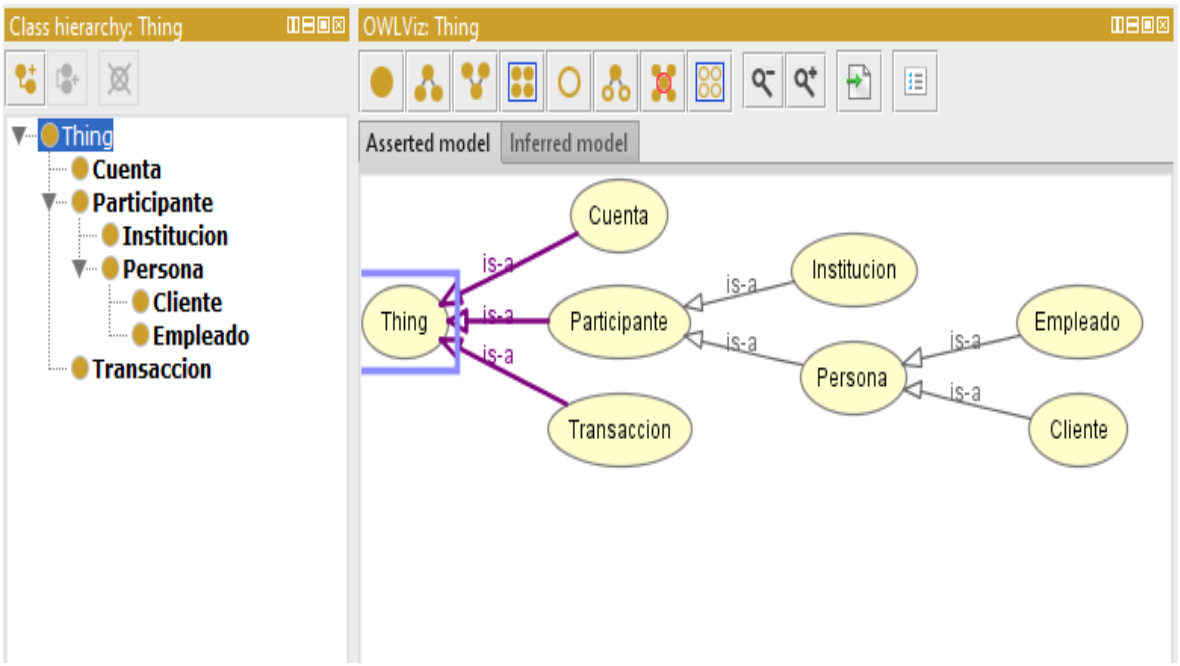


Figura 5.9. Representación de clases de OPAC en Protégé

5.1.6.2 Relaciones

Una relación es una acción que lleva a cabo un individuo y que para su realización se requiere de otro individuo, es decir una acción que conecta dos individuos.

Tabla 5.7. Relaciones entre clases de la ontología PAC

Dominio	Relación	Rango	Tipo
Cliente	abre	Cuenta	Inversa funcional
Empleado	crea	Cuenta	N/A
Cuenta	esAbiertaPor	Cliente	Funcional
Cuenta	esCreadaPor	Empleado	N/A
Cliente	realiza	Transacción	Funcional
Transacción	esRealizadaPor	Cliente	Inversa funcional
Empleado	laboraEn	Institución	Funcional
Institución	tieneDatosDe	Empleado Cliente Transacción Cuenta	N/A

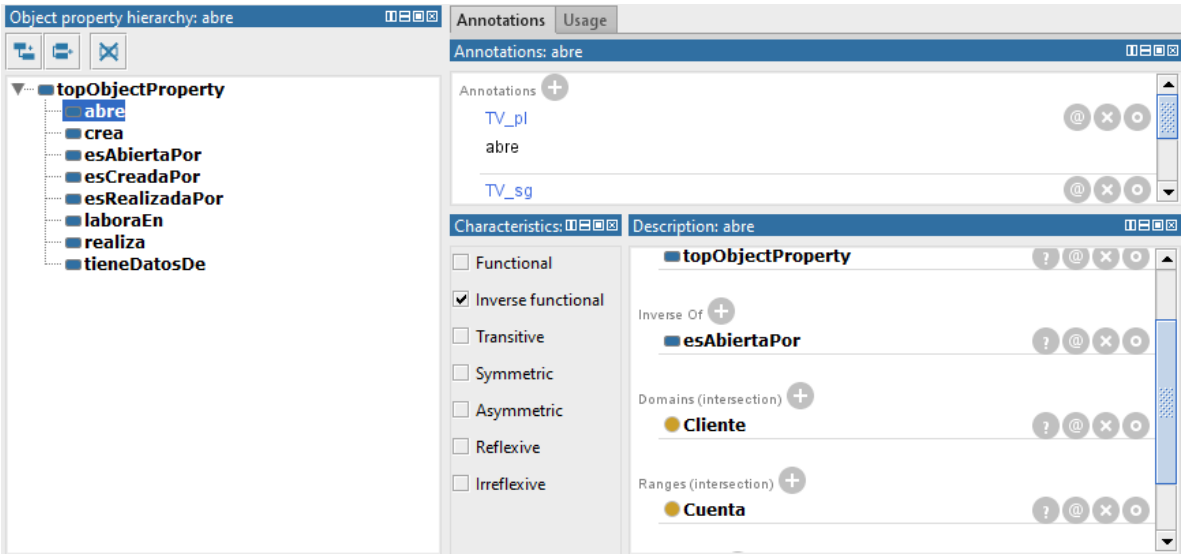


Figura 5.10. Representación de relaciones de OPAC en Protégé

5.1.6.3 Propiedades

En este punto se acomodan las propiedades identificadas en 5.1.5.2 Propiedades, mediante una jerarquía, tal cual serán definidas en la especificación con la herramienta Protégé.

- Cuenta: idCuenta, nombre, estadoCuenta, comportamiento, especificidad, nivelEstabilidad, percepción
- Participante: idParticipante, tipo, comportamiento, especificidad, nivelEstabilidad, percepción
    a. Persona: idPersona, nombre, edad, comportamiento, especificidad, nivelEstabilidad, percepción
        i. Cliente: idCliente, dirección, teléfono, comportamiento, especificidad, nivelEstabilidad, percepción
        ii. Empleado: idEmpleado, comportamiento, especificidad, nivelEstabilidad, percepción
    b. Institución: idInstitución, nombre, giro, comportamiento, especificidad, nivelEstabilidad, percepción
- Transacción. idTransacción, noConfirmación, fechaRealización, comportamiento, especificidad, nivelEstabilidad, percepción

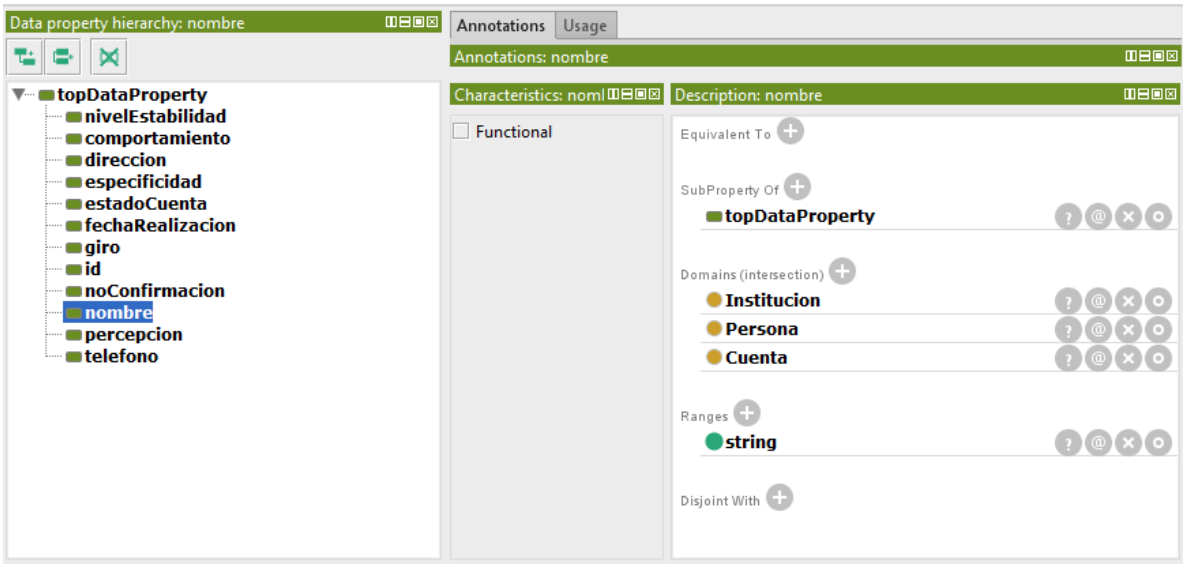


Figura 5.11. Representación de propiedades de OPAC en Protégé

5.1.7 Definición de restricciones

En esta fase se definieron dos tipos de restricciones:

5.1.7.1 Restricciones existenciales

Cliente *solicita* *some* Cuenta
Cliente *realiza* *some* Transacción

5.1.7.2 Restricciones universales

Cuenta esSolcitadoPor *only* Cliente
Transacción *esRealizadaPor* *only* Cliente
Empleado *laboraEn* *only* Institución

En la Figura 5.12 se muestran las restricciones de la clase Cliente:

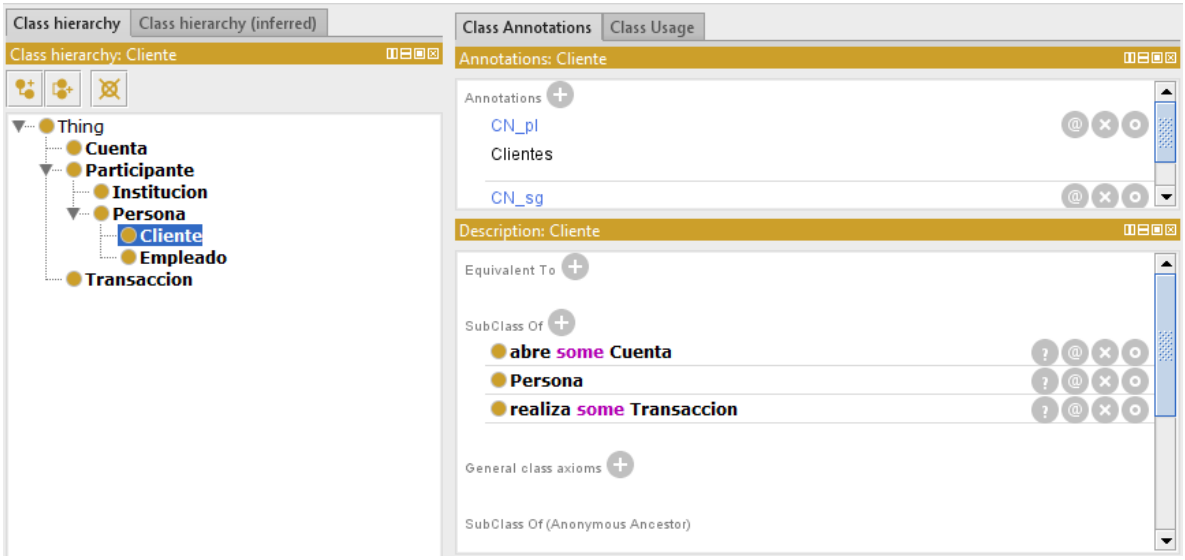


Figura 5.12. Restricciones de clases de OPAC en Protégé

5.1.8 Verificación y validación de la base de conocimiento (Semántica)

En esta fase se lleva a cabo la formalización del patrón de análisis cuenta de manera implícita a través de lógica matemática, es decir, se utiliza un lenguaje formal como lo es la "Lógica Descriptiva".

5.1.8.1 Descripción básica del lenguaje

La notación "SHOIN + dominios concretos" la cual es utilizada por algunos sistemas de lógicas de descripción, entre los cuales se encuentra el editor de ontologías protégé, será el tipo de expresividad DL utilizada para definir la "representación del conocimiento" en un lenguaje basado en: a) Símbolos dependientes de aplicación, b) Símbolos independientes de aplicación.

5.1.8.1.1 Clasificación de símbolos dependiente de aplicación

El conjunto de símbolos dependientes de aplicación (AS por el inglés application dependent symbols), los cuales son objetos y relaciones del mundo real determinados por el dominio de estudio se obtienen de las secciones: 1) 5.1.5 Identificación de términos del dominio (Conceptualización); 2) 5.1.6 Especificación de conceptos y relaciones (Formalización); 3) 5.1.7 Definición de restricciones para la aplicación dependiente de símbolos. En la Tabla 5.8 se define la clasificación para cada subconjunto de AS: 1) Conjunto de conceptos atómicos (AC por el inglés atomic concepts) representando clases de objetos; 2) Conjunto de roles atómicos (AR por el inglés atomic roles) representando relaciones binarias sobre objetos; 3) Conjunto de individuos (IV por el inglés individuals) representando objetos individuales de la aplicación de dominio.

Tabla 5.8. Ejemplo de símbolos dependientes de aplicación

$\boldsymbol{AC}$	=	{ Cuenta, Participante, Transacción, Organización, Persona, Cliente, Empleado }
$\boldsymbol{AR}$	=	{ abre, crea, esAbiertaPor, esCreadaPor, realiza, esRealizadaPor, laboraEn, tieneDatosDe }
$\boldsymbol{IV}$	=	{Banco, Biblioteca, Manufactura, Tienda de Video, Club, Hotel }
$\boldsymbol{AS}$	=	$AC \cup AR \cup IV$
	=	{ Cuenta, Participante, Transacción, Organización, Persona, Cliente, Empleado, *a*bre, crea, esAbiertaPor, esCreadaPor, realiza, esRealizadaPor, laboraEn, tieneDatosDe, Banco, Biblioteca, Manufactura, Tienda de video, Club, Hotel }

5.1.8.1.2 Símbolos (In) dependiente de aplicación

Los símbolos independiente de aplicación (A(In)S por el inglés application independent symbols) que forman el conjunto de conectores definidos por el tipo de notación de lógica descriptiva (DL) elegida se presenta en la Tabla 5.9.

Tabla 5.9. Símbolos (In) dependientes de aplicación para la notación SHOIN + dominios concretos

$$
\begin{aligned}
A(In)S &= \{ALC\} \\
&= \{\, A \text{ (concepto atómico)} \\
&\quad \top \text{ (concepto universal)} \\
&\quad \bot \text{ (concepto vacío)} \\
&\quad \neg \text{ (negación o complemento)} \\
&\quad \sqcap \text{ (conjunción o intersección)} \\
&\quad \sqcup \text{ (disyunción o unión)} \\
&\quad \forall \text{ (cuantificador universal o restricción de valor)} \\
&\quad \exists \text{ (Cuantificador existencial) } \}
\end{aligned}
$$

5.1.8.2 Definición terminológica

En esta fase se define el componente intencional (IC por el inglés intentional component) [25].

5.1.8.2.1 Componente intencional

El conjunto finito de axiomas terminológicos (TA por el inglés terminological axioms) los cuales permiten representar el nivel de conocimiento sobre el dominio de estudio, se determinan a través de la sección 5.1.6 Especificación de conceptos y relaciones (Formalización) utilizando la notación de la lógica descriptiva SHOIN (D). Estos axiomas permiten definir las relaciones generales entre conceptos (o clases), que se mantiene en todo momento, para todos los posibles objetos del domino de conocimiento.

5.1.8.2.1.1 Implicaciones conceptuales

Los axiomas de implicación conceptual definidos para el patrón de análisis de cuenta (AAP por el inglés Account Analysis Pattern), permiten expresar el lenguaje del dominio (ver Figura 5.13) determinando el contexto aplicable de estas reglas.

TA = {

T	
Cuenta ⊑ T	**(1)**
Participante ⊑ T	**(2)**
Transacción ⊑ T	**(3)**
Institución ⊑ Participante	**(4)**
Persona ⊑ Participante	**(5)**
Cliente ⊑ Persona	**(6)**
Empleado ⊑ Persona	**(7)**

}

Figura 5.13. Axiomas terminológicos de AAP

5.1.8.2.1.2 Equivalencias conceptuales.

Los axiomas de equivalencia conceptual representan a través de un concepto definido un conjunto de reglas determinadas para un dominio de estudio. Para este trabajo de investigación no se definieron estos axiomas tratando que las reglas definidas para la formalización del patrón de análisis de cuenta sean genéricas y aplicables a los contextos determinados en el Anexo A: punto 16.

5.1.8.2.1.3 Creación del modelado conceptual

Los axiomas obtenidos en la sección 5.1.8.2.1.1 Implicaciones conceptuales permiten demostrar la relación jerarquía en conceptos con respecto a un un TA (axioma terminológico), definiendo un grafo acíclico el cual valida la estructura para el patrón de estudio (ver Figura 5.14).

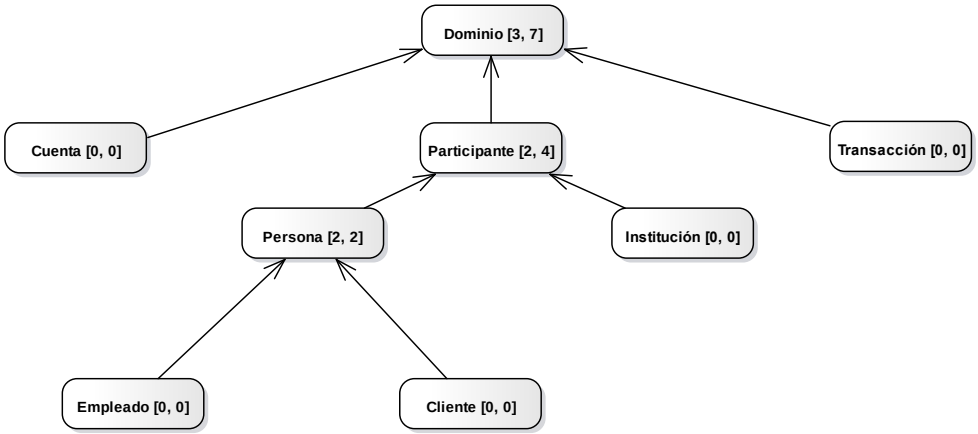


Figura 5.14. Grafo aciclico para el domino de estudio

5.1.8.3 Descripción del universo

En esta fase se define el componente extensional (EC por el inglés extensional component) [25] en base a símbolos (in) dependientes de aplicación.

5.1.8.3.1 Definición de componente extensional

El conjunto finito de axiomas de aserción (AA por el inglés abox assertion) realiza dos tipos de afirmaciones: a) Afirmaciones sobre conceptos, b) Afirmaciones sobre relaciones, las cuales permiten validar el lenguaje del domino definido en los TA (axioma terminológico) en la sección 5.1.8.2.1.1 Implicaciones conceptuales.

5.1.8.3.1.1 Afirmaciones sobre conceptos

Las afirmaciones sobre conceptos indican la pertenencia de individuos sobre conceptos primitivos o definidos (ver Figura 5.15).

AA = {

Cuenta (Banco)	**(8)**
Cuenta (Biblioteca)	**(9)**
Cuenta (Manufactura)	**(10)**
Cuenta (Tienda de Video)	**(11)**
Cuenta (Club)	**(12)**
Cuenta (Hotel)	**(13)**

}

Figura 5.15. Afirmaciones sobre el concepto cuenta de AAP

5.1.8.3.1.2 Afirmaciones sobre roles

Las afirmaciones sobre roles que indican la existencia de relaciones binarias entre individuos del domino de estudio, estas se determinan a través de la sección 5.1.6 Especificación de conceptos y relaciones (Formalización) utilizando la notación de la lógica descriptiva SHOIN (D) (ver Figura 5.16).

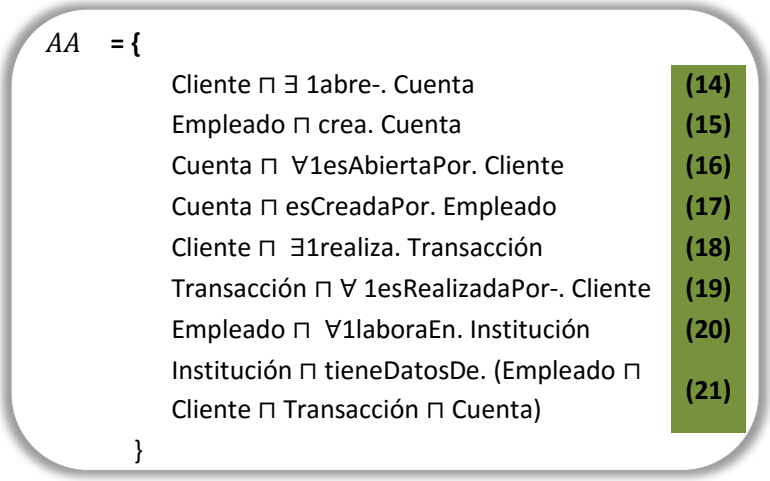


Figura 5.16. Afirmaciones sobre el concepto cuenta de AAP

5.1.8.3.1.3 Generación de equivalencias Sintaxis DL a Sentencias OWL

En esta actividad se realiza la equivalencia de la sintaxis DL definidas en las secciones 5.1.8.2 Definición terminológica y 5.1.8.3 Descripción del universo en sentencias OWL como declaraciones sobre: clases, individuos, propiedades; aserciones sobre: clases, individuos, propiedades, entre otros.

Declaración de clase

La declaración de clases a través de una sintaxis DL a sentencias OWL se realiza de la siguiente manera:

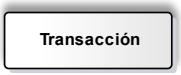
Sintaxis DL	
Cuenta ⊑ T	(1)
Sentencia OWL	
Declaration(Class(:Cuenta))	

Sintaxis DL

Participante ⊑ T (2)

Sentencia OWL

Declaration(Class(:Participante))



Sintaxis DL

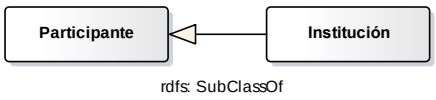
Transaccion ⊑ T (3)

Sentencia OWL

Declaration(Class(:Transacción))

Aserción Subclase

Las aserciones en subclases declaran que todos los individuos que pertenecen a una clase pertenecen también a otra clase. La equivalencia de la sintaxis DL a sentencias OWL se realiza de la siguiente manera:

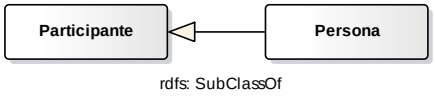


Sintaxis DL

Institución ⊑ Participante (4)

Sentencia OWL

SubClassOf(:Institucion :Participante)

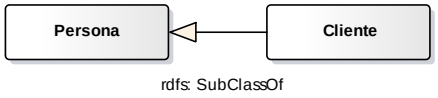


Sintaxis DL

Persona ⊑ Participante (5)

Sentencia OWL

SubClassOf(:Persona :Participante)

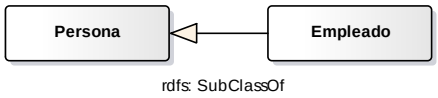


Sintaxis DL

Cliente ⊑ Persona (6)

Sentencia OWL

SubClassOf(:Cliente :Persona)



Sintaxis DL

Empleado ⊑ Persona (7)

Sentencia OWL

SubClassOf(:Empleado :Persona)

Restricciones y axiomas de propiedades de objetos

Las restricciones y axiomas de propiedades afectan directamente a la relación binaria entre objetos. La equivalencia de la sintaxis DL a sentencias OWL se realiza de la siguiente manera:

Sintaxis DL

Cuenta (Banco) (8)

Sentencia OWL

ClassAssertion(:Cuenta :Banco)

Sintaxis DL

Cuenta (Biblioteca) (9)

Sentencia OWL

ClassAssertion(:Cuenta :Biblioteca)

Sintaxis DL

Cuenta (Manufactura) (10)

Sentencia OWL

ClassAssertion(:Cuenta :Manufactura)

Sintaxis DL

Cuenta (Tienda de Video) (11)

Sentencia OWL

ClassAssertion(:Cuenta :Tienda de Video)

Sintaxis DL

Cuenta (Club) (12)

Sentencia OWL

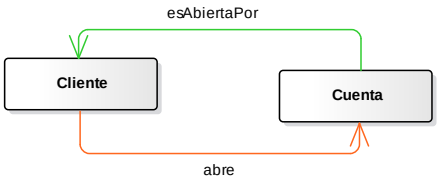
ClassAssertion(:Cuenta :Club)

Sintaxis DL

Cuenta (Hotel) (13)

Sentencia OWL

ClassAssertion(:Cuenta :Hotel)

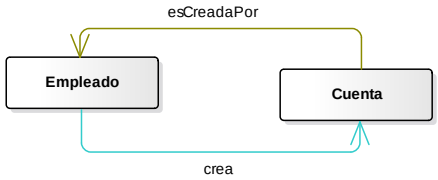


Sintaxis DL

Cliente ⊓ ∃ 1abre-. Cuenta (14)

Sentencia OWL

SubClassOf(:Cliente ObjectSomeValuesFrom(:abre :Cuenta))

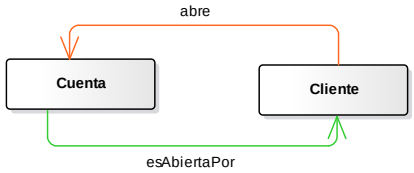


Sintaxis DL

Empleado ⊓ crea. Cuenta (15)

Sentencia OWL

ObjectPropertyDomain(:crea :Empleado)
ObjectPropertyRange(:crea :Cuenta)

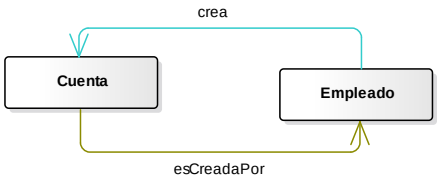


Sintaxis DL

Cuenta ⊓ ∀1esAbiertaPor. Cliente (16)

Sentencia OWL

SubClassOf(:Cuenta
ObjectAllValuesFrom(:esAbiertaPor :Cliente))

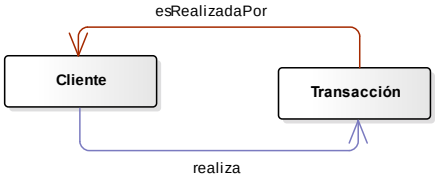


Sintaxis DL

Cuenta ⊓ esCreadaPor. Empleado (17)

Sentencia OWL

ObjectPropertyDomain(:esCreadaPor :Cuenta)
ObjectPropertyRange(:esCreadaPor :Empleado)

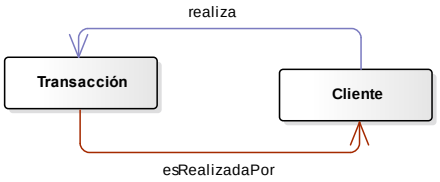


Sintaxis DL

Cliente ⊓ ∃1realiza. Transacción (18)

Sentencia OWL

SubClassOf(:Cliente ObjectSomeValuesFrom(:realiza
:Transaccion))

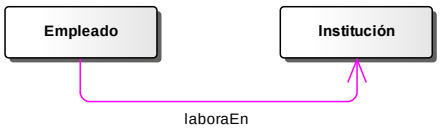


Sintaxis DL

Transacción ⊓ ∀ 1esRealizadaPor-. Cliente (19)

Sentencia OWL

SubClassOf(:Transaccion
ObjectAllValuesFrom(:esRealizadaPor :Cliente))

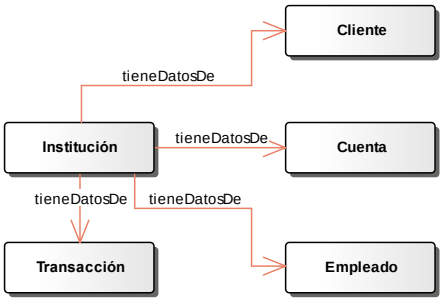


Sintaxis DL

Empleado ⊓ ∀1laboraEn. Institución (20)

Sentencia OWL

SubClassOf(:Empleado
ObjectAllValuesFrom(:laboraEn :Institucion))



Sintaxis DL

Institución ⊓ tieneDatosDe. (Empleado ⊓ Cliente ⊓ Transacción ⊓ Cuenta) (21)

Sentencia OWL

```
ObjectPropertyDomain(:tieneDatosDe :Institucion)
ObjectPropertyRange(:tieneDatosDe :Cliente)
ObjectPropertyRange(:tieneDatosDe :Cuenta)
ObjectPropertyRange(:tieneDatosDe :Empleado)
ObjectPropertyRange(:tieneDatosDe :Transaccion)
```

5.1.8.4 Definición del formalismo básico

La definición del formalismo básico establece una "base de conocimiento (KB por el inglés knowledge base)", el cual es una colección de información, incluida en componentes: a) intencional, b) extensional para el dominio de estudio.

5.1.8.4.1 Obtención de reglas de la base de conocimiento

Se obtienen las reglas (axiomas DL) creados en las sección 5.1.8.2 Definición terminológica1 y 5.1.8.3 Descripción del universo permitiendo definir la base de conocimiento para el dominio de estudio (ver Figura 5.17).

KB = {	
Cuenta ⊑ T	(1)
Participante ⊑ T	(2)
Transacción ⊑ T	(3)
Institución ⊑ Participante	(4)
Persona ⊑ Participante	(5)
Cliente ⊑ Persona	(6)
Empleado ⊑ Persona	(7)
Cliente ⊓ ∃ 1abre-. Cuenta	(8)
Empleado ⊓ crea. Cuenta	(9)
Cuenta ⊓ ∀1esAbiertaPor. Cliente	(10)
Cuenta ⊓ esCreadaPor. Empleado	(11)
Cliente ⊓ ∃1realiza. Transacción	(12)
Transacción ⊓ ∀ 1esRealizadaPor-. Cliente	(13)
Empleado ⊓ ∀1laboraEn. Institución	(14)
Institución ⊓ tieneDatosDe. (Empleado ⊓ Cliente ⊓ Transacción ⊓ Cuenta)	(15)
}	

Figura 5.17. Base de conocimiento para el AAP

5.1.8.4.2 Creación del modelado de datos

Se crea un diagrama visual el cual permite validar las reglas definidas en la base de conocimiento (ver sección 5.1.8.4.1 Obtención de reglas de la base de conocimiento4.3.4.1 Obtención de reglas de la base de conocimiento) a través de la evidencia lógica basado en la experiencia (ver **Figura 5.18)**

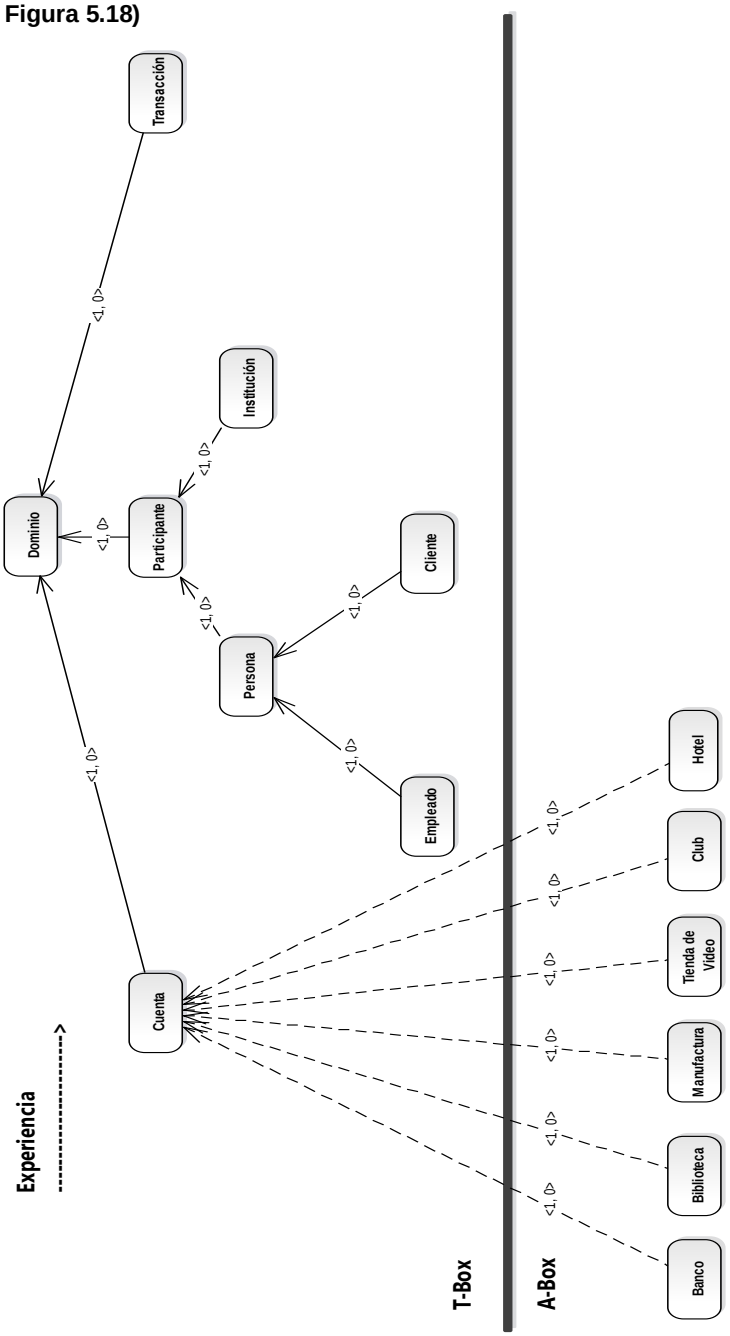


Figura 5.18. Modelado de datos del AAP

5.1.8.5 Interpretación semántica

Se realiza la correspondencia entre los objetos y relaciones del mundo real con los objetos definidos en la base de conocimiento (ver sección 5.1.8.4.1 Obtención de reglas de la base de conocimiento4.3.4.1 Obtención de reglas de la base de conocimiento) en base a contextos aplicables para el modelo de estudio.

5.1.8.5.1 Interpretación de reglas de la base de conocimiento

En esta actividad la descripción de conceptos DL, definidos en la base de conocimiento (ver sección 5.1.8.4.1 Obtención de reglas de la base de conocimiento4.3.4.1 Obtención de reglas de la base de conocimiento) son interpretados con respecto a una posibilidad infinita de un conjunto de objetos del $\text{dominio de interpretación}$ y una asociación de conceptos atómicos, roles atómicos, e individuos para objetos del dominio de interpretación. Un ejemplo de interpretación (I) para la base de conocimiento (KB) del patrón de análisis de estudio se presenta en la Tabla 5.10.

Tabla 5.10. Ejemplo de Interpretación de reglas de la base de conocimiento para cuentas de Biblioteca

Sea $AC = \{Cuenta, Transaccíon, Cliente, Empleado, Institución\}$, $AR = \{ abre, crea, esAbiertaPor, esCreadaPor, realiza, esRealizadaPor, laboraEn , tieneDatosDe \}$ y $IV = \{ cen, ana, bib, rub, pre, res, ren \}$

Sea I = $(\Delta^I, .^I)$ donde

Δ^I	=	{Banco, Biblioteca, Manufactura, Tienda de Video, Club, Prestar material, Reservar material, Renovar material, Hotel, Rubí, Ramiro, José, Guadalupe, Ana, CENIDET}
$Cuenta^I$	=	{Banco, Biblioteca, Manufactura, Tienda de Video, Club, Hotel}
$Transacción^I$	=	{Prestar material, Reservar material, Renovar material}
$Cliente^I$	=	{ Rubí, Ramiro, José, Guadalupe, Homero, Loreli}
$Empleado^I$	=	{Ana}
$Institución^I$	=	{CENIDET}
$abre^I$	=	{(Rubí, Biblioteca)}
$esAbiertaPor^I$	=	{(Biblioteca, Rubí)}
$crea^I$	=	{(Ana, Biblioteca)}
$esCreadaPor^I$	=	{(Biblioteca, Ana)}
$realiza^I$	=	{(Rubí, Prestar material)}
$esRealizadaPor^I$	=	{(Prestar material, Rubí)}
$realiza^I$	=	{(Rubí, Reservar material)}
$esRealizadaPor^I$	=	{(Reservar material, Rubí)}
$realiza^I$	=	{(Rubí, Renovar material)}
$esRealizadaPor^I$	=	{(Renovar material, Rubí)}
$laboraEn^I$	=	{(Ana, CENIDET)}
$tieneDatosDe^I$	=	{(CENIDET, Ana)}
cen^I	=	CENIDET
ana^I	=	Ana
bib^I	=	Biblioteca
rub^I	=	Rubí
pre^I	=	Prestar material
res^I	=	Reservar material
ren^I	=	Renovar material

Entonces I es una (posible) Interpretación DL con respecto a KB (ver Figura 5.19).

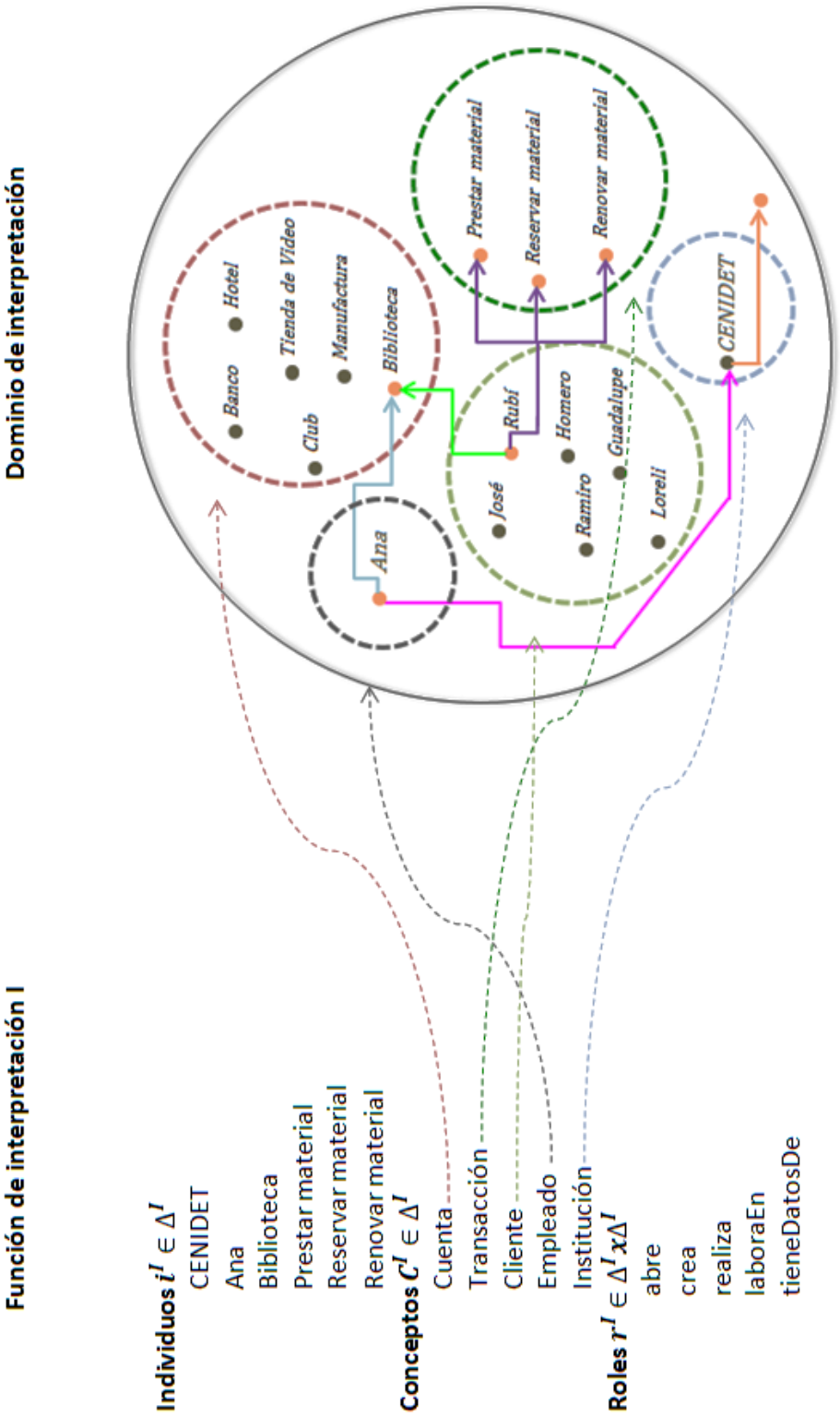


Figura 5.19. Función y dominio de interpretación para base de conocimiento del AAP

Con el ejemplo presentado en Tabla 5.10 se comprueba que la base de conocimiento es consistente porque existe un modelo de interpretación que satisface los axiomas definidos para la base de conocimiento (ver Anexo B, Definición 1.28).

5.1.8.5.2 Estructuración del modelo dominio del patrón de análisis

En esta actividad se lleva a cabo la estructuración semántica del modelo de estudio permitiendo visualizar un diagrama definido de acuerdo a las reglas determinadas en las fases anteriores (ver Figura 5.20).

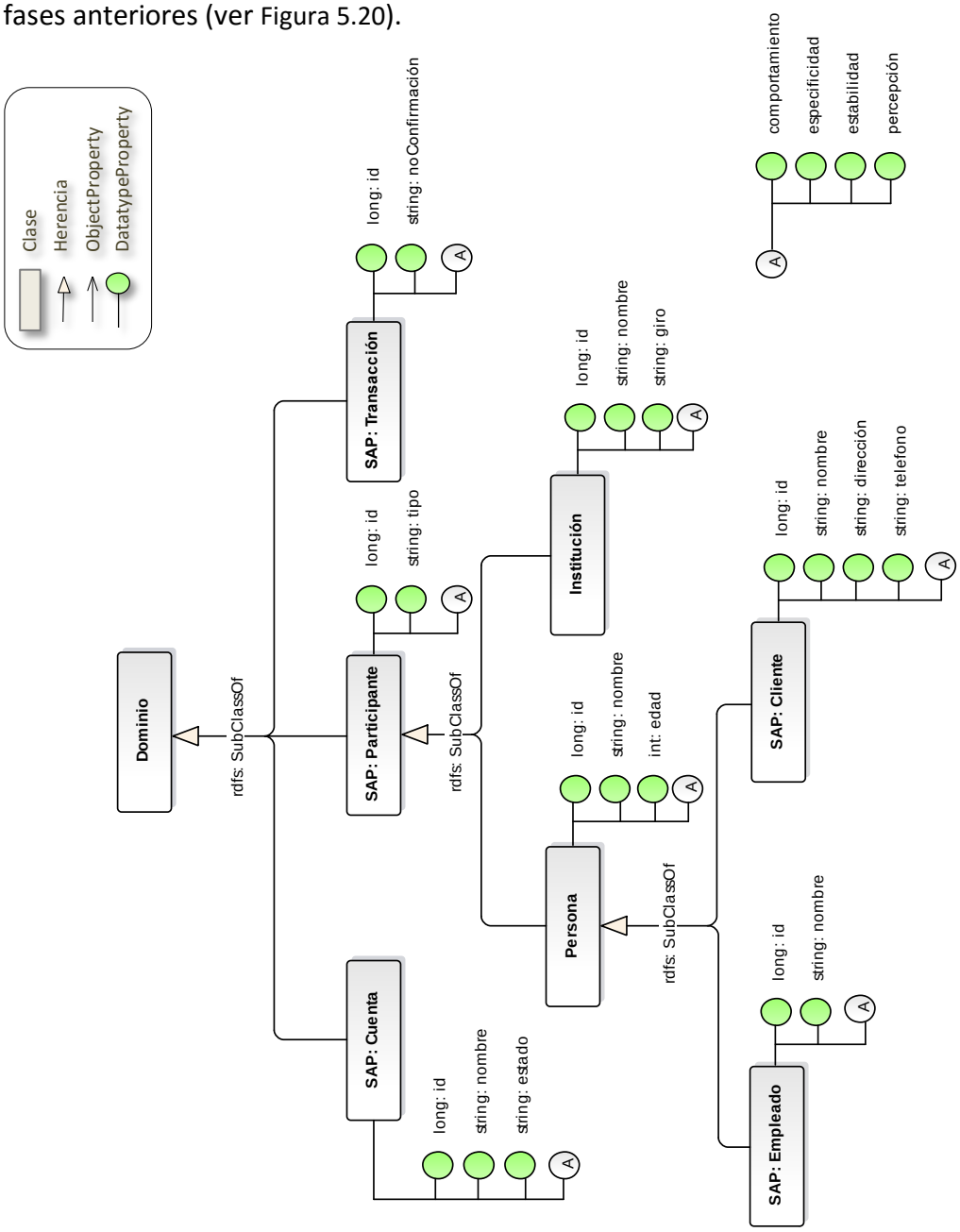


Figura 5.20. Estructuración del modelo de estudio

5.1.9 Instanciación de la ontología

En este paso se crearon los individuos de la ontología. Y consistió en poblar la ontología, asignando valores a cada una de las propiedades de las clases (ver Figura 5.21).

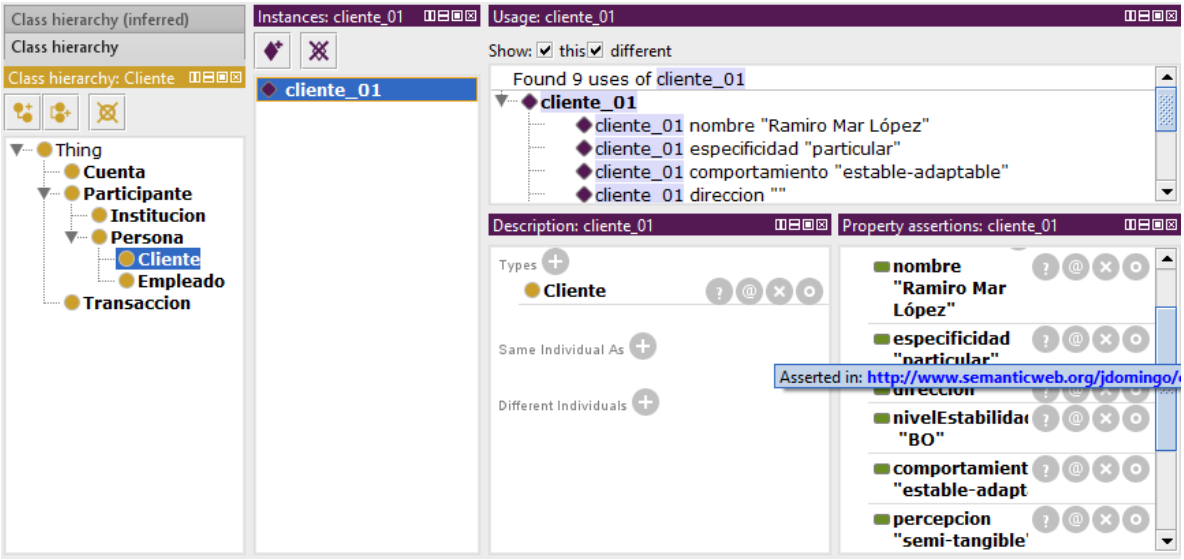


Figura 5.21. Individuos de clases de OPAC en Protégé

En este capítulo se presentó toda la documentación necesaria del proceso de para llevar a cabo la validación de un patrón de análisis utilizando la metodología MEVASE la cual fue incluida en la metodología. De igual forma se obtuvo como producto la ontología para el patrón de análisis de cuenta desarrollado a través de MECOPA. Esta documentación sirvió para reforzar el entendimiento de dicha metodología, fungiendo como un ejemplo directo de su utilización, resultados fueron exitosos, con esto queda sustentada la investigación. En el capítulo 6 se presenta conclusiones generales así como los posibles trabajos futuros, mostrados en el siguiente capítulo (Capítulo 10).

Capítulo 6. Conclusiones y Trabajos Futuros

En este capítulo se presentan las conclusiones obtenidas a través del proceso de investigación desarrollado para llevar a cabo este trabajo de tesis y los que se podrían considerar como trabajos futuros.

6.1. Conclusiones

De acuerdo a los resultados presentados podemos concluir que se cumplió con el alcance y el objetivo general establecido en la tesis ya que se alcanzaron todos y cada uno de los objetivos específicos (ver Capítulo 1 sección 1.3.2 Objetivos específicos):

- Se identificó una notación formal adecuada a través de la lógica descriptiva, para validación y composición de patrones de análisis y para su inclusión en algún lenguaje para ontologías.
- Se desarrolló a través de la Metodología para la Construcción de Ontologías de Patrones de Análisis (MECOPA) la ontología para el patrón de análisis de cuenta.
- Se desarrolló el Metodología validación de consistencia de patrones de análisis (MEVASE), la cual permite la formalización de patrones de análisis a través de lógica matemática, es decir, utiliza como lenguaje formal la “Lógica Descriptiva”.
- Se incluyó la metodología MEVASE en la Metodología MECOPA, para asegurar la consistencia de los patrones de análisis desarrollados a través de esta metodología.
- Se validó el uso de ontologías en la validación del patrón de cuenta, para representar el patrón resultante.
- Se crearon nuevas formar de representar la estructura consistente de un patrón de análisis a través de diagramas visuales.

Este trabajo de investigación intenta sentar las bases para difundir y facilitar la utilización de patrones de análisis con la finalidad de innovar en el desarrollo de software aprovechando al máximo las ventajas otorgadas a través de estos patrones de análisis.

6.2 Trabajos futuros

Los resultados obtenidos han permitido sugerir mejoras y trabajos sobre la misma línea de investigación:

1. Documentar y clasificar más patrones de análisis.
2. Desarrollar un sistema que permita generar los diagramas propuestos durante las fases de creación de ontologías para patrones de análisis en MECOPA.
3. Desarrollar metodología para la combinación semántica de patrones de análisis.

ANEXOS

En esta sección se presentan anexos relevantes para la compresión de algunas secciones correspondientes al tema de tesis.

Anexo A. Plantilla OPPAE

En este anexo se presenta la plantilla para patrones de análisis [17], se llevó a cabo un caso de prueba con el patrón de análisis de cuenta. Se logró documentar diagramas visuales del patrón de estudio los cuales son: 1) Diagrama de dependencia; 2) Diagrama de contribuciones; 3) Diagrama de prioridad; 4) Diagrama de Clases; 5) Diagrama de Secuencia; 6) Diagrama de Estado.

1. Nombre

Patrón de análisis de cuenta

2. También conocido como

Este patrón sólo es conocido por su nombre original: “Patrón de análisis de cuenta”.

3. Historial

{ Fecha: 1999
Autores: Eduardo B. Fernández y Ying Liu}

{ Fecha: 2015
Autores: José Domingo Juárez Hernández,
Moisés González García.
Cambios: Adaptar el formato de patrón original a
una estructura definida por la plantilla de
[3].}

4. Ajustes estructurales

No se tiene información referente a las secciones de Trampas anti-patrones y Patrones de diseño.

5. Problema

Sin el concepto de cuenta los usuarios necesitan llevar grandes cantidades de dinero, pueden tener problemas para reservar artículos a comprar o prestar, y tendrían graves problemas enviando fondos a lugares remotos.

6. Motivación

- Hay una variedad extensa de cuentas y es necesario un modelo genérico para describir el manejo de todos los tipos de cuentas, sin necesidad de especificar estas cuentas a detalle.

- Las cuentas tienen aspectos dinámicos como también estáticos. Por ejemplo, una cuenta de usuario tiene varios estados que afectan la forma en que es usada, no se puede retirar dinero de una cuenta congelada, por ejemplo. Cualquier solución debe reflejar y hacer cumplir los contextos definidos por estos estados.
- Debe haber una representación explicita en el modelo para los documentos utilizados normalmente en la práctica, Ej., un registro de préstamo debe corresponder a un objeto de clase. Esto hace la generación de estos documentos fáciles y el modelo es ahora más comprensible.
- Diferentes cuentas tienen diferentes tipos de transacciones, el modelo debe describir solamente un resumen, aspectos comunes de transacción.
- El modelo de análisis puede ser una representación fiel de los requerimientos sin incluir detalles de implementación. Hay que tener en cuenta que estos requerimientos pueden aparecer en diferentes dominios.
- El patrón puede describir una unidad semántica fundamental. Esto significa que el patrón puede ser bastante simple de aplicar a una variedad de situaciones.

Ejemplo: En una institución bancaria, donde los clientes tienen diferentes tipos de cuentas, ej., comprobación, ahorro, préstamo, hipotecas, etc. Los clientes utilizan estas cuentas para manejar sus transacciones financieras, ej., firmar cheques para realizar pagos, depositar dinero en sus cuentas, etc. Los clientes reciben del banco tarjetas para facilitar sus transacciones. Para la comodidad de sus clientes, los bancos pueden tener diversas sucursales u oficinas localizadas en diferentes lugares. Por lo general, una cuenta esta en a cargo de una sucursal especifica.

7. Contexto

Hay muchas instituciones, ej., bancos, bibliotecas, clubes, entre otros, que necesitan proporcionar a sus miembros o clientes una cuenta para que puedan manejar sus obligaciones financieras, cargo de comidas, comprar artículos, reservar y usar materiales, etc. Esto es muy convincente para el cliente, quien no necesita llevar grandes cantidades de dinero o necesita prestar libros, equipo deportivo, o vehículos. Manejar cuentas eficientemente y correctamente es importante para una institución para mantener a sus clientes felices.

8. Aplicabilidad

Este patrón puede ser aplicado para mantener seguimientos de las cuentas de clientes en instituciones, donde estos clientes pueden realizar diferentes tipos de transacciones con sus cuentas.

9. Requerimientos

9.1. Requerimientos funcionales (FR)

RF 1 Abrir una cuenta. La información de cliente (nombre, ID, dirección, etc.) es registrada, y el crédito es verificado. Una cuenta es creada y las tarjetas de la institución son proporcionadas a sus clientes.

RF 2 Realizar una transacción. Los clientes pueden realizar transacciones de diferentes tipos con sus cuentas. Por ejemplo, en una biblioteca los usuarios cargan libros a sus cuentas, renuevan un préstamo, pagan multas.

RF 3 Cerrar una cuenta. La cuenta es eliminada. Algunas políticas de instituciones pueden aplicarse.

9.2. Requerimientos no funcionales (NFR)

RNF 1 Rendimiento. El sistema debe tener una respuesta rápida a todas las peticiones de los clientes.

RNF 2 Seguridad. Una transacción solo puede ser cancelada por el cliente que la solicitó.

9.3. Dependencia y contribuciones

Las Figura A.1 y Figura A.2 muestran las dependencias y contribuciones. El requerimiento no funcional 1 (Rendimiento) influye negativamente en el requerimiento no funcional 2 (Seguridad): RNF 1 puede dar lugar a un tiempo insuficiente para realizar la verificación de

acceso del cliente, lo que no permite el cumplimiento del RNF 2. El requerimiento no funcional 2 (Seguridad) influye negativamente en el requerimiento no funcional 1 (Rendimiento): el requerimiento de seguridad puede dar lugar a un over-time en la verificación de acceso, por lo cual no cumple con la estimación inicial de tiempo de RNF 1.

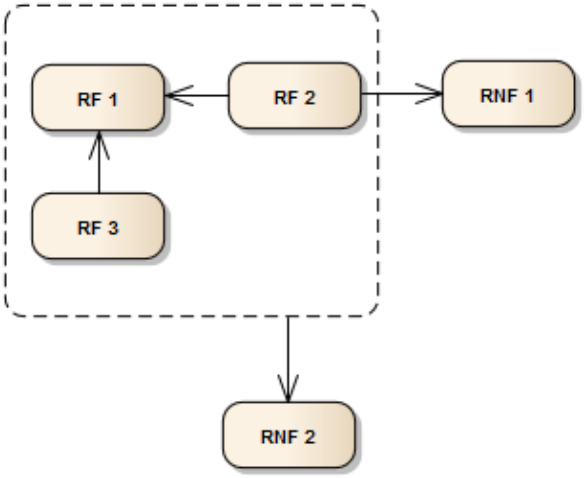


Figura A.1. Diagrama de Dependencias patrón de cuenta

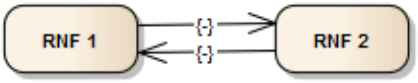


Figura A.2. Diagrama de Contribuciones del patrón de cuenta

9.4. Identificación de conflictos y guía de solución

En el diagrama de contribuciones, se identificó un conflicto entre los requerimientos no funcionales 1 y 2. Este conflicto puede resolverse mediante la asignación de prioridades a éstos requerimientos.

9.5. Prioridades

Las prioridades son mostradas en diagrama de prioridades que proporciona una visión en conjunto de los requerimientos más importantes del sistema. En la Figura A.3 se muestra un diagrama de prioridad que refleja el orden en que se debe satisfacer cada requerimiento de tal forma que se respeten sus dependencias.

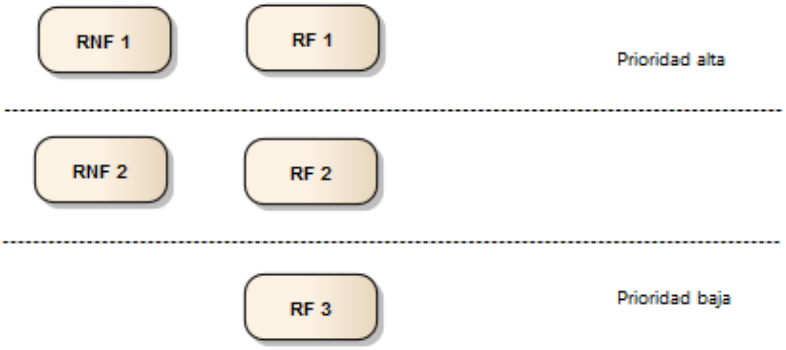


Figura A.3. Diagrama de prioridad del patrón de cuenta.

9.6. Participantes

Cliente y Personal

10. Modelado

10.1. Estructura

10.1.1. Diagrama de clases

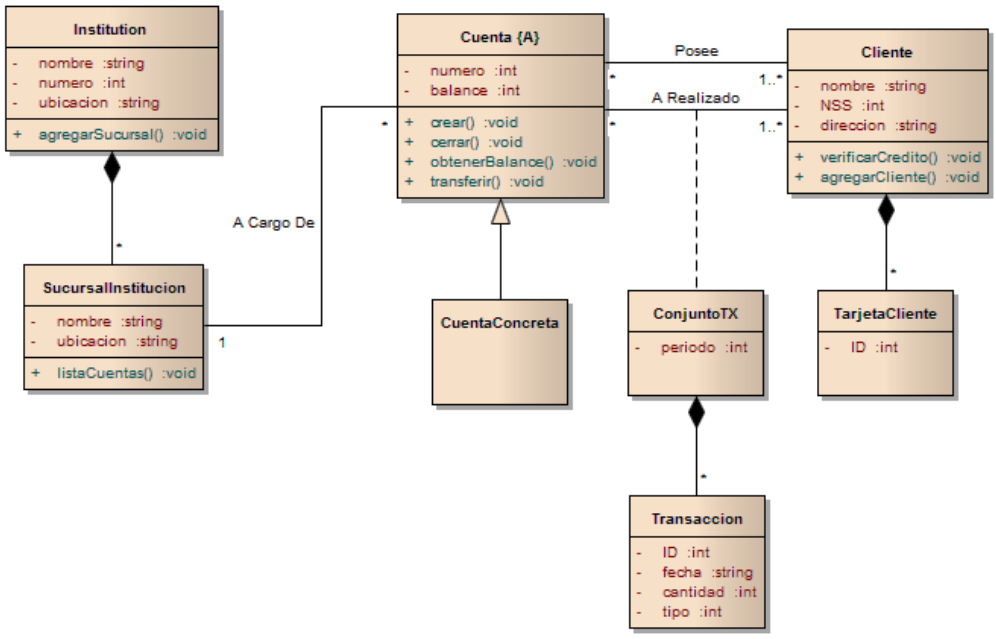


Figura A.4. Diagrama de clases del patrón de cuenta

10.1.2. Descripción de las clases

La Figura A.4 es un diagrama de clase para la realización de este caso de uso. La clase abstracta **Cuenta** puede tener una variedad de cuentas concretas. A los clientes se les da un cierto número de tarjetas y cuentas propias. Con estas tarjetas los clientes pueden realizar diferentes tipos de transacciones. La colección de todas las transacciones en una cuenta para un periodo determinado está en la clase **ConjuntoTX**. La institución genérica que puede tener varias sucursales se describe por la clase **SucursalInstitucion**. Las cuentas están a cargo de una sucursal específica.

10.2. Comportamiento

10.2.1. Diagrama de secuencia

La Figura A.5 muestra un diagrama de secuencia demostrando como un cliente abre una cuenta. Se asume que es un nuevo cliente (indicado por la operación **agregarCliente**). Una cuenta es creada y agregada a una sucursal (operación **agregarCuenta**). Un número de cuenta es regresado al cliente.

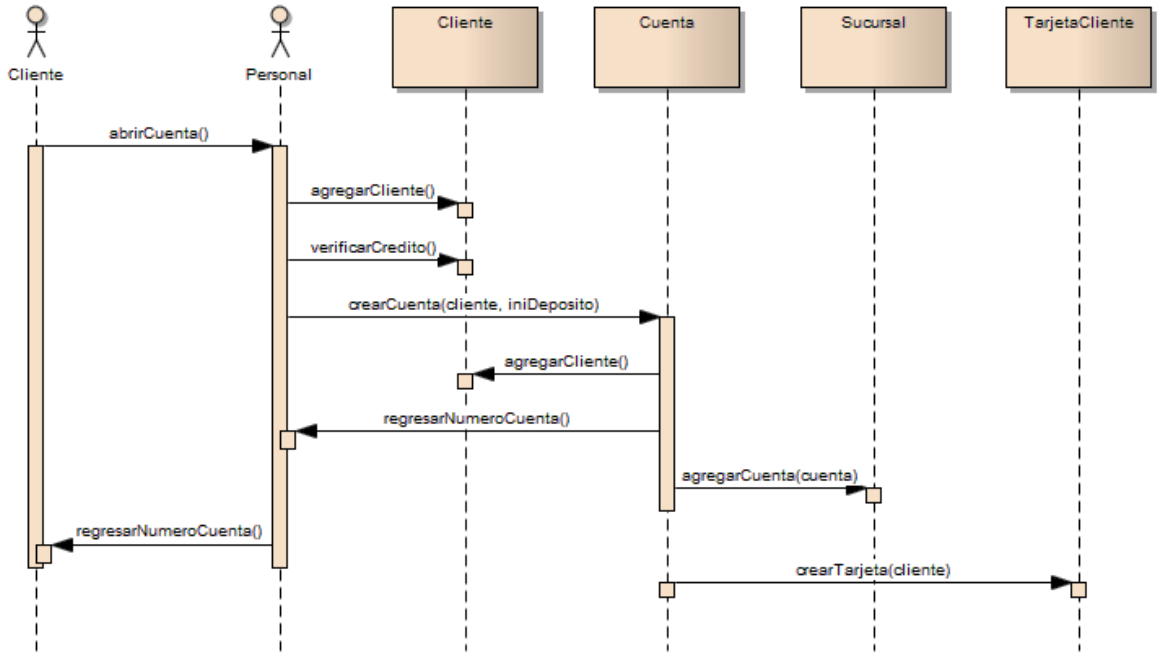


Figura A.5. Diagrama de secuencia del patrón de cuenta

10.2.2. Diagrama de estados

La Figura A.6 muestra un diagrama de estados para la clase **Cuenta**. Se asume que una cuenta puede ser desactivada, ej., si el cliente está ausente por algún tiempo, o congelado, si el cliente es un delincuente. El "súper estado" indica que la cuenta puede ser cerrada desde cualquier "estado".

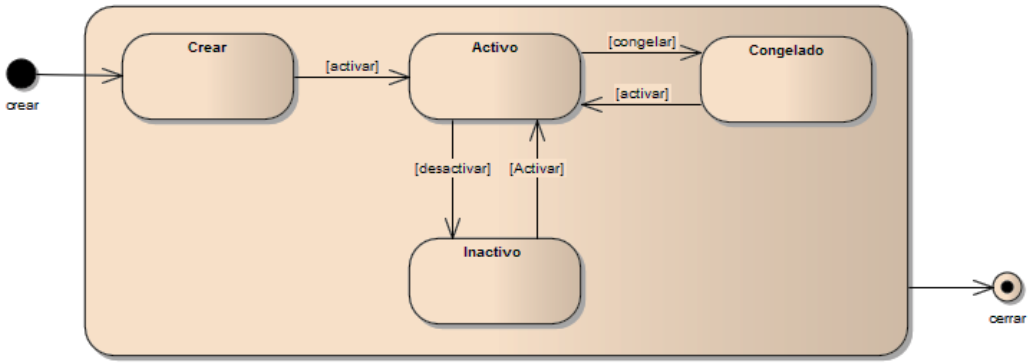


Figura A.6. Diagrama de estados para la clase Cuenta

11. Contexto resultante

El patrón debe ser adaptado para su aplicación específica, ya que no todas las situaciones descritas por este patrón son exactamente iguales (como se indica en el apartado Consecuencias). Algunos aspectos que no se representan en el patrón son:

- Diferentes tipos de clientes. Cada variedad de clientes (individuales, corporativos) podría ser manejando en una forma especial.
- Aspectos históricos, excepto para el historial de transacciones realizado en un determinado periodo de tiempo. Cuando una cuenta es cerrada, su información puede ser trasferida a un archivo histórico.
- Las funciones de administración de una cuenta, como una auditoria.
- Políticas para el manejo de una cuenta cerrada, congelada, y activación. Estos varían entre instituciones y puede ser manejadas a través de restricciones OCL (Lenguaje de restricciones de objetos) o a través de reglas de negocio.

12. Consecuencias

- Este modelo proporciona una descripción efectiva de las necesidades y puede ser utilizado para dirigir el diseño e implementación del sistema de software. No utilizar un modelo similar resultaría en un código difícil a extender y probablemente incorrecto.
- Otros casos de uso posibles para manejar cuentas son: congelar cuenta y activar/desactivar cuenta.
- Puede describir una cuenta manejada en muchos tipos de instituciones.
- Documentar por ejemplo el registro de una transacción (ej., una ficha de depósito) es representado en el modelo por objetos de clases específicas.
- Agregar instancias del **patrón Autorización** [28], el patrón puede incluir los requerimientos de autorización para los roles en sus casos de uso. La seguridad es un aspecto muy importante para las cuentas porque de las responsabilidades financieras están involucradas.

13. Ejemplos

- Aplicaciones de escritorio para la administración de diferentes tipos cuentas (Bancos, Bibliotecas, Tiendas de video, Clubes, Hoteles).
- Aplicaciones web para la administración de diferentes tipos cuentas (Bancos, Bibliotecas, Clubes).
- Aplicaciones móviles que permiten hacer diferentes tipos de operaciones en cuentas personales (Bancos).

14. Patrones relacionados

Este patrón es una refactorización y extensión del **patrón rendición de cuentas** de Fowler [29]. El patrón de Fowler es demasiado abstracto para muchos propósitos y se ha hecho el patrón más concreto mientras conserva su capacidad para ser aplicado en muchas situaciones. El concepto de rendición de cuentas de Fowler incluye responsabilidades de cómo ser un gerente o un empleado y las correspondientes rendiciones de cuenta del puesto. Además, el modelo de Fowler no considera aspectos dinámicos y considera las transacciones como un aspecto separado.

Para considerar cuentas y transacciones juntas se capturo más cercanamente el significado de cuentas en estos tipos de instituciones. Ralph Johnson and Joe Yoder tienen desarrollado un conjunto de **patrones bancarios**, incluyendo **patrones contables** [30]. No consideran también aspectos dinámicos y tiene algunos puntos complementarios para el "patrón cuenta" enfatizando aspectos monetarios. Hay [31] tiene un capítulo sobre contabilidad, discutiendo aspectos económicos y es también complementario al "patrón cuenta". Los patrones de IBM para e-business incluye un **patrón de acceso** a la cuenta [32] pero su descripción tiene pocos detalles. Como se indica anteriormente, el Partido y los patrones de reglas de Negocio pueden ser combinados con este patrón. Otros patrones complementarios incluyen **Reservación y Uso de Entidades** [33] e **Inventario** (Gerente) [34].

15. Directrices de diseño

Refinar el diagrama de clase para especificar los tipos de los atributos, especificar las tablas respectivas (al menos en la tercera forma normal) y sus claves primarias y foráneas (si se pretende utilizar una base de datos relacional en la aplicación).

16. Usos conocidos

Los siguientes son ejemplos de uso:

- Bancos, donde los clientes tiene cuentas financieras de diferentes tipos.
- Bibliotecas, donde los usuarios pueden prestar libros y cintas.
- Cuentas de Manufacturación, donde los materiales son cargados (pedidos).
- Tiendas de video, donde los clientes pueden prestar o comprar películas.
- Clubes, donde los miembros pueden cargar comidas y objetos a sus cuentas.
- Hoteles, donde los clientes que se quedan en el hotel pueden cargar comidas y llamadas telefónicas.

Anexo B. Lógica Descriptiva

En este anexo se presenta el estudio de la lógica descriptiva (DL por description logic) para la representación formal de un patrón de análisis. Sé un ejemplo relacionado con la representación formal (es decir, se ira definiendo la gramática) del patrón de análisis de cuenta, el cual es el caso de estudio del tema de tesis.

1. Lógica de descripción

En esta actividad se llevó a cabo el estudio de la lógica descriptiva (DL, por sus siglas en inglés) para la representación formal de un patrón de análisis. El Lenguaje de Atributos de Conceptos con Complementos (ALC, por sus siglas en inglés)[35] es la lógica descriptiva básica la cual es punto de partida.

Durante la explicación de la sintaxis utilizada por la DL, se tratarán ejemplos relacionados con la representación formal (es decir, se ira definiendo la gramática) del patrón de análisis de cuenta, el cual es el ejemplo de estudio.

1.1. Introducción

Las lógicas de descripción (DL, por sus siglas en inglés)[25] son unas familias de formalismos para representación de conocimiento[36]. La DL define la terminología a utilizar y permite especificar propiedades a objetos de un dominio de aplicación.

Actualmente, la DL ha ganado importancia como la base formal para la representación de ontologías. Normalmente la DL comparte dos propiedades importantes:

1. Tienen una semántica precisa y sin ambigüedad basada en la lógica.
2. Implicación lógica y la satisfacibilidad es decidible.

Este conjunto de propiedades son la base para aplicaciones independientes, ya que formalizan y complementan servicios de inferencia para la representación de conocimiento en una DL.

La representación del conocimiento está organizada en una $base\ de\ conocimiento$ DL. Una base de conocimiento DL consta de dos componentes: un $TBox$, el cual es una definición terminológica, y un $ABox$ que consiste en aserciones sobre individuos de un dominio de discurso. La definición terminología en un TBox DL comprende $conceptos$, que representan clases de objetos, y $roles$ que representan relaciones binarias sobre objetos de un dominio. Los individuos, propiedades que están definidos en el ABox, representan objetos únicos del dominio.

Ejemplo 1.1 (TBox, Concepto, Rol)

El siguiente axioma terminológico define el concepto de "Cuenta de banco", como una "Cuenta" que es "abierta" por algún "Cliente".

$$CuentaBanco \doteq Cuenta \sqcap \exists esAbiertaPor.Cliente$$

$CuentaBanco$ es un concepto representando la clase de objetos que son "cuentas de banco". $Cuenta$ es un concepto que representa la clase de "cuentas". El concepto $Cliente$ representa "clientes". $esAbiertaPor$ es un rol representando la relación binaria entre cuentas y participantes, por instancias, clientes.

- El símbolo $\doteq$ es una definición de equivalencia sobre dos conceptos. El axioma anterior define el conjunto de instancias del concepto $CuentaBanco$ como "igual" al conjunto de instancias del concepto $Cuenta \sqcap \exists esAbiertaPor.Cliente$.
- El símbolo $\sqcap$ expresa la conjunción de conceptos. $\sqcap \exists esAbiertaPor.Cliente$ representa el concepto de instancia del cual son relacionados a algunas instancias del concepto $Cliente$ a través del rol $esAbiertaPor$.

En la sección 1.2, se define la semántica precisa de axiomas terminológicos.

Ejemplo 1.2 (ABox, Individuos, Aserciones)

Las siguientes aserciones ABox definen algunas instancias de los conceptos $Cuenta$ y $Cliente$ a través del rol $esAbiertaPor$.

$Cuenta\ (cta)$	cta es una instancia del concepto $CuentaBanco$
$Cliente\ (cte)$	cte es una instancia del concepto $Cliente$
$esAbiertaPor\ (cta, cte)$	cta está en la relación $esAbiertaPor$ con cte

En la sección 1.2, se define la semántica precisa de aserciones ABox.

La DL permite definir un importante servicio de razonamiento: clasificación de conceptos a lo largo de su jerarquía de $subsunción$. Un concepto más general D subsume un concepto más específico C, denotado como $\mathrm{C} \sqsubseteq \mathrm{D}$, si todas las instancias del concepto C son también instancias del concepto D. Además los servicios de razonamiento tales como la $equivalencia\ de\ concepto$ (dos o más conceptos tienen el mismo conjunto de instancias),

$conceptos\ disjuntos$ (dos o mas conceptos no tienen instancias en común), y el $concepto\ de\ satisfacibilidad$ (algún concepto quizá tiene instancias) se pueden reducir a concepto de subsunción.

Ejemplo 1.3 Subsunción

Sea KB una base de conocimiento DL conteniendo el axioma del Ejemplo 1.1:

$$CuentaBanco \doteq Cuenta \sqcap \exists esAbiertaPor.Cliente$$

Entonces el conocimiento base (KB, por sus siglas en inglés) implica que el concepto $CuentaBanco$ es subsumido por el concepto $Cuenta$, es decir, cada instancia de $CuentaBanco$ es una instancia de $Cuenta$.

Además, los servicios de razonamientos disponibles para el conocimiento base DL son clasificación de los individuos (es decir, determinando el conjunto de conceptos que un individuo is-a (es una) instance-of (instancia-de)), recuperación de instancia (es decir, determinando el conjunto de individuos de un concepto) y satisfacibilidad (ABox) (ver sección 1.3).

1.2 Sintaxis y semánticas

La familia de conocimiento DL representan formalismos que comprenden muchos lenguajes que difieren considerablemente en expresividad y complejidad computacional. Las lógicas de descripción difieren en los constructores disponibles para la definición de conceptos y roles para axiomas terminológicos y aserciones ABox.

El lenguaje de atributos con complementos (ALC, por sus siglas en inglés) es una la lógica descriptiva básica. ALC brinda negación, conjunción, y disyunción de conceptos, así como también cuantificadores universal y existencia de roles. El axioma terminológico del Ejemplo 1.1 es un ALC.

1.2.1 Sintaxis ALC

Las lógicas de descripción definen una representación del conocimiento en un lenguaje basado en S$ímbolos\ dependientes\ de\ \ aplicaciones\ y\ \ independientes\ de\ aplicaciones$. Los símbolos dependiente de aplicación, consiste de un conjunto de conceptos atómicos representado clases de objetos, un conjunto de roles atómicos representando relaciones

binarias sobre objetos, y un conjunto de individuos representando objetos individuales de la aplicación de dominio. Los símbolos independiente de aplicación son el conjunto de conectores para construir formulas. Mientras los símbolos dependiente de aplicación se puede ser elegida arbitrariamente, el conjunto de conectores es fijo para un DL dado. La expresividad de un DL es determinada por los constructores disponibles para un concepto específico y roles.

Definición 1.4 (Símbolos dependientas de aplicación)

El conjunto numerable AS denota el conjunto de $símbolos\ \ dependientes\ de\ aplicacion$ para la lógica de descripción. AS se divide en tres pares de subconjuntos disjuntos:

$$\begin{aligned} IV &:= AS\backslash(AC \cup AR), && \text{el conjunto de } individuos, \\ AC &:= AS\backslash(IV \cup AR), && \text{el conjunto de } conceptos\ atomicos, \\ AR &:= AS\backslash(IV \cup AC), && \text{el conjunto de } roles\ atomicos. \end{aligned}$$

Ejemplo 1.5 (Símbolos (In) dependiente de aplicación)

El símbolo dependiente de aplicación se utiliza en el Ejemplo 1.1 y 1.2:

$$\begin{aligned} AC &= \{CuentaBanco, Cuenta, Cliente\} \\ AR &= \{esAbiertaPor\} \\ IV &= \{cta, cte\} \\ AS &= AC \cup AR \cup IV \\ &= \{CuentaBanco, Cuenta, Cliente, esAbiertaPor, cta, cte\} \end{aligned}$$

El símbolo independiente de aplicación del Ejemplo 1.1 es:

- $\doteq$ (Equivalencia de conceptos)
- $\sqcap$ (Conjunción de conceptos o intersección)
- $\exists$ (Cuantificación existencial)

Definición 1.6 (Descripción de concepto ALC)

Sea $A \in AC$ un concepto atómico y $R \in AR$ un rol atómico. Entonces el conjunto de descripción de conceptos ALC (o simplemente conceptos) C_{ALC} es el conjunto de expresiones mínimas que son generados por las siguientes reglas de sintaxis [36]:

C, D	$\longrightarrow$	A	(concepto atómico)
		$\top$	(concepto universal)
		$\bot$	(concepto vacío)
		$\neg C$	(negación o complemento)
		$C \sqcap D$	(conjunción o intersección)
		$C \sqcup D$	(disyunción o unión)
		$\forall R.C$	(cuantificador universal o restricción de valor)
		$\exists R.C$	Cuantificador existencial

Nota 1.7 (descripción de concepto ALC)

La descripción de un concepto especifica una clase de objetos $intencionalmente$ para describir las propiedades comunes de sus instancias. La semántica de descripción de conceptos ALC se define en la Definición 1.22.

Ejemplo 1.8 (Descripción de concepto ALC)

Sea el conjunto de conceptos atómicos AC y el conjunto de roles atómicos AR:

$$AC = \{Cuenta, CuentaBanco, Cliente\}$$
$$AR = \{abre, esAbiertaPor\}$$

Entonces los siguientes son descripciones de conceptos ALC:

$Cuenta$	(concepto atómico)
$\exists esAbiertaPor.Cliente$	(cuantificador existencial)
$Cuenta \sqcap \exists esAbiertaPor.Cliente$	(conjunción)
$\forall abrir(\neg Cuenta \sqcup Cuentabanco)$	(cuantificador universal)
$\neg\exists abrir.\forall esAbiertaPor.Cliente$	(negación)

Los siguientes no son conceptos ALC:

$esAbiertaPor$	($entregadoPor$ no es un concepto atómico)
$esAbiertaPor.Cliente$	(falta cuantificador)
$\exists\neg esAbiertoPor.Cliente$	(la negación de roles no es permitido en ALC)
$\exists(esAbiertoPor \sqcup abre).\top$	(La unión de roles no está disponible en ALC)
$\forall Cuenta.CuentaBanco$	(La cuantificación de conceptos no está permitido)

Nota 1.9 (Constructores de roles)

Hay lógicas descriptivas que permiten expresiones complejas de roles como la negación y la unión de reglas como se muestra en el Ejemplo 1.8

Las descripciones conceptuales son bloques básicos de construcción de axiomas terminológicos y aserciones ABox.

Definición 1.10 (Axiomas termológicos)

Sea C_{ALC} el conjunto de conceptos ALC para los conceptos atómicos AC y roles atómicos AR. Entonces f es un $axioma\ terminológico$ si y solo si $f = C \sqsubseteq D$ o $f = C = D$ para algunos conceptos $C, D \in C_{ALC}$.

$$TA_{ALC} := \{C \sqsubseteq D \mid C, D \in C_{ALC}\} \cup \{C \doteq D \mid C, D \in C_{ALC}\}$$

denota (indica) el conjunto de axiomas terminológicos ALC.

Nota 1.11 (Axiomas terminológicos)

En ALC, hay solo dos posibles tipos de axiomas terminológicos: implicaciones o inclusiones conceptuales $(C \sqsubseteq D)$ y $quivalencias\ o\ igualdades\ conceptuales$ $(C \doteq D)$.

- $C \sqsubseteq D$ expresa que todas las instancias de descripción del concepto C son también instancias de descripción del concepto D.
- $C \doteq D$ expresa que el conjunto de instancias de descripción del concepto C es igual al conjunto de instancias de descripción del concepto D. La semántica precisa se da en la Definición 1.24.

El Ejemplo 1.1 muestra una instancia de un axioma terminológico.

Definición 1.12 (TBox ALC)

Un $TBox$ es un conjunto finito, posiblemente un conjunto vacío de axiomas terminológicos.

Sea TA_{ALC} el conjunto de todos los axiomas terminológicos ALC. Entonces T es un $TBox\ ALC$ si y solo si T es un subconjunto de TA_{ALC}.

Nota 1.13 (TBox ALC)

Los TBoxes representan el nivel de conocimiento del esquema (estructura) sobre el dominio de discurso. Un TBox define relaciones generales entre conceptos (o clases), que se mantiene en todo momento para todos los posibles objetos de domino de conocimiento. Por lo tanto, un TBox puede ser considerado como una representación basada en la lógica de un diagrama entidad-relación o un diagrama de clase UML. Esta representa la estructura estática del dominio de discurso.

Definición 1.14 (Aserción ABox)

Sea C_{ALC} el conjunto de descripción de conceptos ALC, AR el conjunto de roles atómicos, y IV el conjunto de individuos. Entonces para $C \in C_{ALC}$, $R \in AR$, y $a, b \in IV$

- $C(a)$ es un concepto de aserción.
- $R(a, b)$ es un rol de aserción.
- Nothing (Nada) entonces es un concepto o rol de aserción.

Una $Aserción\ (afirmación)\ ABox\ ALC$ puede ser un concepto de aserción o un role aserción. $\mathrm{AA_{ALC}}$ denota el conjunto de aserción ABox ALC.

Definición 1.15 (ABox ALC)

Un $ABox$ es un conjunto finito, posiblemente un conjunto vacío de aserciones ABox.

$Sea\ AA_{ALC}$ el conjunto de aserción ABox ALC. Entonces A es un ABox ALC si y solo si A es un subconjunto finito de AA_{ALC}.

Definición 1.16 (Base de conocimiento ALC)

KB es una base de conocimiento ALC si y solo si hay un TBox ALC (T) y un ABox ALC (A) de tal manera que $KB \coloneqq T \cup A$.

Ejemplo 1.17 (TBox ALC, ABox ALC, Conocimiento Base)

Lo siguiente es una base de conocimiento ALC sobre un dominio de cuenta:

KB = {

$\top \sqsubseteq \forall esAbiertaPor.Cliente$, **1)**

$\exists esAbiertaPor.\top \sqsubseteq Cuenta$, **2)**

$Cliente \sqcap Cuenta \sqsubseteq \bot$, **3)**

$CuentaBanco \doteq Cuenta \sqcap \exists esAbiertaPor.Cliente$, **4)**

$Cliente(cte1)$), **5)**

$esAbiertaPor(cta1, cte1)$, **6)**

$esAbiertaPor(cta1, cte2)$, **7)**

$esAbiertaPor(cta2, cte2)$ **8)**

}

Las expresiones 1) al 4) son axiomas terminológicos que comprende el TBox del KB. Las expresiones 5) al 8) son aserciones que comprenden el ABox del KB.

El axioma 1) es un $rango$ definido por el rol $esAbiertaPor$. Especifica que cada objeto (T) es solo "abierta por" una instancia del concepto $Cliente$, es decir, todos los individuos, que aparecen en el rango del rol $esAbiertaPor$, son instancia del $Cliente$. Este es el caso para los individuos $cte1$ y $cte2$ en las aserciones 6), 7), y 8).

El axioma 2) es una definición del $dominio$ de rol $esAbiertaPor$. Específicamente que todos los objetos, que son abiertas por algo ($\exists esAbiertaPor.\top$), son instancias de $Cuenta$, es decir, todos los individuos, que aparecen en I dominio del rol $esAbiertaPor$, son instancias de $Cuenta$. Es el caso para los individuos $cta1$ y $cta2$ en las aserciones 6), 7) y 8).

El axioma 3) es un axioma de $disyunción$. Específica, que ninguno objeto (individuo) es al mismo tiempo instancia de $Cliente$ y una instancia de $Cuenta$.

El axioma 4) es una definición terminología. Define el concepto $CuentaBanco$ como igual a cuentas que son abiertas por al menos un cliente. Como un resultado, $cta1$ es una instancia de $CuentaBanco$ ya que $cta1$ es una instancia de $Cuenta$ porque del axioma 2) y, además, $cta1$ es abierta por algún cliente como una consecuencia de la aserción en 5) y 6).

Nota 1.18 (TBox ALC, ABox ALC, Conocimiento Base (KB))

Actualmente los sistemas de razonamiento DL soportan una sintaxis simple para la definición del dominio, rango, y axiomas de disyunción [25].

ALC no es lo suficientemente expresivo para representar una propiedad funcional sobre un rol (por ejemplo, cada cuenta es abierta por $exactamente\ un$ cliente) o restricciones de cardinalidad (por ejemplo, cada cuenta es abierta por $al\ menos$ dos y $a\ lo\ mucho$ por tres diferentes clientes). Hay extensiones para ALC que permiten la representación de la propiedad funcional y restricciones de cardinalidad (Ver sección 1.2.3).

1.2.2. Semántica ALC

La descripción de conceptos DL, axiomas terminológicos y aserciones ABox son interpretado con respecto a una posibilidad infinita de un conjunto de objetos llamados $dominio\ de\ interpretación$ y una asociación de conceptos atómicos, roles atómicos, e individuos para objetos del dominio de interpretación. Las definiciones semánticas de un DL asigna a cada axioma terminológico y aserción ABox a un valor de verdad bajo a una interpretación dada. Por lo tanto, axiomas terminológicos y aserciones ABox pueden ser vistos como formulas cerradas que evalúan un $verdadero\ o\ falso$ dentro de una interpretación dada.

Definición 1.19 (Interpretación DL)

Sea AC un conjunto de conceptos atomicos, AR el conjunto de roles atomicos, y IV el conjunto de individuos. Una $interpretación\ DL\ (I)$ es un par (Δ^I, $.^I$) donde:

- Δ^I es un conjunto no vacío de objetos llamado $dominio\ de\ interpretación.$
- $.^I$ es una función asignada a cada concepto atómico $A \in AC$ un conjunto de objetos $A^I \subseteq \Delta^I$, cada rol atómico $R \in AR$ una relación binaria $R^I \subseteq \Delta^I\ x\ \Delta^I$, y cada individuo $a\ \in IV$ un objeto $a^I \in \Delta^I$.

Ejemplo 1.20 (Interpretación DL)

Sea $AC = \{Cuenta, Cliente\}, AR = \{esAbiertaPor\}, y\ IV = \{ban, bib, man, clu\}$.

Sea I = (Δ^I, $.^I$) donde

$$\begin{aligned}
\Delta^I &= \{Banco, Biliboteca, Manufactura, Club, \\
&\quad Rubí, Ramiro, José, Guadalupe\} \\
Cuenta^I &= \{Banco, Biblioteca, Manufactura, Club\} \\
Cliente^I &= \{Ramiro, José\ Guadalupe\} \\
esAbiertaPor^I &= \{(Banco, José), (Club, Guadalupe), (Biblioteca, Ramiro)\} \\
ban^I &= Banco \\
bib^I &= Biblioteca \\
clu^I &= Club \\
jos^I &= José
\end{aligned}$$

Entonces I es una (posible) Interpretación DL.

Nota 1.21 (asunción de nombre único)

Muchos sistemas de razonamiento para lógicas descriptivas aplican la *asunción de nombre único* (UNA, por sus siglas en inglés) para la interpretación de individuos. Bajo la asunción de nombre único, se cumple lo siguiente para cada interpretación I:

$$\forall a, b\ \in IV :\ a^I = b^I \rightarrow a = b$$

Es decir, dos individuos diferentes son siempre interpretados como objetos diferentes del dominio de interpretación.

La interpretación del Ejemplo 1.23 cumple con la asunción de nombre único. Si la interpretación bib^I del individuo bib fue cambiado a $bib^I = Banco$, la UNA seria violada. En el Ejemplo 1.23, siempre se aplica la UNA.

La semántica de descripciones de conceptos se define por la extensión de la función de interpretación $.^I$ a descripciones de conceptos por inducción definiendo la interpretación extendida para todos los constructores de conceptos de la respectiva DL.

Definición 1.22 (Semántica de descripción de conceptos ALC)

Sea $I = (\Delta^I, .^I)$ una interpretación DL para conceptos atómicos AC y roles atómicos AR. Sea $C, D \in C_{ALC}$ conceptos ALC. Entonces la extensión de I a descripciónes de conceptos ALC se define inductivamente como:

$$\begin{aligned}
(\mathsf{T})^I &:= \Delta^I \\
(\bot)^I &:= \emptyset \\
(\neg C)^I &:= \Delta^I \backslash C^I \\
(C \sqcup D)^I &:= C^I \cup D^I \\
(C \sqcap D)^I &:= C^I \cap D^I \\
(\exists R.C)^I &:= \{a \in \Delta^I | \exists b \in \Delta^I : (a,b) \in R^I \wedge b \in C^I\} \\
(\forall R.C)^I &:= \{a \in \Delta^I | \forall b \in \Delta^I : (a,b) \in R^I \rightarrow b \in C^I\}
\end{aligned}$$

Ejemplo 1.23 (Semántica de descripción de conceptos ALC)

Para la interpretación I del Ejemplo 1.20, obtenemos:

$$\begin{aligned}
\mathsf{T}^I &= \Delta^I = \{Banco, Biblioteca, Manufactura, Club, \\
&\quad Rubí, Ramiro, José, Guadalupe\} \\
\bot^I &= \emptyset \\
(Cliente \sqcap Cuenta)^I &= Cliente^I \cap Cuenta^I = \emptyset \\
\neg(Cliente \sqcup Cuenta)^I &= \Delta^I \backslash (Cliente^I \cup Cuenta^I) = \{Rubí\} \\
(\exists esAbiertaPor.\mathsf{T})^I &= \{a \in \Delta^I | \exists b \in \Delta^I : (a,b) \in esAbiertaPor^I \wedge \\
&\quad b \in \mathsf{T}^I\} \\
&= \{Banco, Club, Biblioteca\} \\
(\forall esAbiertaPor.Cliente)^I &= \{a \in \Delta^I | \forall b \in \Delta^I : (a,b) \in esAbiertaPor^I \rightarrow \\
&\quad b \in Cliente^I\} = \Delta^I \\
(\forall esAbiertaPor.\neg Cliente)^I &= \{a \in \Delta^I | \forall b \in \Delta^I : (a,b) \in esAbiertaPor^I \rightarrow \\
&\quad b \in \Delta^I \backslash Cliente^I\} \\
&= [Manufactura, Rubí, Ramiro, José, \\
&= Guadalupe\}
\end{aligned}$$

Nótese que $(\forall esAbiertaPor.\neg Cliente)^I$ es equivalente a $\neg\exists esAbiertaPor.Cliente$ y por tanto es interpretado como un conjunto de objetos del dominio de interpretación Δ^I que no son abiertos por algún cliente. El conjunto de objetos, que son abiertos por algún cliente, es $\{Banco, Club, Biblioteca\}$ (ver Ejemplo 1.20), obtenemos:

$$\begin{aligned} & (\forall esAbiertaPor.\neg Cliente)^I \\ = \ & (\neg\exists esAbiertaPor.Cliente)^I = \Delta^I \backslash \{Banco, Club, Biblioteca\} \\ = \ & \{Manufactura, Rubí, Ramiro, José, Guadalupe\} \end{aligned}$$

Basado en la interpretación extendida, es posible definir cuando un axioma terminológico o aserción ABox se mantiene con respecto a una interpretación dada.

Definición 1.24 (Semántica de Axiomas y Aserciones)

Sea $C, D \in C_{ALC}$ descripciones de conceptos ALC, $R \in AR$ un rol atomico, y $a, b \in IV$ individuos. Sea I una interpretación extendida para descripciones de conceptos.
Entonces

$$\begin{aligned} I \ &\vDash\ C \sqsubseteq D \quad \text{iff} \quad C^I \subseteq D^I \\ I \ &\vDash\ C \doteq D \quad \text{iff} \quad C^I = D^I \\ I \ &\vDash\ C(a) \quad \text{iff} \quad a^I \in C^I \\ I \ &\vDash\ R(a,b) \quad \text{iff} \quad (a^I, b^I) \in R^I \end{aligned}$$

Para un axioma o aserción f, $I \vDash f$ es leído como "f es verdadero en I" o "f se mantiene en I" o "f es valido en I" o "I sadisface f" o "I es un modelo de f".

Ejemplo 1.25 (Semántica de Axiomas y aserciones)

Sea I la interpretación del Ejemplo 1.20. Entonces se cumple lo siguiente:

$I \vDash Cuenta(ban)$	Porque $ban^I \in Cuenta^I$
$I \nvDash esAbiertaPor(bib, jos)$	Porque $(bib^I, jos^I) \notin esAbiertaPor^I$
$I \vDash (\exists esAbiertaPor.\top)(bib)$	Porque $bib^I \in (\exists esAbiertaPor.\top)^I$ (consultar Ejemplo 1.23)
$I \vDash Cuenta \sqcap Cliente \sqsubseteq \bot$	Porque $(Cuenta \sqcap Cliente)^I \subseteq \bot^I$

	(consultar Ejemplo 1.23)
$I \models \mathsf{T} \doteq \forall esAbiertaPor.Cliente$	Porque $\mathsf{T}^I = (\forall esAbiertaPor.Cliente)^I$ $(consultar\ \mathrm{E}jemplo\ 1.23)$
$I \not\models Cuenta \sqsubseteq \exists esAbiertaPor.\mathsf{T}$	Porque $Cuenta^I \not\subseteq (\exists esAbiertaPor.\mathsf{T})^I$ (consultar Ejemplo 1.20 y 1.23)

La relación $\models$ entre interpretaciones y axiomas/aserciones se pueden extender a bases de conocimientos de la siguiente forma.

Definición 1.26 (Modelos de Bases de conocimientos DL)

Sea KB una $base\ de\ conocimiento\ ALC$ y I una interpretación. Entonces $I \models KB$ (se lee I es un modelo de KB o I satisface KB) si y solo si $I \models f$ para cada axioma/aserción $f \in KB$.

Ejemplo 1.27 (Modelos de bases de conocimiento DL)

Considerar los siguientes TBox y ABoxes:

$$\begin{aligned} \mathsf{T} &= \{Cuenta \sqcap Cliente \sqsubseteq \bot, \\ &\quad\ \mathsf{T} \doteq \forall esAbiertaPor.Cliente\} \\ \mathsf{A}_1 &= \{esAbiertaPor(ban, jos)\} \\ \mathsf{A}_2 &= \{Cuenta(jos)\} \end{aligned}$$

Sea I la interpretación del Ejemplo 1.20. Entonces:

$I \models T$	(consultar Ejemplo 1.25)
$I \models T \cup A_1$	(consultar Ejemplos 1.25 y 1.20)
$I \not\models A_2$	(consultar Ejemplo 1.20)
$I \not\models T \cup A_1 \cup A_2$	Porque $I \not\models Cuenta(jos)$

No todas las bases de conocimientos actualmente tienen un modelo. Si una base de conocimiento contiene contradicciones lógicas no hay una interpretación que satisfaga $cada$ axioma y aserciones de la base de conocimiento.

Definición 1.28 (base de conocimiento consistente)

Sea KB una base de conocimiento ALC. Entonces KB es $consistente$ $(o\ satisface)$ si y solo si tiene un modelo $I \vDash KB$.

Ejemplo 1.29 (base de conocimiento consistente)

Considere el TBox T y ABox A_1 y A_2 del Ejemplo 1.27. Entonces $T \cup A_1$ es consistente porque la interpretación del ejemplo 1.20 es un modelo de $T \cup A_1$ (consultar Ejemplo 1.27).

También, $T \cup A_2$ es consistente porque $I = (\Delta^I, .^I)$ donde $\Delta^I = \{Hotel\}$, $Cuenta^I = \{Hotel\}$, $Cliente^I = \emptyset$, $esAbiertaPor^I = \emptyset$, $hot^I = Hotel$ es un modelo de $T \cup A_2$.

En contraste, $T \cup A_1 \cup A_2$ no es consistente. Se asume, que hubo una interpretación $I \vDash T \cup A_1 \cup A_2$.

Entonces A_2 implica $hot^I \in Cuenta^I$. Como una consecuencia de ABox A_1 y el axioma T $\doteq \forall esAbiertaPor.Cliente$ in T, se mantiene: $hot^I \in Cliente^I$.

El hecho, es que $hot^I \in Cuenta^I$ y $hot^I \in Cliente^I$, viola el axioma $Cuenta \sqcap Cliente \sqsubseteq \bot$, es decir, $I \nvDash Cuenta \sqcap Cliente \sqsubseteq \bot$.

Esta es una contradicción a la asunción $I \vDash T \cup A_1 \cup A_2$ y por lo tanto $T \cup A_1 \cup A_2$ demuestra que es inconsistente.

Nota 1.30 (base de conocimiento consistente)

Hay algoritmos formales y completos para comprobar la consistencia de una base de conocimiento en ALC y las lógicas de descriptivas más relevantes (ver sección 3.1.4).

Definición 1.31 (Implicación Lógica)

Sea KB una base de conocimiento ALC y f un axioma o aserción ALC. Entonces $KB \vDash f$ (se lee KB implica lógicamente f o KB conlleva (supone) f) si y solo si todos los modelos de KB son modelos de f:

$$\forall I: I \vDash KB \rightarrow I \vDash f$$

Ejemplo 1.32 (Implicación lógica)

Sea T el TBox y A_1, A_2 los ABoxes del ejemplo 1.27. Entonces

$$
\begin{aligned}
T &\models Cliente \sqsubseteq \neg Cuenta \\
T &\not\models Cuenta \sqsubseteq \exists esAbiertaPor.Cliente \\
T \cup A_1 &\models Cliente(jos) \\
T \cup A_1 &\models \neg Cuenta(jos) \\
T \cup A_1 &\models (\exists esAbiertaPor.Cliente)(ban) \\
T \cup A_1 &\not\models Cuenta(ban) \\
T \cup A_1 &\not\models \neg Cuenta(ban) \\
T \cup A_1 \cup A_2 &\models Cuenta(jos) \\
T \cup A_1 \cup A_2 &\models \neg Cuenta(jos) \\
T \cup A_1 \cup A_2 &\models \top \doteq \bot
\end{aligned}
$$

Desde la base de conocimiento $T \cup A_1 \cup A_2$ no se tiene un modelo, esto implica, cualquier axioma o aserción.

Nota 1.33 (Implicación Lógica)

Los axiomas y aserciones no contenidos en una base de conocimiento (KB) pero lógicamente implicado por KB son llamados $conocimiento\ implícito$ representado por KB. En el Ejemplo 1.32 se muestra que las bases de conocimiento inconsistentes no son adecuadas para derivar conocimiento implícito porque lógicamente no implican ningún axioma o aserción. Se muestra que la satisfacibilidad e implicación lógica son decidibles para ALC y también para la mayoría de otras lógicas de descripción relevantes.
La satisfacibilidad y la implicación lógica forma la base para servicios de inferencia estándar para las lógicas de descripción (sección 1.3).

1.2.3 Expresividad de las lógicas descriptivas

La expresividad de las lógicas descriptivas extiende a ALC para agregar varios constructores de conceptos y introducir constructores de roles. La expresividad de las lógicas descriptivas como $SHIQ$ [37] y $SHOQ(D)$ [38] han ganado relevancia en el contexto de la web semántica

porque construyen las bases formales del estándar W3C para lenguajes de ontología web (OWL).

- Restricciones numéricas calificadas en roles: (Revisar…)

  $\exists \leq^n R.C$ representa el conjunto de objetos que tiene como máximo n R roles incluidos siendo instancias de C:

  $$(\exists \leq^n R.C)^I := \{a \in \Delta^I || \{b \in \Delta^I | (a,b) \in R^I \wedge b \in C^I\}| \leq n\}$$

  $\exists \geq^n R.C$ representa el conjunto de objetos que tiene que tiene como mínimo n R R incluidos siendo instancias de C:

  $$(\exists \geq^n R.C)^I := \{a \in \Delta^I || \{b \in \Delta^I | (a,b) \in R^I \wedge b \in C^I\}| \geq n\}$$

- roles transitivos: Sea R un rol. Entonces R^+ se interpreta como el cierre transitivo de la interpretación R, es decir, $(R^+)^I := \{(a,b) \in \Delta^I x \Delta^I | \exists a_0, \dots, a_n \in \Delta^I : a_0 = a \wedge a_n = b \wedge \forall i \in \{1..n\} : (a_{i-1}, a_i) \in R^I\}$.

- Roles inversos: R^- denota el inverso del rol $R \in AR$ tal que:

  $R^{-I} := \{(b,a) \in \Delta^I x \Delta^I | (a,b \in R^I)\}$.

- Jerarquías de roles: Para los roles $S, R \in AR$, el axioma $S \vee R$ expresa

  $I \vDash S \sqsubseteq R \; si \; y \; solo \; si \; S^I \subseteq R^I$ para alguna interpretación I.

- nominales: los nominales permiten la construcción de conceptos mediante la enumeración de conjuntos finitos de individuos.

  $\{i_1, i_2, \dots, i_n\}$, donde $i_1, i_2, \dots, i_n \in IV$, denota (indica) un concepto que se interpreta como $(i_1, i_2, \dots, i_n)^I := \{i_1^I, i_2^I, \dots, i_n^I\}$.

Algunas de las expresiones más relevantes de las lógicas descriptivas son $SHIQ$ y $SHOIQ$. $SHIQ$ es una extensión de ALC, incluye el cierre transitivo de roles, jerarquías de roles, roles inversos y restricciones numéricas calificadas. $SHIQ$ es soportado por muchos sistemas de razonamiento DL. $SHOIQ$ agrega nominales a $SHIQ$ y se parece a la expresividad del estándar de representación del conocimiento OWL DL.

1.3 Servicios de Inferencia

Los servicios de inferencia o razonamiento para lógicas descriptivas se pueden dividir en servicios de razonamiento para bases de conocimiento con un TBox y ABox.

1.3.1 Servicios de inferencia para TBoxes

Los servicios de inferencia estándar para un TBox (T) DL incluyen:

- Comprobación de satisfacilidad: una descripción de concepto $C \in C_{DL}$ es sadisfacible con respecto a T si y solo hay un modelo $I \vDash T$ tal que $C^I \neq \emptyset$. Conceptos insasdisfacibles dentro de un TBox usualmente indican un grave error de modelado que puede fácilmente conducir a inconsistentes bases de conocimientos cuando se agregan aserciones ABox.

- Comprobación de subsunción o clasificación de conceptos: comprobar la subsunción es decidir si $T \vDash C \sqsubseteq D$ para descripción de conceptos DL $C, D \in C_{DL}$. Si T conlleva $C \sqsubseteq D$ entonces D subsume C con respecto a T, es decir, todas las instancias de C también son instancias de D. D puede ser visto como una generalización de C o, a la inversa, C como una especialización de D.

  Se puede demostrar fácilmente que la relación de subsunción en conceptos con respecto a un TBox es una orden parcial y por lo tanto definen un grafo acíclico dirigido en conceptos de un TBox. Organizar conceptos de un TBox a lo largo de su orden de subsunción se llama clasificación de conceptos. La jerarquía de generalización/especialización de conceptos de un TBox proporciona información útil sobre la estructura del dominio representado y se puede ser utilizar para realizar consultas rápidas.

- Comprobación de equivalencia: la comprobación de equivalencias es decidir si $T \vDash C \doteq D$ para conceptos $DL\ C, D \in C_{DL}$. Los conceptos, que describen la misma clase de objetos, pueden ser vistos como sinónimos. Los sinónimos pueden indicar redundancias no intencionales dentro de la base de conocimiento. Las equivalencias son útiles para simplificar la descripción de conceptos y realizar consultas rápidas.

- Comprobación de conceptos disyuntos: esto es decidir si $T \vDash C \cup D = \perp$ para los conceptos $DL\ C, D \in C_{DL}$. Puede ser importante demostrar que los dos conceptos diferentes no pueden compartir ninguna instancia.

Todos los servicios de inferencia pueden reducirse ya sea a satisfacibilidad o subsunción. Por lo tanto, si la satisfacibilidad es decidible entonces todas las tareas de inferencia estándar son decidible.

1.3.2 Servicios de inferencia para TBoxes y ABoxes

Cuando combinamos un TBox T con un ABox A, otros servicios de razonamiento pueden ser útiles. Sea $KB := T \cup A$ una base de conocimiento que contiene tanto un TBox y un Abox. Entonces los siguientes servicios de inferencia estándar para KB pueden ser utilizados:

- Comprobación de consistencia: KB es consistente, si hay un modelo $I = KB$ (consultar Definición 1.28). Es importante descubrir bases de conocimiento inconsistentes ya que lógicamente no implican alguna declaración (consultar Ejemplo 1.32) y por lo tanto no pueden ser utilizados por los servicios de razonamiento basado en implicaciones lógicas como se indica en la sección previa 1.3.1. Nótese que los TBoxes ALC no pueden ser inconsistentes por que la interpretación (I) con un dominio vacío Δ^I es un modelo de algún TBox. Sin embargo, los TBoxes pueden contener conceptos insatisfacibles, que pueden llevar a una base de conocimientos inconsistente cuando se agrega un ABox. Por lo tanto, es importante descubrir conceptos insatisfacibles en TBoxes (sección 1.3.1).

- Comprobación de instancia: esto es decidir si $KB \vDash C(a)$ o $KB \vDash R(a, b)$ para una descripción de concepto $DL\ C \in C_{DL}$, un rol $R \in R_{DL}$, y individuos $a, b \in IV$. R_{DL} denota el conjunto de expresiones de roles de la lógica descriptiva DL. R_{ALC} es igual a el conjunto de roles atómicos AR porque ALC no proporciona algunos constructores de roles.

- Recuperación de instancia: IV_{KB} permite denotar el conjunto de individuos que aparecen en algunos axiomas o aserciones de la base de conocimiento KB. Entonces la recuperación de instancia determina el conjunto $\{a \in IV_{KB} | KB \vDash C(a)\}$ para una descripción de concepto $DL\ C \in C_{DL}$.

- Clasificación de los individuos: esta es la determinación del conjunto de conceptos, un individuo es una instanceof (instancia de). Sea AC_{KB} el conjunto de conceptos atómicos que aparecen en algún axioma o aserción de la base de conocimiento KB.

Entonces la clasificación de individuos, es determinar el conjunto $\{A \in AC_{KB} | KB \vDash A(a)\}$ para un individuo $a \in IV$.

- Recuperación del rol incluido: Sea IV_{KB} denotar el conjunto de individuos que aparecen en la base de conocimiento KB. Entonces la recuperación del rol incluido es determinar el conjunto:

$$\{b \in IV_{KB} | KB \vDash R(a, b)\}$$ para un rol DL $R \in R_{DL}$ y una instancia $a \in IV$.

Los servicios de inferencia que $recuperan\ instancias$ y $recuperan\ un\ rol\ incluido$ (rol relleno) son a menudo utilizados como mecanismos de consulta estándar a bases de conocimiento DL. Todos los servicios de inferencia anteriores pueden ser reducidos a la comprobación de consistencia.

3.1.4 Propiedades relevantes de la lógica descriptiva

Una propiedad importante de las lógicas descriptivas más relevante es la decidibilidad de satisfacibilidad en descripción de conceptos y la decidibilidad de bases conocimiento consistente. Algún otro servicio de inferencia estándar como se lista en la sección 1.3.1 y 1.3.2 pueden ser reducidos a conceptos de satisfacibilidad o base de conocimiento consistente.

Las lógicas de descripción decidibles incluyen:

- $ALCN$ que es ALC con restricciones numéricas indefinidas.
- $SHIQ$
- $SHOIQ$

Hay un número de algoritmos formales y completos para comprobación de la base de conocimiento consistente. La mayoría de los sistemas de razonamientos actuales implementan algún tipo de $algoritmo\ tableau$ [39]. Los cálculos de Tableau constan de un conjunto formal y completo de reglas de transformación para la ampliación exhaustiva de la Abox de una base de conocimiento dado a ABoxes equivalentes en el que las contradicciones lógicas se vuelven obvias.
Además de la decidibilidad, la complejidad computacional de los servicios de inferencia DL es más relevante para aplicaciones prácticas. Desafortunadamente, la comprobación de la

satisfacibilidad de las lógicas de descripción es exponencial de acuerdo al tamaño de la base de conocimientos. Por otro lado, agregar más expresividad no añade mucho a la complejidad computacional de una lógica descriptiva.

Algunos resultados de complejidad para lógicas descriptivas [40] relevantes son:

- Comprobar satisfacibilidad de descripciones de conceptos AL con respecto a un TBox no vacio es EXPTIMe-hard. AL es ALC sin complemento y con una forma restringida de cuantificación existencial.

- Comprobar la consistencia del base de conocimientos $SHIQ$ (incluyendo un TBox) es EXPTIME-complete.

- Comprobación de consistencia de base de conocimientos $ALCFIO$ es NEXPTIME-hard. $ALCFIO$ es ALC extendida con roles funcionales, roles inversos y nominales (ver sección 1.2.3).

- Razonamiento en bases de conocimiento $SHOIQ$ es NEXPTIME-complete.

- Razonamiento en OWL-Lie es EXPTIME-completé. Esto es porque OWL-Lite contiene AL y está contenida en $SHIQ$.

- Razonamiento en OWL-DL. Apoyo amplio para la representación de conocimiento, es NEXPTIME-complete. Esto es porque OWL-DL contiene $ALCFIO$ y está contenido en $SHOIQ$.

En resumen, el razonamiento en las lógicas descriptivas más relevantes es al menos EXPTIME-hard. Afortunadamente, en aplicaciones prácticas un error puede ser evitado. Sin embargo, se debe tener mucho cuidado en la estructuración de la base de conocimiento para evitar errores.

Referencias

[1] A foundation for building stable analysis a foundation for building stable analysis. Haitham Safwat Hamza. University of Nebraska, 2002.

[2] Almacenamiento y uso de patrones de análisis apegados a su semántica. Ramiro Mar Lopez Ramirez. Tesis de Maestría. CENIDET, 2015.

[3] A systematic analysis patterns specification. R. Raminhos, M. Pantoquilho. Desenvolvimento de Novas Tecnologias, 2829-516 Caparica, Portugal.

[4] On the integration of stable analysis patterns with traditional patterns. Haitham S. Hamza. Computer Science& Engineering Dept., University of Nebraska-Lincoln, Lincoln, NE 68588, USA.

[5] Towards A Pattern Language for Developing Stable Software Patterns. Haitham Hamza and Mohamed Fayad. Computer Science and Engineering Dept., University of Nebraska-Lincoln Lincoln, NE 68588, USA.

[6] Analysis Patterns: Reusable Object Models. Fowler Martin. Published 1996, Pag. 384.

[7] No TitleThinking objectively: an introduction to software stability, Magazine Communications of the ACM. Mohamed Fayad. Magazine Communications of the ACM," Vol. 44, No. 9, Page 95–98, 2001.

[8] Software quality attributes and trade-offs. Patrik Berander, Lars-Ola Damm, Jeanette Eriksson, Tony Gorschek, Kennet Henningsson, Per Jönsson, Simon Kågström, Drazen Milicic, Frans Mårtensson, Kari Rönkkö, Piotr Tomaszewski. Blekinge Institute of Technology, 2005.

[9] Methontology: from ontological art towards ontological engineering. Mariano Fernández, Asunción Gómez Pérez, Natalia Juristo. Laboratorio de Inteligencia Artificial Facultad de Informática Universidad Politécnica de Madrid, 1997.

[10] IEEE standard for developing software life cycle processes. Software Engineering Standards Committee of the IEEE Computer Society, 1997.

[11] Construcción de una Ontología Mediante Métodos Semiautomáticos. Osvaldo Daniel Fernández Bonilla. Tesis de Maestría. CENIDET, 2010.

[12] Estudio de factibilidad para la formulación de Reglas de Combinación de Patrones de Diseño en Arquitecturas Orientadas a Objetos. Sergio Rene Gonzales Castañeda. Tesis de Maestría. CENIDET, 2010.

[13] Generación de Ontologías En OWL a Partir de Diagramas de Clases de UML Mediante MDA (Arquitectura Dirigida Por Modelos). Juan Antonio Galido Garcia. Tesis de Maestría. CENIDET, 2011.

[14] Creación de una Ontología para el Dominio de e-Gobierno en México. Norma Elizabeth Valencia Vargas. Tesis de Maestría. CENIDET, 2011.

[15] Ambiente de Modelado de Arquitecturas de Software Conducido por Reglas de Combinación de Patrones de Diseño. José Salazar Robles. Tesis de Maestría. CENIDET, 2013.

[16] Reglas para la Combinación de Patrones de diseño. Héctor Uriel Pérez Rojas. Tesis de Maestría. CENIDET, 2014.

[17] Prototipo para la representación visual de patrones de análisis usando UML. Rubí Celia Martínez Jiménez. Tesis de Maestría. CENIDET, 2013.

[18] Hierarchical patterns: a way to organize (analysis) patterns. Lubor Sesera. Systemics, cybernetics and informatics, Vol. 3, No. 4, Page 37–40, 2004.

[19] The Umbrella Pattern Language. Haitham S. Hamza and Mohamed E. Fayad. Computer Science and Engineering Dept. University of Nebraska-Lincoln, 2004.

[20] Towards a Framework for Analysis Patterns Evaluation. Haitham Hamza , Ahmed Mahdy , and Mohamed E. Fayad. Computer Science and Engineering Department University of Nebraska-Lincoln, 2004.

[21] Improving Analysis Patterns Reuse: An Ontological Approach. Haitham S. Hamza. Computer Science & Engineering Dept. University of Nebraska-Lincoln, 2004.

[22] Enterprise Ontology: Theory and Methodology. Jan L.G. Dietz. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2006.

[23] The Role of Analysis Patterns in Systems Analysis. Anna Bobkowska, Jakub Grabowski Gdansk. University of Technology, Poland, 2004.

[24] Composing analysis patterns to build complex models: Flight reservation. Zhen Jiang and Eduardo B. Fernandez. West Chester University, college of the sciences & mathematics, 2014.

[25] Document Verification with Temporal Description Logics. Franz Weitl. Fakultat f ¨ ur Informatik und Mathematik ¨ Universitat Passau, 2007.

[26] Terminología y Sociedad del conocimiento. Amparo Alcina, Elena Rambla, Esperanza Valero. Leseprobe, Zu, 2009.

[27] Designing Accounting Transaction Ontology. Irvan Iswandi. School of Electrical Engineering and Informatics Institut Teknologi Bandung Bandung, Indonesia, 2014.

[28] A pattern language for security models. Eduardo B. Fernandez and Rouyi Pan. Dept. of Computer Science and Eng. Florida Atlantic University, 2001.

[29] Analysis Patterns: Reusable Object Models. Fowler Martin. New edition, 1997.

[30] Inventory and accounting, Part of Banking patterns. R. Johnson and J. Yoder, 2000.

[31] Data Model Patterns: A Metadata Map. D Hay. Imprint Morgan Kaufmann, 2006.

[32] Patterns for e-business. Endrei, Mark, and Carla Sadtler. IBM Corp, 2004.

[33] An Analysis Pattern for Reservation and Use of Reusable Entities. Eduardo B. Fernandez and Xiaohong Yuan. Dept. of Computer Science and Engineering, Florida Atlantic University, 1999.

[34] Stock Manager : An Analysis Pattern for Inventories. Eduardo B. Fernandez. Dept. of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431.

[35] Lógicas para la red. Antonia Huertas. Universitat Oberta de Catalunya, Computer Science, Multimedia and Telecommunications, 2006.

[36] *The Description Logic Handbook: Theory, Implementation, and Applications.* Franz Baader, Ian Horrocks and Ulrike Sattler. *(2nd Edition)*, chapter 3. Cambridge University Press, 2007.

[37] Reasoning with Individuals for the Description Logic SHIQ. Ian Horrocks , Ulrike Sattler , and Stephan Tobies. Department of Computer Science, University of Manchester, UK, 2000.

[38] Ontology Reasoning in the SHOQ (D) Description Logic. Ian Horrocks and Ulrike Sattler. Department of Computer Science University of Manchester, UK, 1994.

[39] Tableau Algorithms for Description Logics. Franz Baader. Theoretical Computer Science RWTH Aachen Germany.

[40] Disponible en internet. Complexity of reasoning in Description Logics. School of Computer Science. http://www.cs.manchester.ac.uk/, 2016.