Universidad de las Ciencias Informáticas

Facultad 3

Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

José Wilfredo De La Paz Valdés

Tutores:

Ing. Yordanis García Leiva

MsC. Yarina Amoroso Fernández

Cotutor:

Ing. Reinier Silverio Figueroa

Junio, 2016

“Año 58 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

José Wilfredo De La Paz Valdés

_____ _____

MsC. Yarina Amoroso Fernández Ing. Yordanis García Leiva

_____

Ing. Reinier Silverio Figueroa

AGRADECIMIENTOS

Agradezco a todas aquellas personas a las cuales he conocido y que de una forma u otra han hecho más agradable, divertida e interesante mi estancia en la universidad.

Un agradecimiento especial a mi mamá que a pesar de las circunstancias nunca dejó de apoyarme, darme consejos y valor suficiente para enfrentar los problemas.

A mi hermana por todas las veces que sacrificó su bienestar por el mío.

A la mujer que me dio el mayor regalo que jamás me han dado, a mi hijo por existir y darme alegría hasta los momentos más difíciles.

A mis padres adoptivos Gretter y Ricardo por aguantarme todas las malcriadeces de niño grande.

A mis compañeros de cuarto que fueron muchos, pero los tengo a todos muy presentes, por celebrar conmigo los buenos momentos y apoyarme en los malos, Alian, Juan Carlos, Adrián, Ángel, Ricardo Longo y, por último, pero no menos importante David en su condición de refugiado.

A mis tutores, si porque son varios, por ayudarme siempre que lo necesité, Yordanis, Reinier, Yarina, Juan David y todas aquellas otras personas que me dio al menos una idea, pues a ellos también los considero tutores.

A mis compañeros de tesis Yamila y Verónica por alentarme a seguir trabajando y por ocuparme de más a Yordanis.

A todos mis compañeros de aula por darme el placer de compartir la universidad con ellos.

Para finalizar un agradecimiento al tribunal por su trabajo impecable y por haberme dado la medida para ser un mejor ingeniero.

Agradezco sinceramente a todos por hacer posible la realización de esta tesis, gracias por hacer posible que hoy, sea un graduado orgulloso de la Universidad de las Ciencias Informáticas.

DEDICATORIA

A mi mamá por ser mi consejera y a la vez cómplice de todas mis locuras.

A mi abuela por darme ánimos hasta cuando todo iba bien.

A mi hermana y mi sobrina por darme su cariño tan necesario.

Al amor de mi vida, Lucy.

A mi hijo autor principal de esta tesis.

A Gretter y Ricky por hacer posible tantas cosas que parecían imposibles.

A todos mis amigos.

RESUMEN

La ambigüedad es el término que hace referencia a aquellas estructuras gramaticales que pueden entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión. Como respuesta al tema, surge la técnica de desambiguación del sentido de las palabras, permitiendo la determinación desde el punto de vista computacional del significado de las palabras, para su uso en un contexto dado, a partir de un conjunto de sentidos posibles. Los textos jurídicos no están exentos de la existencia de términos ambiguos. En el campo de la informática se han realizado algunos trabajos con el propósito de analizar este problema lingüístico, sin embargo, ninguno brinda una propuesta de desambiguación de textos jurídicos. La presente investigación tiene como objetivo, desarrollar una herramienta informática que permita reducir la cantidad de ambigüedad sintáctica presente en textos de la Legislación Cubana, contribuyendo a la desambiguación de los mismos. La solución está basada en la formulación de un método de desambiguación basado en inteligencia artificial, compuesto por cuatro fases, en las cuales se definen un conjunto de reglas heurísticas, a partir de las categorías sintácticas que originan ambigüedad estructural en un texto. Estas fueron implementadas con el uso de tecnologías de código abierto. La herramienta obtenida constituye un resultado para el Grupo de Investigación de Informática Jurídica del Centro de Gobierno Electrónico.

PALABRAS CLAVES: ambigüedad, desambiguación, herramienta informática, Legislación Cubana, reglas heurísticas, sintáctica, textos jurídicos

ABSTRACT

Ambiguity is the term that refers to those grammatical structures that can be understood in various ways or to support different interpretations and therefore to give reason to doubt, uncertainty or confusion. In answer to the subject, the technique to sense disambiguation of words arises, allowing the determination from the computational point of view of the meaning of words, for using them in a given context, from a set of possible meanings. The legal texts are not exempt of the existence of ambiguous terms. In the field of computer science there have been done some work in order to analyze this linguistic problem, however, none offers a proposal for disambiguation of legal texts. This research aims to develop a software tool to reduce the amount of syntactic ambiguity present in texts of The Cuban Legislation, contributing to the disambiguation of them. The solution is based on the formulation of a disambiguation method based on artificial intelligence, it consists of four phases, in which a set of heuristic rules are defined, from the syntactic categories that cause structural ambiguity in a text. These were implemented with the use of open source technologies. The tool obtained is a result from the Legal Research Group of the Electronic Government Center.

KEYWORDS: *ambiguity, Cuban Legislation, disambiguation, heuristic rules, legal texts, software tool, syntactic*

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1. Introducción 5
1.2. Conceptos fundamentales asociados al dominio del problema 5
1.3. Revisión del estado del arte 7
1.3.1. Métodos y herramientas para la desambiguación 7
1.4. Metodología de desarrollo de software 10
1.5. Herramientas de Ingeniería del Software Asistidas por Computadoras 11
1.6. Lenguajes de programación 12
1.7. Entorno de Desarrollo Integrado 13
1.8. Patrones de diseño 14
1.9. Pruebas 14
1.10. Conclusiones parciales 15

CAPÍTULO 2: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA INFORMÁTICA 16

2.1. Introducción 16
2.2. Descripción del proceso de desarrollo 16
2.2.1 Fase de Exploración 16
2.2.1.1 Requisitos funcionales 16
2.2.1.2 Requisitos no funcionales 17
2.2.1.3 Historias de usuario 18
2.2.2 Fase de Planificación 20
2.2.2.1 Estimación de esfuerzo por historias de usuarios 20
2.2.3 Fase de Iteraciones por entrega 21
2.2.3.1 Plan de duración de las iteraciones 22
2.3. Diseño de la herramienta 22
2.3.1 Arquitectura 22
2.3.2 Patrones de diseño 23
2.3.3 Estándares de codificación 24
2.3.4 Diagrama de clases del diseño 24
2.4. Método de desambiguación 25
2.5. Descripción del sistema 29
2.6 Conclusiones parciales 29

CAPÍTULO 3: PRUEBAS A LA HERRAMIENTA INFORMÁTICA 31

3.1 Introducción 31

3.2 Técnica de validación de los requisitos 31

3.3 Validación del diseño 31

3.3.1 Relaciones entre clases (RC) 31

3.3.2 Tamaño Operacional de clases (TOC) 34

3.4 Pruebas 36

3.4.1 Pruebas de unidad 36

3.4.1.1 Método de caja blanca 36

3.4.1.2 Método de caja negra 39

3.4.2 Pruebas de aceptación 40

3.5 Aporte de la investigación 42

3.6 Conclusiones parciales 44

CONCLUSIONES GENERALES 45

RECOMENDACIONES 46

BIBLIOGRAFÍA REFERENCIADA 47

ANEXOS 49

Anexo 1: Diagrama de clases del diseño 49

Anexo 2: Acta de Aceptación de los requisitos. 50

Anexo 3: Acta de Liberación Interna de Productos de Software. 51

Anexo 4: Acta de aceptación de la herramienta por parte del cliente. 52

ÍNDICE DE FIGURAS

Figura 1. Diagrama de paquete (Fuente: elaboración propia). 23

Figura 2. Diagrama de clases del diseño acotado a sus clases y relaciones (Fuente: elaboración propia). 25

Figura 3. Fases del método de desambiguación (Fuente: elaboración propia).....26

Figura 4. Representación en (%) de los resultados de la aplicación de la métrica RC (Fuente: elaboración propia). 33

Figura 5. Representación en (%) de los resultados de la aplicación de la métrica TOC (Fuente: elaboración propia). 35

Figura 6. Código de la funcionalidad *detectarAmbiguedad* (Fuente: elaboración propia). 37

Figura 7. Grafo del camino básico del código de la funcionalidad *detectarAmbiguedad* (Fuente: elaboración propia). 38

Figura 8. No conformidades detectadas al aplicar el método de caja negra (Fuente: elaboración propia). 40

Figura 9. Análisis del texto ambiguo a través de la herramienta (Fuente: elaboración propia). 43

Figura 10. Análisis del texto desambiguado a través de la herramienta (Fuente: elaboración propia). 43

Figura 11. Diagrama de clases del diseño de la herramienta (Fuente: elaboración propia). 49

ÍNDICE DE TABLAS

Tabla 1. Conjunciones sintácticamente ambiguas *(Zapata, et al., 2007).* 6
Tabla 2. Preposiciones subordinantes *(Zapata, et al., 2007).* 6
Tabla 3. HU “Realizar desambiguación sintáctica” 18
Tabla 4. HU “Realizar propuesta de desambiguación al usuario” 19
Tabla 5. HU “Brindar información al usuario” 20
Tabla 6. Estimación de esfuerzo por HU. 21
Tabla 7. Plan de duración de las iteraciones 22
Tabla 8. Rango de valores para medir la afectación de los atributos de calidad (RC). 32
Tabla 9. Rango de valores para medir la afectación de los atributos de calidad (TOC). 34
Tabla 10. Caso de prueba para el camino básico #4. 39
Tabla 11. Caso de prueba de la HU “Importar documento” 39
Tabla 12. Caso de prueba de Aceptación de la HU “Realizar desambiguación sintáctica”. 41

INTRODUCCIÓN

El lenguaje es el gran instrumento de comunicación del que dispone la humanidad. El mismo está íntimamente ligado a la civilización y es la herramienta humana más desarrollada que existe, pues permite explicar ideas, sentimientos e impresiones. Cuando las ideas transmitidas de forma verbal o escrita, pueden interpretarse de diversas maneras, en dependencia del contexto, ocurre el fenómeno de la ambigüedad lingüística.

La ambigüedad lingüística puede afectar a los textos, sin importar sus orígenes. Los textos jurídicos no están exentos de la existencia de este fenómeno que interfiere en su comprensión. Este es un tema de investigación que ha acompañado a las ciencias jurídicas en comunión con las ciencias filológicas y la lingüística computacional, para llegar a contar con herramientas que, basadas en reglas y modelos de redacción de documentos legales, ayuden a los operadores jurídicos a redactar y revisar los documentos que emiten.

La ambigüedad se divide en tres tipos fundamentales: la semántica, la léxica y la sintáctica. La sintáctica abarca prácticamente cualquier contexto, pues es propia del idioma y viene acompañada al uso de preposiciones y conjunciones. Desde el punto de vista sintáctico, la ambigüedad genera diferentes representaciones para una misma frase.

En el ámbito internacional se han realizado estudios y obtenido métodos de solución que permiten la reducción de la ambigüedad sintáctica en diferentes tipos de contextos. En Cuba existen pocos avances en el diseño e implementación de soluciones informáticas que posibiliten mejorar la compresión de textos jurídicos y reducir la ambigüedad sintáctica que presenten. La Sociedad Cubana de Derecho e Informática, organización académica que contribuye al desarrollo de la Informática Jurídica en Cuba y el Centro de Gobierno Electrónico (CEGEL), adjunto a la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI), han realizado algunos trabajos en relación al tema, los que resultan insuficientes aún en la eliminación o reducción de los problemas de ambigüedad sintáctica presentes en los textos de la Legislación Cubana, caracterizados por:

- La existencia de dificultades lingüísticas que posibilitan la ocurrencia de ambigüedad en la interpretación de los términos.
- Incertidumbre en la interpretación y comprensión de los escritos.
- Inconsistencia en el análisis de contenidos.

- Baja comprensión de los textos.

La situación problemática antes descrita ha generado el siguiente **problema de investigación:**

¿Cómo reducir la ambigüedad sintáctica presentes en textos de la Legislación Cubana y contribuir a la desambiguación de los mismos?

Tomando en cuenta el problema antes propuesto se define como **objeto de estudio:** el proceso de desambiguación de textos.

Para ello se identifica como **campo de acción:** el proceso de desambiguación de los textos de la Legislación Cubana.

Determinándose como **objetivo general:** desarrollar una herramienta informática que permita reducir la ambigüedad sintáctica presente en textos de la Legislación Cubana y contribuir a la desambiguación de los mismos.

Se desglosan del objetivo general los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación en función de los términos relacionados con la ambigüedad y desambiguación de textos.
- Desarrollar una herramienta informática que permita reducir la cantidad de ambigüedades sintácticas presentes en textos de la Legislación Cubana.
- Validar la herramienta informática aplicando diferentes métodos y métricas de validación de software.

Para dar cumplimiento a los objetivos propuestos se definen las siguientes **tareas de la investigación:**

- Desarrollo del marco teórico referencial de la investigación en función de los términos relacionados con la ambigüedad y desambiguación de textos.
- Caracterización de cada una de las herramientas a utilizar en la investigación.
- Identificación y validación de los requisitos funcionales y no funcionales a tener en cuenta en el desarrollo de la solución.
- Definición de la arquitectura a utilizar en el desarrollo del trabajo.
- Implementación de los métodos definidos para la desambiguación de textos legales.

- Realización de pruebas de validación a la herramienta informática obtenida, aplicando métodos y métricas.

- Realización del informe final de la investigación.

Determinándose como **idea a defender:** el desarrollo de una herramienta informática permitirá reducir la ambigüedad sintáctica presente en los textos de la Legislación Cubana y contribuye a la desambiguación de los mismos.

Métodos Teóricos:

Analítico – sintético: el análisis se utilizó para descomponer el problema de investigación en partes lógicas, buscando un mejor entendimiento del proceso de desambiguación de textos legales. La síntesis permitió la unión de las partes previamente analizadas, descubrir las relaciones esenciales que existen entre estas y sintetizar todo como una solución integrada.

Inductivo – deductivo: la inducción permitió realizar un razonamiento partiendo de lo particular a lo general, reflejando en el resultado final lo común que existe en cada bibliografía estudiada. La deducción permitió a través de los conocimientos generales adquiridos durante la investigación, inferir otros conocimientos lógicos como las conclusiones parciales y generales de la investigación.

Análisis histórico – lógico: se utilizó para identificar la situación actual en relación al estado de los procesos de desambiguación y mejora en la comprensión de textos legales, permitiendo definir los autores más reconocidos y los trabajos más importantes realizados hasta el momento, vinculados con el tema de investigación.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1: Fundamentación Teórica.

Se describen conceptos relacionados con la problemática antes mencionada, así como el estudio de los trabajos existentes en la actualidad para la desambiguación de textos. Además, se describe la metodología que guio todo el proceso de desarrollo del software, el lenguaje de programación, el entorno de desarrollo integrado (IDE, según sus siglas en inglés) y las herramientas de ingeniería de software asistida por computadora (CASE, según sus siglas en inglés) utilizadas en la implementación de la solución propuesta.

Capítulo 2: Análisis, diseño e implementación de la herramienta informática.

En el capítulo se aplican las fases de Exploración, Planificación e Iteraciones por entrega definidas por la metodología XP (*Extreme Programming* o Programación Extrema), generando los artefactos correspondientes a cada una de estas. Se describe también el método de desambiguación propuesto y las reglas heurísticas definidas. Además, se describe la arquitectura y patrones de diseño empleados.

Capítulo 3: Pruebas a la herramienta informática.

Se aplica la fase de producción de la metodología XP, con la realización de pruebas unitarias y pruebas de aceptación a la herramienta propuesta, describiéndose los resultados generados. Además, se exponen los resultados de la aplicación de técnicas de validación de requisitos y métricas de validación del diseño.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se realiza la revisión del estado del arte sobre la desambiguación de textos, describiéndose algunos métodos y trabajos realizados hasta el momento en relación al tema. Además, se presentan las principales definiciones asociados al dominio del problema. También se describe la metodología, lenguaje de programación, entorno de desarrollo integrado y herramienta CASE utilizada en la realización de la tesis.

1.2. Conceptos fundamentales asociados al dominio del problema

El término ambigüedad se define en correspondencia con la rama de la ciencia que lo utiliza. Teniendo en cuenta la naturaleza del problema de investigación propuesto, el autor del presente trabajo decide analizar las siguientes definiciones:

Ambigüedad: dicho especialmente del lenguaje, que puede entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión *(rae.es, 2016).*

Ambigüedad: término que hace referencia a aquellas estructuras gramaticales que pueden entenderse de varios modos o admitir distintas interpretaciones y dar, por consiguiente, motivo a dudas, incertidumbre o confusión *(Ramos, 2009).*

Ambigüedad: puede presentarse cuando es posible admitir diferentes interpretaciones a partir de la representación de una oración; también, se presenta cuando existe confusión al tener diversas estructuras asociadas a la misma oración *(Zapata, et al., 2007).*

Teniendo en cuenta el objetivo de la presente investigación, el autor decide aplicar a lo largo del trabajo la definición de ambigüedad dada por el autor Carlos Zapata en el artículo científico *"Un Método para la desambiguación sintáctica de tipo coordinativo y preposicional"*. Entre los tipos de ambigüedad que existen se encuentra la sintáctica:

Ambigüedad sintáctica: se presenta cuando una oración tiene asociada más de una representación sintáctica, es decir, cuando más de una regla gramatical representa dicha oración (*Zapata, et al., 2007*).

La ambigüedad sintáctica se divide en 2 grupos fundamentales:

Ambigüedad sintáctica coordinativa: se puede presentar, cuando una oración contiene más de una palabra de tipo conjunción. Esta ambigüedad se clasifica en copulativa, disyuntiva o mixta (*Zapata, et al., 2007*).

Es importante señalar que no todas las conjunciones generan ambigüedad sintáctica. En la tabla 1 se listan las conjunciones sintácticamente ambiguas que pueden causar ambigüedad en un texto.

Tabla 1. Conjunciones sintácticamente ambiguas *(Zapata, et al., 2007).*

Conjunciones coordinantes	
Copulativas	**Disyuntivas**
y	o
e	u

Ambigüedad sintáctica preposicional: se puede presentar, cuando una oración contiene una palabra de tipo preposición (*Zapata, et al., 2007*).

Es necesario aclarar que no todas las preposiciones causan ambigüedad sintáctica preposicional. A continuación, en la tabla 2, se muestran cada una de las preposiciones que generan ambigüedad sintáctica y el sentido en que son ambiguas.

Tabla 2. Preposiciones subordinantes *(Zapata, et al., 2007).*

Preposiciones	**Sentido en que son ambiguas**
a	lugar, tiempo e instrumento
con	contenido, compañía e instrumento
de	materia, pertenencia y origen

en	lugar y tiempo

A continuación, se describen otros conceptos básicos necesarios para la comprensión del dominio de la presente investigación.

Análisis sintáctico: consiste en determinar las funciones de las palabras o grupos de palabras dentro de la oración (*Zapata, et al., 2007*).

Procesamiento del lenguaje natural (PLN): trata todo tipo de fenómenos lingüísticos de forma automática, se define como una parte esencial de la Inteligencia Artificial que investiga y formula mecanismos computacionalmente efectivos y faciliten la interacción hombre-máquina (*Zapata, et al., 2007*).

Lingüística computacional: constituye un campo científico de carácter interdisciplinar, vinculado a la lingüística y a la informática, cuyo fin fundamental es la elaboración de modelos computacionales que reproduzcan distintos aspectos del lenguaje humano y que faciliten el tratamiento informatizado de las lenguas *(Pérez,2009).*

Desambiguación de sentido de palabras (WSD): es el problema de seleccionar un sentido de un conjunto de posibilidades predefinidas para una palabra dada en un texto o discurso (Ramos, 2009).

1.3. Revisión del estado del arte

Con el fin de interpretar automáticamente el lenguaje natural, se han adelantado estudios acerca de los diversos tipos de ambigüedades que puede presentar un texto, sus causas y posibles soluciones. Para corregir el problema de la ambigüedad, se han propuesto, probado e implementado algunas estrategias de desambiguación.

1.3.1. Métodos y herramientas para la desambiguación

El español es uno de los lenguajes más difíciles de tratar informáticamente, pues la mayoría de los recursos léxicos disponibles se encuentran en inglés. Actualmente existen varios métodos y herramientas para la desambiguación de un texto. Estos métodos se clasifican en: métodos estadísticos, métodos de inteligencia artificial y métodos híbridos.

Métodos estadísticos: los métodos estadísticos basan su funcionamiento en la aplicación de fórmulas y herramientas estadísticas para determinar las representaciones gramaticales correctas para un contexto. Algunos recursos utilizados por estos métodos son las matrices de probabilidad y modelos probabilísticos (*Zapata, et al., 2007*).

Métodos de inteligencia artificial: los métodos de inteligencia artificial (IA), basan su funcionamiento en conceptos y recursos de la IA, aplicados a la desambiguación. Requieren de extensos repositorios de información y utilizan recursos como las redes neuronales, ontologías, sistemas basados en conocimiento y reglas heurísticas (*Zapata, et al., 2007*).

Métodos híbridos: los métodos híbridos combinan técnicas de los métodos estadísticos y los basados en la inteligencia artificial. Se utilizan indistintamente para cada una de las fases de la desambiguación (*Zapata, et al., 2007*).

En correspondencia con los métodos citados anteriormente, a continuación, se analizan algunos de los trabajos y soluciones informáticas realizadas a partir de la aplicación de estos.

Trabajo basado en métodos estadísticos: el autor Antonio Molina, de la Universidad Politécnica de Valencia, en su tesis *"Desambiguación en procesamiento del lenguaje natural mediante técnicas de aprendizaje automático"*, propone un método supervisado de desambiguación del sentido de las palabras basado en los modelos de Markov (MM) especializados. Estos métodos utilizan matrices de probabilidades, donde cada estado corresponde a una categoría morfosintáctica[1] y el número de estados corresponde al número de categorías asociadas a una palabra. El método consiste en dos tareas fundamentales: la selección de las características relevantes para la tarea de desambiguación, mediante la definición del alfabeto de símbolos utilizado en un MM y la especialización o redefinición de los estados del modelo a partir de la información disponible en los datos de entrenamiento. Este método requiere recursos computacionales escasos para el idioma español, como es el caso de los corpus[2] anotados semánticamente (*Zapata, et al., 2007*).

Trabajo basado en técnicas de inteligencia artificial: los autores del trabajo *"Resolución de la ambigüedad mediante redes neuronales. Procesamiento del Lenguaje Natural"*, exponen un método

[1] Morfosintáctica: los niveles básicos en que se agrupan los fenómenos lingüísticos son el fónico, el morfosintáctico y el semántico"

[2] Conjunto de datos, textos u otros materiales sobre determinada materia que pueden servir de base para una investigación o trabajo.

de resolución de ambigüedad léxica basado en el Modelo de Espacio Vectorial (MEV). Cada sentido de una palabra es representado con un vector, así como el contexto de la palabra a desambiguar. Las entradas del algoritmo están representadas por los vectores, procesados mediante el algoritmo LVQ *(Learning Vector Quantization*), a través de una función de similitud se comparan los vectores que representan el contexto de cada palabra a desambiguar con cada uno de los vectores de sus sentidos. El sentido representado por el vector de mayor similitud, será el designado como sentido desambiguado (*Zapata, et al., 2007*).

Trabajo basado en métodos híbridos: los autores del trabajo *"Una aproximación para resolución de ambigüedad estructural empleando tres mecanismos diferentes"*, proponen un método de resolución de la ambigüedad estructural haciendo uso de información léxica, sintáctica y semántica. Se combinan tres técnicas: reglas ponderadas, patrones de manejo y proximidad semántica. El método contiene módulos que brindan variantes con pesos, encargados de recopilar y procesar los pesos brindados. (*Zapata, et al., 2007*).

Natural Language Toolkit (NLTK): es un paquete de herramientas y recursos libres para un espectro amplio de tareas dentro de PLN. NLTK está destinado a apoyar la investigación y la enseñanza en PLN o áreas relacionadas, que incluyen la lingüística empírica, las ciencias cognitivas, la inteligencia artificial, la recuperación de información y el aprendizaje de la máquina *(NLTK.org, 2014)*. El paquete integra diferentes tipos de corpus: texto simple, texto etiquetado con su categoría, etiquetado con sintaxis superficial, etiquetado con sintaxis profunda e incluso simples listas de palabras o léxicos. Cada corpus ofrece una serie de métodos para leer sus datos palabra por palabra, oración por oración, párrafo por párrafo o por unidades de etiquetado *(Manterola, et al., 2010).*

3LB-SAT (3LB-Herramienta de Anotación Semántica): es una herramienta para el etiquetado semántico de corpus multilingüe. Tiene como principales características: estar orientado a la palabra, permite introducir el corpus en diferentes formatos y usa *EuroWordnet*[3] para consultar el sentido de las palabras en cuatro idiomas (español, catalán, euskara e inglés). Anota las palabras monosémicas[4] automáticamente; y muestra todas las apariciones de un lema en el texto, siendo posible asociar más de un *synset*[5] con una aparición de un lema *(Bisbal, et al., 2003).*

[3] EuroWordnet: diccionario electrónico semántico que tiene como fin la construcción de una base de datos léxico-semántica para las lenguas castellano, holandés, italiano e inglés.

[4] Monosémicas: son las palabras que cuentan con un significado único.

[5] Synset: conjunto de sinónimos el cual se considera representativo de un concepto lexicalizado.

Herramienta informática para la evaluación de la ambigüedad en textos legales: basada en la definición de un conjunto de reglas gramaticales, implementadas a partir del uso de técnicas del procesamiento del lenguaje natural y la minería de textos. Permite identificar las ambigüedades presentes en documentos de la Legislación Cubana, brindar una clasificación de las mismas y evalúa cuantitativa y cualitativamente el grado de ambigüedad presente en los textos analizados, teniendo en cuenta la cantidad de oraciones que contienen *(Carrazana, 2015).* La herramienta además de detectar las ambigüedades las clasifica en léxicas, semánticas y sintácticas**.**

Las soluciones antes mencionadas poseen funcionalidades para detectar los sentidos de la palabra y hacer análisis tanto sintáctico como semántico, acciones necesarias para representar la ambigüedad en un texto y trabajar en su reducción. A pesar de esto, no son capaces de representar la ambigüedad en todos los sentidos y su alcance está restringido solo al análisis. Teniendo en cuenta estas limitaciones, se hace necesario el desarrollo de la solución informática propuesta en el presente trabajo.

1.4. Metodología de desarrollo de software

Una metodología de desarrollo de software es un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (*Sommerville, 2005*).

Las metodologías son diversas y se encuentran clasificadas en 2 grandes grupos, las tradicionales o pesadas y las ágiles o ligeras. Estas han evolucionado de manera significativa en las últimas décadas, permitiendo el éxito o el fracaso de muchos de los sistemas desarrollados para distintas áreas, por tanto, es de vital importancia que la selección se haga de manera cuidadosa, teniendo en cuenta todos los factores que puedan influir en el éxito del proyecto a realizar.

Para el desarrollo de la solución propuesta, se decide utilizar una metodología ágil, por estar en presencia de un proyecto pequeño y de corta duración, donde se incluye al cliente como parte del equipo de desarrollo. A continuación, se realiza una descripción de las metodologías ágiles más destacadas.

XP: es una metodología ágil, centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, el aprendizaje de los desarrolladores y un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y

el equipo de desarrollo. Se define como adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. *(Beck,1999)*

AUP: es una versión simplificada del Proceso Unificado de Rational (RUP). Esta describe, de manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando métodos ágiles y conceptos que aún se mantienen válidos en RUP. Esta metodología se basa en la gestión de riesgos, planteando que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Especialmente relevante en este sentido, es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos claves del software, los cuales determinan los riesgos técnicos. Consta de 4 fases: Concepción, Elaboración, Construcción y Transición *(AUP, 2016)*

Scrum: metodología ágil y flexible, que se utiliza para gestionar el desarrollo de software, realizando entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor. Por ello, está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados rápidos. Los requisitos son cambiantes o poco definidos, la innovación, la competitividad y la productividad son fundamentales *(ProyectosAgiles.org, 2014).*

Teniendo en cuenta la necesidad de integrar la herramienta propuesta con la realizada en la UCI, donde se utilizó la metodología XP, el autor de la presente investigación decide continuar utilizando la misma metodología. Otros de los motivos por los cuales es seleccionada XP, son las características del proyecto: su corta duración, los pocos integrantes del equipo y que el cliente forma parte del equipo, contribuyendo a fomentar las relaciones interpersonales y buen clima de trabajo.

1.5. Herramientas de Ingeniería del Software Asistidas por Computadoras

Las herramientas CASE comprenden un conjunto de programas de diferentes tipos, empleados para ayudar en las actividades del proceso de desarrollo de software, tales como: análisis de requisitos, modelado de sistemas, depuración y pruebas *(Sommerville, 2005).*

Visual Paradigm para UML 8.0 es una herramienta CASE multiplataforma, que permite el modelado UML (*Unified Modeling Language*). Proporciona a los desarrolladores una plataforma que permite una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. También facilita la interoperabilidad con otras herramientas CASE y con los entornos de desarrollos más usados. Permite

la generación de una amplia gama de diagramas, incluyendo diagramas de entidad-relación.(*Visual-Paradigm.com, 2015*).

Los proyectos de software según su complejidad necesitan ser modelados y documentados. Visual Paradigm ofrece una serie de diagramas que permite la documentación y modelado de los mismos, independientemente de su complejidad.

A pesar de existir otras herramientas CASE con características semejantes, estas son de uso privativo. Teniendo en cuenta lo anterior, el autor del presente trabajo decide utilizar Visual Paradigm, considerando además la existencia de una licencia UCI para su uso.

1.6. Lenguajes de programación

Un lenguaje de programación es un lenguaje artificial utilizado para definir una secuencia de instrucciones para ser procesadas en un ordenador. Está formado por un conjunto de símbolos, palabras claves y reglas gramaticales, utilizadas para construir sentencias sintáctica y semánticamente correctas *(Rodríguez, 2003)*.

Existen variados tipos de lenguajes de programación, entre los más usados se destacan C#, C++ y Java. A continuación, se describen algunos de estos:

C#: conocido como C Sharp, está orientado a objetos y fue desarrollado y estandarizado por Microsoft como parte de su plataforma .NET en 1999. Entre sus principales ventajas se destaca que está: completamente orientado a objetos, facilitando el uso de este paradigma de programación. C# soporta la encapsulación, herencia y polimorfismo, incluye mecanismos de control de acceso, gestión dinámica de la memoria del sistema para evitar el desbordamiento, posee normas de sintaxis que evitan la conversión entre tipos de datos que no son compatibles y un alto grado de manejo de excepciones *(Sharp, 2015).*

C++: diseñado a mediados de los años 80 por Bjarne Stroustrup. Es un lenguaje orientado a objetos al que se le añadieron características y cualidades de las que carecía el lenguaje C. C++ es caracterizado también como multiparadigma, pues admite la programación estructurada y la orientada a objetos. Una particularidad de este, es la posibilidad de redefinir los operadores (sobrecarga de operadores) y de poder crear nuevos tipos que se comporten como tipos fundamentales. Además permite trabajar tanto a alto como a bajo nivel *(Class, 2004).*

JAVA: es un lenguaje sencillo e independiente de plataforma creado en 1990 por Sun Microsystems. También es necesario destacar su capacidad multihilo, robustez y su integración al protocolo TCP/IP, pero su sencillez, portabilidad y seguridad, lo han hecho un importante lenguaje (Ciberaula.com, 2016).

Teniendo en cuenta que, la herramienta desarrollada en la UCI, se encuentra implementada en el lenguaje de programación Java, el autor de la presente investigación decide utilizar este lenguaje, con el propósito de mantener una documentación regular y facilitar una futura integración y mantenimiento de la herramienta.

1.7. Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado, traducido del inglés *Integrated Development Environment* (IDE), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, puede utilizarse para varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE proveen de un marco de trabajo amigable para la mayoría de los lenguajes de programación, tales como C++, Python, Java, C#, Delphi**,** Visual Basic, entre otros. (*Programaciondesarrollo.es, 2016*). Existen varios IDE entre los que se destacan:

Eclipse: nace como entorno de desarrollo para Java y se ha convertido en una plataforma que permite el desarrollo y la integración de herramientas, diseñado para ser extendido de forma indefinida a través de plugins[6]. La principal ventaja radica en que su arquitectura está diseñada de forma tal que la mayoría de las funcionalidades proporcionadas están localizadas en uno o varios *plugins* relacionados entre sí, haciendo a Eclipse una herramienta extensible. Posee buen rendimiento, pues inicia y reacciona de forma rápida y hace uso de poca memoria *(Eclipse, 2015).*

NetBeans IDE constituye una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrita en Java, pero sirve para otros lenguajes de programación. Existen además un número importante de módulos para extenderlo. Es un producto libre y gratuito, sin restricciones de uso (*Netbeans.org,2016*).

Teniendo en cuenta las características, la experiencia y el dominio sobre el uso de NetBeans, que posee el autor del presente trabajo, se selecciona este IDE en su versión 7.4, para el desarrollo de la

[6] Plugins: es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

solución informática propuesta. También se analizó la compatibilidad que tiene con Java, lenguaje de programación definido por el equipo de proyecto.

1.8. Patrones de diseño

Los patrones de diseño se aplican a un elemento específico del diseño, como un agregado de componentes, para resolver un problema determinado. Representan una descripción de las clases y objetos, comunicándose entre sí para resolver un problema de diseño general en un contexto particular. El uso de patrones posibilita estandarizar el modo en que se realiza el diseño y proporciona reusabilidad, extensibilidad y mantenimiento del código *(Pressman, 2003).*

Los patrones de diseño se pueden agrupar en dos grandes grupos; los GRASP (*General Responsibility Assignment Software Patterns*, en inglés), patrones generales de software para asignación de responsabilidades y los GOF (*Gang of Four*, en inglés), encargados de la inicialización, agrupación y comunicación de los objetos. En el capítulo 2, se describen los patrones de diseño empleados en la implementación de la herramienta propuesta.

1.9. Pruebas

Las pruebas son un elemento crítico para garantizar la calidad del software y representan una revisión final de las especificaciones del diseño y la codificación (*Pressman*, *2003*). Existen cuatro niveles de pruebas:

- **Pruebas de unidad:** se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño, ya sea un componente o un módulo del software *(Pressman, 2003).*

- **Pruebas de integración:** es una técnica para probar la arquitectura del software, mientras se aplican las pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño *(Pressman, 2003).*

- **Pruebas de sistema:** abarca una serie de pruebas diferentes, cuyo propósito principal es ejercitar profundamente el sistema de cómputo. Aunque cada prueba tiene un propósito distinto, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones adecuadas *(Pressman, 2003).*

- **Pruebas de aceptación:** una vez culminado el proceso de pruebas por parte del equipo de desarrollo, es indispensable, que el cliente verifique que el producto ha sido desarrollado con

las normas y criterios establecidos y cumple con todos los requisitos especificados *(Zapata, 2013).*

1.10. Conclusiones parciales

- El análisis de los conceptos relacionados con la desambiguación y las categorías sintácticas que originan ambigüedad estructural en un texto, posibilitaron la definición de las reglas heurísticas descritas en el capítulo 2 de la presente investigación.

- El estudio de los métodos de desambiguación permitió al autor del presente trabajo definir un método basado en inteligencia artificial utilizando reglas heurísticas, para el desarrollo de solución propuesta.

CAPÍTULO 2: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA INFORMÁTICA

2.1. Introducción

En el presente capítulo se describen los artefactos generados durante el desarrollo de la solución propuesta, a partir de la aplicación de las fases de exploración, planificación e iteraciones por entrega de la metodología seleccionada. Además, se definen los patrones de diseño y arquitectura empleada; así como la descripción del método de desambiguación propuesto.

2.2. Descripción del proceso de desarrollo

Teniendo en cuenta la metodología definida en el capítulo 1 de la presenta investigación, a continuación, se describe como fueron aplicadas las fases de exploración, planificación e iteraciones por entrega y los artefactos generados en cada una de ellas. La fase producción será descrita en el capítulo 3.

2.2.1 Fase de Exploración

En la fase de exploración se describen los requisitos funcionales y no funcionales de la herramienta a través de las historias de usuario (HU). El equipo de desarrollo también se familiariza con las tecnologías y herramientas a utilizar en el desarrollo del proyecto.

2.2.1.1 Requisitos funcionales

Los requisitos funcionales (RF) describen lo que el sistema debe hacer, así como las condiciones y capacidades que debe cumplir el software o producto en general, para que las peticiones del cliente queden satisfechas (*Sommerville, 2005*). A continuación, se listan los requisitos funcionales definidos para la herramienta propuesta:

RF1: Importar documento

RF2: Realizar desambiguación sintáctica

RF3: Realizar propuesta de desambiguación al usuario

RF4: Brindar información al usuario

RF5: Editar texto ambiguo

2.2.1.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) describen las características del software en relación a las condiciones que este debe cumplir para lograr un funcionamiento correcto. Para el desarrollo de la presente investigación se definieron los siguientes RNF:

- **Requisitos de usabilidad**

  RNF 1: La herramienta siempre tendrá visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento de sus funcionalidades

  RNF 2: El usuario no tendrá que dar más de tres clics para acceder a las funcionalidades de la herramienta.

  RNF 3: La herramienta debe proporcionar mensajes que sean informativos y orientados al usuario final.

  RNF 4: La herramienta debe contar con una interfaz amigable y con botones que tengan nombres sugerentes para que usuarios inexpertos puedan interactuar fácilmente con el software.

  RNF 5: Emplear el mismo formato de letra en toda la herramienta.

- **Requisito de rendimiento**

  RNF 6: La herramienta no debe tardar más de tres segundos en mostrar los resultados de una petición hecha por el usuario.

- **Requisito de software**

  RNF 7: Se debe tener instalada en la computadora la Máquina Virtual de Java en su versión 8 o superior y un lector de formato de documento (.pdf) para el funcionamiento de la herramienta.

  RNF 8: La herramienta puede ejecutarse en los sistemas operativos Windows, Linux y OS X.

- **Requisitos de hardware**

  RNF 9: Para ejecutar la herramienta la computadora debe contar con las siguientes características:

- ✓ 512 MB de memoria RAM.
- ✓ Procesador a 3.2 GHz, equivalente o superior.
- ✓ Capacidad mínima de 128 GB de disco duro

2.2.1.3 Historias de usuario

La HU es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas *(Canós, 2002)*.

Para el desarrollo del presente trabajo, se identificaron un total de 6 HU, a continuación, se muestran las 3 HU que presentan una prioridad alta en el negocio:

Tabla 3. HU "Realizar desambiguación sintáctica"

Historia de Usuario	
Número: 02	**Nombre de la Historia de Usuario:** Realizar desambiguación sintáctica
Modificación a la Historia de Usuario: Ninguna	
Usuario: José Wilfredo de la Paz Valdés	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 3 semanas
Programador responsable: José Wilfredo de la Paz Valdés	

Descripción: realizar la desambiguación sintáctica de un texto importado en la herramienta en el cual se hallan detectado ambigüedades de tipo coordinativa o preposicional.
Observaciones: Ninguna

Tabla 4. HU "Realizar propuesta de desambiguación al usuario"

Historia de Usuario	
Número: 03	**Nombre de la Historia de Usuario:** Realizar propuesta de desambiguación al usuario
Modificación a la Historia de Usuario: Ninguna	
Usuario: José Wilfredo de la Paz Valdés	**Iteración asignada:** 2
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1 semana
Programador responsable: José Wilfredo de la Paz Valdés	
Descripción: mostrar al usuario la propuesta de desambiguación del texto.	
Observaciones: el resultado le será mostrado al usuario en forma de propuesta, dando la posibilidad de decidir que modificaciones debe realizar sobre el documento, para lograr su desambiguación.	

Tabla 5. HU "Brindar información al usuario"

Historia de Usuario	
Número: 04	**Nombre de la Historia de Usuario:** Brindar información al usuario
Modificación a la Historia de Usuario: Ninguna	
Usuario: José Wilfredo de la Paz Valdés	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Bajo	**Puntos reales:** 1 semana
Programador responsable: José Wilfredo de la Paz Valdés	
Descripción: mostrar al usuario las razones por las cuales las oraciones señaladas son posiblemente ambiguas.	
Observaciones: su función es complementar la propuesta de desambiguación para ayudar a decidir al usuario la acción a realizar sobre el texto ambiguo.	

2.2.2 Fase de Planificación

En la fase de planificación el cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario para su implementación. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura pocos días *(Letelier, et al., 2006).*

2.2.2.1 Estimación de esfuerzo por historias de usuarios

La estimación de esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores, utilizando como medida el punto. Un punto, equivale a una semana ideal de

programación. Si la estimación supera las 3 semanas, la HU deberá ser dividida hasta que pueda ser desarrollada en un tiempo factible. En caso que el esfuerzo sea menor de una semana, la HU será combinada por otra. *(Letelier, et al., 2006).* En la siguiente tabla se muestra la estimación de esfuerzos por HU para el desarrollo de la herramienta propuesta:

Tabla 6. Estimación de esfuerzo por HU.

Historias de usuario	Puntos de estimación
Importar documento	1
Realizar desambiguación sintáctica	2
Realizar propuesta de desambiguación al usuario	1
Brindar información al usuario	1
Editar texto ambiguo	1

2.2.3 Fase de Iteraciones por entrega

Esta fase incluye las iteraciones definidas por el equipo de proyecto para organizar el desarrollo de la herramienta propuesta a través del plan de entregas. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se establece la arquitectura del sistema utilizada en el resto del proyecto. Al final de la última iteración el sistema queda listo para entrar a la fase de producción *(Letelier, et al., 2006).* A continuación, se describen cada una de las iteraciones definidas para el desarrollo de la solución propuesta:

Iteración #1: en esta iteración se implementó la HU 2, refiriéndose al requisito funcional realizar desambiguación sintáctica.

Iteración #2: en esta iteración se implementaron las HU 3 y 4, referentes a los requisitos funcionales realizar propuesta de desambiguación al usuario y brindar información al usuario.

Iteración #3: en esta iteración se implementaron las HU 1 y 5, referentes a los requisitos funcionales importar documento y editar texto ambiguo.

2.2.3.1 Plan de duración de las iteraciones

Después de tener la estimación de esfuerzo por HU, es necesario crear el plan de duración para cada una de las iteraciones definidas, que tiene como objetivo mostrar la duración y el orden en que serán implementadas las HU dentro de cada iteración. Para la implementación de la herramienta propuesta se necesita un total de 6 semanas, dividas en 3 iteraciones, como se muestra en la tabla 7.

Tabla 7. Plan de duración de las iteraciones

Iteraciones	Historias de usuario	Duración
1	Realizar desambiguación sintáctica	2 semanas
2	Realizar propuesta de desambiguación al usuario	2 semanas
	Brindar información al usuario	
3	Importar documento	2 semanas
	Editar texto ambiguo	

2.3. Diseño de la herramienta

La metodología de desarrollo de software XP propone que se realicen diseños simples y sencillos dentro de cada una de las iteraciones definidas en la fase de iteraciones por entrega, que permitan un fácil entendimiento a los desarrolladores, permitiendo reducir el tiempo de realización las tareas de implementación.

2.3.1 Arquitectura

Para el desarrollo de la presente investigación se utilizó el patrón arquitectónico programación por capas, también conocida como n-capas. Esta arquitectura permite separar la lógica de negocios de la

lógica de diseño. Su principal ventaja es que permite llevar a cabo la implementación en varios niveles y en caso de existir algún cambio, solo se realiza en el nivel requerido, sin necesidad de revisar todo el código de la herramienta propuesta.

Las capas definidas son: presentación, negocio y útiles. La capa de presentación es la que interactúa con el usuario, brindando información, realizando la captura de los datos y devolviéndole los resultados de la desambiguación. Esta capa se comunica con la de negocio y la de útiles.

La capa de negocio es la encargada de recibir la información de la capa presentación, realizar el procesamiento de los datos y enviar una respuesta a esta última. Además, contiene las funcionalidades necesarias para el proceso desambiguación sintáctica. Esta capa también se comunica con la de útiles.

La capa útiles contiene las funcionalidades comunes para todos los paquetes. A continuación, en la figura 1, se muestra la estructura por paquetes de la solución propuesta aplicando la arquitectura antes descrita.

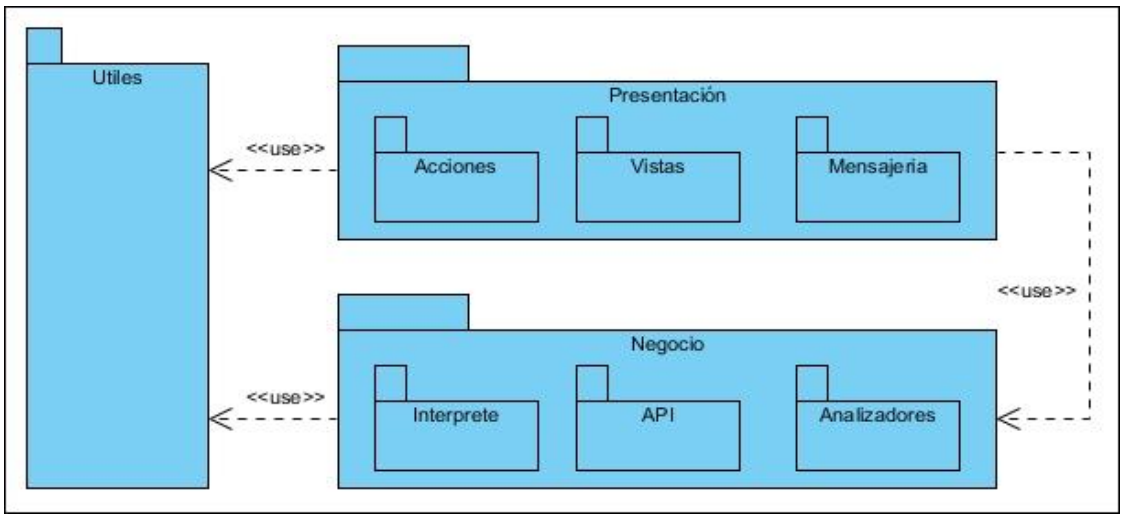

Figura 1. Diagrama de paquete (Fuente: elaboración propia).

2.3.2 Patrones de diseño

Para el desarrollo de la herramienta propuesta se emplearon un conjunto de patrones de diseño, que constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. A continuación, se describen los patrones empleados:

Patrones de diseño GRASP

Creador: se refleja en las clases que tienen la responsabilidad de instanciar objetos de otras clases. El uso de este patrón se evidencia en la clase *GestorAnalisis y CotroladorPrincipal*.

Experto: se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan. El uso de este patrón se evidencia en la clase *Parser.*

Controlador: es el experto coordinando el trabajo de otros expertos. Este patrón se evidencia en la clase *GestorAnálisis* que es la encargada de buscar los analizadores y el acceso a los recursos léxicos.

2.3.3 Estándares de codificación

El estándar de codificación empleado en el desarrollo de la herramienta fue definido por el equipo de desarrollo. A continuación, se describe el mismo:

- Todas las nomenclaturas a utilizar se definen en idioma español.
- Los nombres de los paquetes y las clases se escriben en mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escriben de igual forma.
- Los nombres de los métodos se escriben en minúscula, en caso de ser un nombre compuesto, las siguientes palabras se escriben en mayúscula.
- Los identificadores para las variables y los parámetros se identifican con letras minúsculas y en caso de ser un nombre compuesto las siguientes palabras se escriben en mayúscula.
- La clase gestora de negocio comienza con el prefijo Gestor y luego el nombre de la clase (*GestorAnalisis.java*).
- Los nombres de variables o funciones deben ser lo suficientemente descriptivos, sin exceder de 30 caracteres.
- Todas las funciones deben tener comentarios explicando su finalidad.

2.3.4 Diagrama de clases del diseño

Un diagrama de clases del diseño es una estructura estática que describe como está compuesto un sistema, mostrando las clases, atributos, operaciones y las relaciones entre los objetos que lo componen *(Larman, 2004)*. En la figura 2 se muestra el diagrama de clases del diseño definido en la presente investigación, acotado solo a sus clases y relaciones. En el Anexo 1 se observa el diagrama con sus clases, relaciones y atributos.

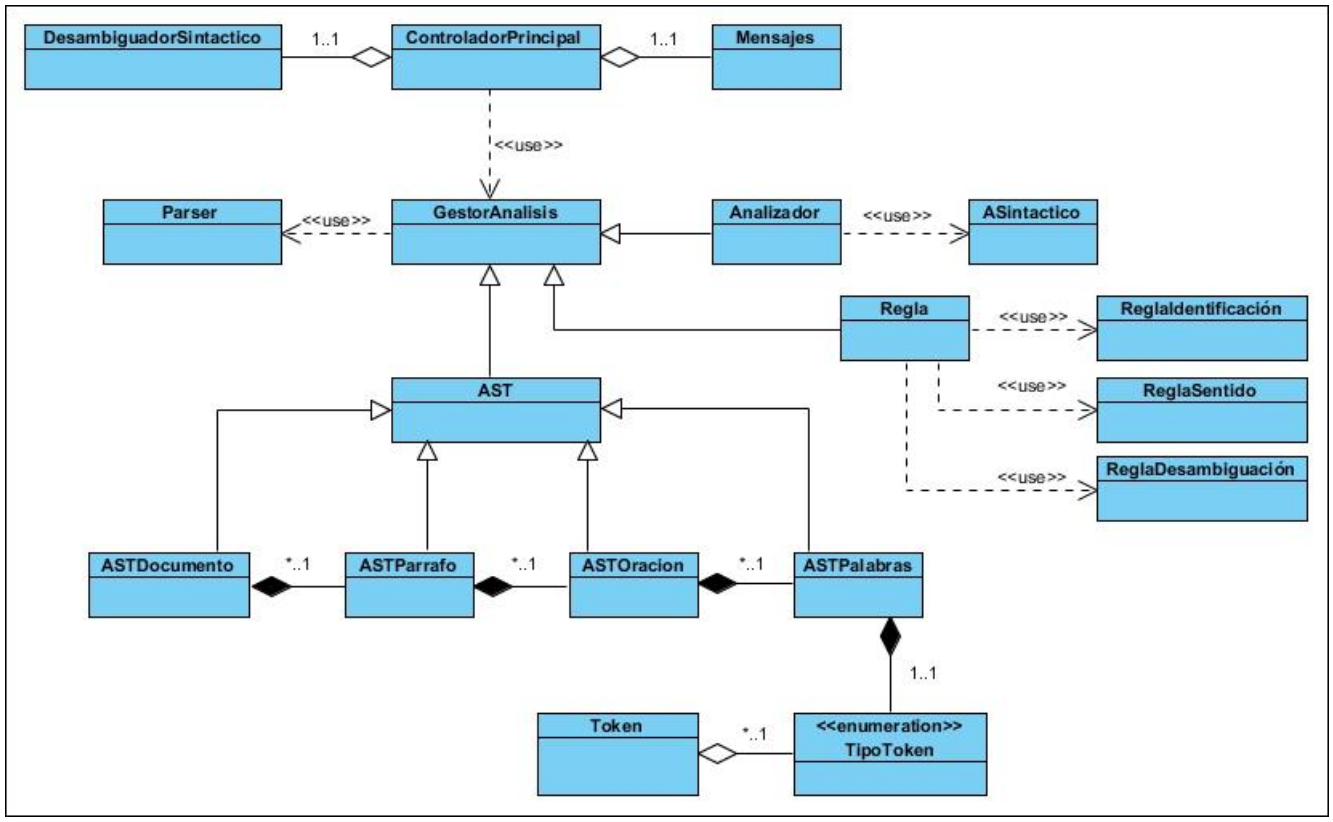

Figura 2. Diagrama de clases del diseño acotado a sus clases y relaciones (Fuente: elaboración propia)**.**

El diagrama de clases muestra las relaciones entre las clases de la herramienta propuesta. La clase *GestorAnalisis* es la encargada del negocio de la aplicación, manejando el flujo de información para identificar, determinar el sentido y desambiguar cada una de las oraciones. La clase *ControladorPrincipal* es la encargada de controlar la vista y mostrar la información al usuario. La clase *Analizador* se ocupa de realizar el análisis sintáctico inicial necesario para la identificación de las ambigüedades. Por otra parte, la clase *Parser* genera los árboles sintácticos asociadas a las oraciones, mientras que la clase *Regla* evalúa que reglas se cumplen en cada fase del proceso.

2.4. Método de desambiguación

Teniendo en cuenta las características de los métodos de desambiguación analizados en el capítulo 1, el autor del presente trabajo decide utilizar un método basado en inteligencia artificial. La propuesta se realiza con el propósito de disminuir el número de representaciones sintácticas en una oración, cuando presenta ambigüedad estructural originada por conjunciones y/o preposiciones. El mismo se divide en cuatro fases: identificación, determinación del sentido, desambiguación y resultados. El desarrollo de las tres primeras fases, está basado en un conjunto de reglas heurísticas definidas a partir de las características de las categorías sintácticas (conjunciones y preposiciones) que originan

ambigüedad estructural en una oración, analizadas en el capítulo 1. A continuación, se detallan cada una de las fases y las reglas definidas.

Fase de identificación de la ambigüedad:

Se realiza el análisis sintáctico de la oración para determinar la función de cada una de las palabras que la conforman. Una vez terminado el análisis sintáctico, se procede a aplicar un primer grupo de reglas heurísticas, con el propósito de detectar y clasificar las ambigüedades presentes en la oración. A partir de los resultados obtenidos, se aplica la fase de determinación del sentido o se puede pasar directamente a la fase de desambiguación, como se muestra en la figura 3.

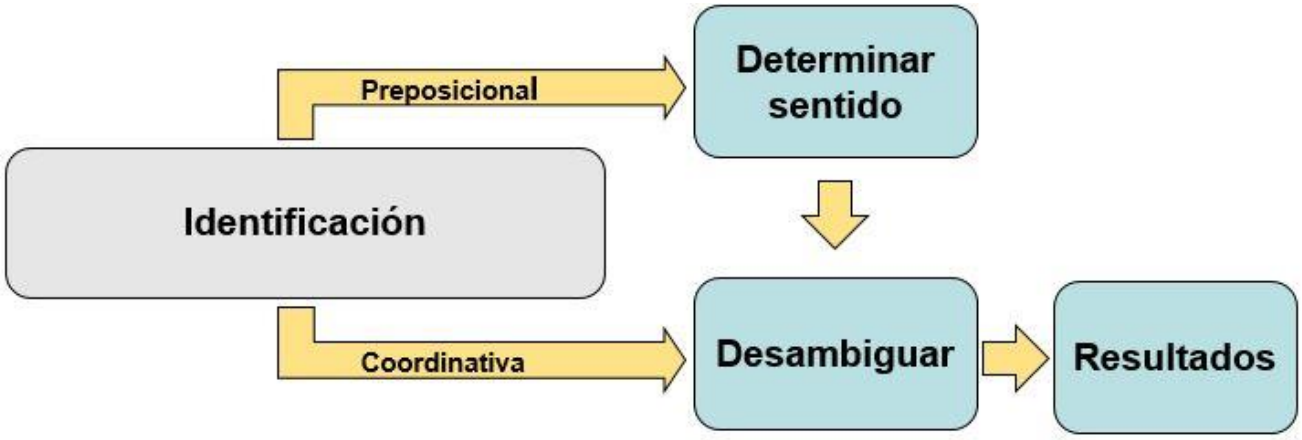

Figura 3. Fases del método de desambiguación (Fuente: elaboración propia).

A continuación, se listan las reglas heurísticas definidas para la fase de identificación de las ambigüedades:

Regla 1: si una oración contiene más de una conjunción sintácticamente ambigua y estas conjunciones pertenecen al grupo de conjunciones coordinantes copulativas, entonces la oración presenta ambigüedad coordinativa copulativa.

Regla 2: si una oración contiene más de una conjunción sintácticamente ambigua y estas conjunciones pertenecen al grupo de conjunciones coordinantes disyuntivas, entonces la oración presenta ambigüedad coordinativa disyuntiva.

Regla 3: si una oración contiene más de una conjunción sintácticamente ambigua y estas conjunciones pertenecen al grupo de conjunciones coordinantes disyuntivas o al grupo de conjunciones coordinantes copulativas entonces la oración presenta ambigüedad coordinativa mixta.

Regla 4: si una oración contiene al menos una preposición separable sintácticamente ambigua entonces la oración presenta ambigüedad preposicional.

Fase de determinación del sentido:

Si en la primera fase se detectaron ambigüedades de tipo preposicional, es necesario antes de pasar a la fase de desambiguación, determinar el sentido de las preposiciones. Para desarrollar esta fase se definieron un segundo grupo de reglas heurísticas, que permiten extraer información semántica de la oración analizada.

Reglas para la extracción de información semántica:

Si se cumple la **Regla 4:**

Regla 5: se identifica la preposición que ha generado la ambigüedad sintáctica.

Regla 5.1: si la preposición es “a”:

Regla 5.1.1: si la preposición se encuentra sucedida por un sintagma nominal cuyo núcleo sea un sustantivo que puede representar locación, entonces el sentido de la preposición es de lugar.

Regla 5.1.2: si la preposición se encuentra sucedida por un sintagma nominal cuyo núcleo sea un sustantivo que actúe o represente un punto en el tiempo, entonces el sentido de la preposición es de tiempo.

Regla 5.1.3: si la preposición se encuentra sucedida por un sintagma nominal cuyo núcleo sea un sustantivo que actúe o represente un medio o instrumento, entonces el sentido de la preposición es de instrumento.

Regla 5.2: si la preposición es “con”:

Regla 5.2.1: si la preposición se encuentra sucedida por un sustantivo que represente un instrumento, entonces el sentido de la preposición es de instrumento.

Regla 5.2.2: si la preposición se encuentra sucedida por un sustantivo que actúe o represente una persona u objeto animado, entonces el sentido de la preposición es de compañía.

Regla 5.2.3: si la preposición se encuentra sucedida por un sustantivo que actúe o represente una sustancia o material, entonces el sentido de la preposición es de contenido.

Regla 5.3: si la preposición es “de”:

Regla 5.3.1: si la preposición se encuentra sucedida por un sustantivo que represente un tipo de material, entonces el sentido de la preposición es de materia.

Regla 5.3.2: si la preposición se encuentra sucedida por un sustantivo que actúe o represente una persona u objeto animado, entonces el sentido de la preposición es de pertenencia.

Regla 5.3.3: si la preposición se encuentra sucedida por un sustantivo que actúe o represente un lugar, entonces el sentido de la preposición es de origen.

Regla 5.4: si la preposición es “en”:

Regla 5.4.1: si la preposición se encuentra sucedida por un sustantivo que represente un lugar, entonces el sentido de la preposición es de lugar.

Regla 5.4.2: si la preposición se encuentra sucedida por un sustantivo que represente un punto en el tiempo, entonces el sentido de la preposición es de tiempo.

Fase de desambiguación:

En esta fase se modifican las oraciones posiblemente ambiguas, aplicando la regla de desambiguación correspondiente al tipo de ambigüedad detectada en la fase de identificación.

Si se cumplen las **Reglas 1, 2 o 3** entonces:

Regla 6: se modifica la oración posiblemente ambigua, sustituyendo por comas todas las conjunciones consecutivas excepto la última.

Si se cumple la **Regla 4** entonces:

Regla 7: se modifica la oración posiblemente ambigua, eliminado del texto la preposición y la parte posterior a la misma.

Fase de resultados:

En esta fase se ofrecen al usuario elementos a modo de información y sugerencias, con el propósito de esclarecer el motivo de las posibles ambigüedades sintácticas identificadas en el texto, así como brindar una propuesta de desambiguación. A partir de las sugerencias indicadas por la herramienta, el usuario puede decidir si aplica o no modificaciones sobre las posibles ambigüedades señaladas en el texto.

Las informaciones y sugerencias ofrecidas al usuario parten de la fase de identificación, indicando las causas que hacen posiblemente ambiguas las oraciones señaladas en el texto. Como segundo elemento, si el tipo de ambigüedad identificada es preposicional, se indica la necesidad de verificar el sentido que toma la preposición. Mediante el análisis sintáctico, se ofrece como tercer elemento, información al usuario sobre la posible existencia de errores de redacción en el texto. Como cuarto y último elemento, la herramienta brinda al usuario una propuesta de desambiguación de las oraciones señaladas como posiblemente ambiguas.

2.5. Descripción del sistema

La herramienta informática propuesta permite identificar y corregir la existencia de términos sintácticamente ambiguos en textos de la Legislación Cubana. La misma es capaz de señalar las oraciones, en un texto previamente importado, donde existe ambigüedad sintáctica. Además, brida una propuesta de desambiguación al usuario, teniendo en cuenta los elemento descritos en el epígrafe anterior. Si el usuario entiende que la solución propuesta no le es factible, puede editar el texto a su conveniencia.

2.6 Conclusiones parciales

- El empleo de la metodología XP en función de describir el proceso de desarrollo de la herramienta, facilitó la realización de un trabajo organizado y estructurado, generando los artefactos necesarios en cada fase, en correspondencia con el cronograma de desarrollo propuesto.

- El uso de una arquitectura n-capas y el empleo de patrones de diseño, garantizaron obtener una solución de software con poca dependencia entre clases, flexible al mantenimiento y la aceptación de cambios.

- El método de desambiguación propuesto a partir de la definición de reglas heurísticas constituye la base para la obtención de la herramienta informática que responde al objetivo general de la presente investigación.

CAPÍTULO 3: PRUEBAS A LA HERRAMIENTA INFORMÁTICA

3.1 Introducción

En el presente capítulo se describe el proceso de validación de los requisitos y el resultado de la aplicación de métricas al diseño de la solución propuesta. En el capítulo se aplica la fase de producción de la metodología XP, en la cual se le realizan pruebas a la herramienta desarrollada antes de ser entregada al cliente. Además, se exponen los artefactos obtenidos en esta fase, tales como: el acta de liberación y el acta de aceptación de la solución de software.

3.2 Técnica de validación de los requisitos

Con el objetivo de demostrar que los requerimientos previamente definidos cumplen con las expectativas del cliente, se aplicaron las técnicas de validación de requisitos:

- **Revisiones formales de los requisitos:** se realizaron revisiones formales a cada uno de los requisitos por parte del cliente y del equipo de desarrollo, obteniéndose un total de 4 no conformidades de tipo técnica, 3 de redacción, 2 de ortografía y 2 de formato, las que fueron corregidas en tiempo, generándose un Acta de Aceptación por parte del cliente (ver Anexo 2).

- **Construcción de prototipos:** se confeccionaron prototipos no funcionales, dando la posibilidad al cliente de poder comprobar de forma visual cómo quedaría la herramienta, obteniéndose buena aceptación del cliente.

3.3 Validación del diseño

Para comprobar la calidad del diseño, se emplearon las métricas de diseño relaciones entre clases (RC) y tamaño operacional de clase (TOC).

3.3.1 Relaciones entre clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase *(Pressman, 2002).*

A continuación, se explican los pasos llevados a cabo para aplicar la métrica:

1. Determinar la cantidad de relaciones de uso (CRU) que poseen las clases a medir.

2. Calcular el promedio de las CRU.

3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 8.

Tabla 8. Rango de valores para medir la afectación de los atributos de calidad (RC).

Atributos de calidad	**Clasificación**	**Criterio**
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
Complejidad de mantenimiento	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio

En la siguiente figura se muestra el resultado de la aplicación de la métrica RC:

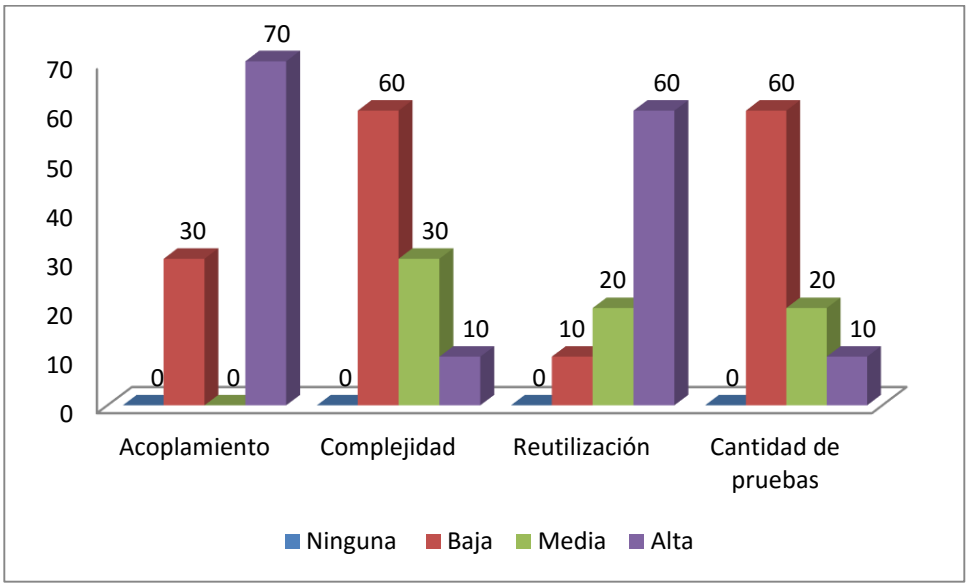

Figura 4. Representación en (%) de los resultados de la aplicación de la métrica RC (Fuente: elaboración propia).

Acoplamiento: según los resultados mostrados en la Figura 4, el 30% de las clases posee un bajo acoplamiento y el otro 70% presenta un alto acoplamiento.

Complejidad de mantenimiento: según los resultados que se muestran en la figura anterior, el 60% de las clases se comportan de forma satisfactoria pues, son de fácil soporte.

Reutilización: según los resultados que se mostraron en la Figura 4, el 60% de las clases tienen un alto grado de reutilización.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el 60% de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

A partir del análisis de los resultados mostrados en la Figura 4, se concluye que los atributos de calidad del diseño evaluados en las clases de la herramienta propuesta, a través de la métrica RC, se comportan de forma satisfactoria. La complejidad de mantenimiento y la cantidad de pruebas son bajas, en consecuencia, el grado de reutilización es alto. Es importante señalar que a pesar de que las clases del diseño presentan alto acoplamiento, este valor no implica un incorrecto diseño de las clases, teniendo en cuenta las características del negocio.

3.3.2 Tamaño Operacional de clases (TOC)

La métrica TOC es aplicada a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización *(Pressman, 2002).*

A continuación, se explican los pasos llevados a cabo para aplicar la métrica:

1. Cálculo del umbral. El umbral se toma del tamaño general de una clase, este constituye la suma de todas las operaciones que posee la clase.
2. Calcular el promedio de los umbrales.
3. Teniendo en cuenta los valores antes obtenidos, se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 9.

Tabla 9. Rango de valores para medir la afectación de los atributos de calidad (TOC).

Atributos de calidad	**Clasificación**	**Criterio**
Responsabilidad	Baja	Umbral<= Promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral > 2* promedio
Complejidad de implementación	Baja	Umbral <= Promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral > 2* promedio
	Baja	Umbral > 2* promedio

Reutilización	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral <= Promedio

En la siguiente figura se muestra el resultado de la aplicación de la métrica TOC:

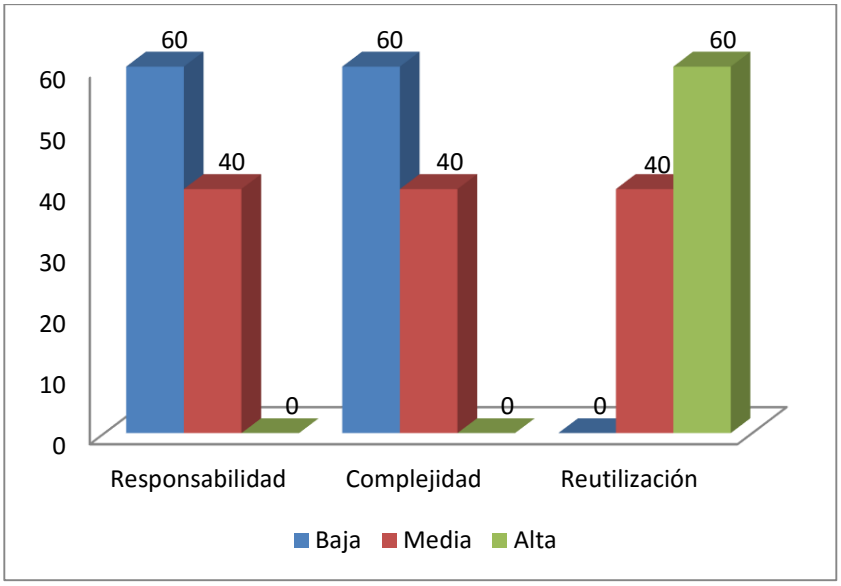

Figura 5. Representación en (%) de los resultados de la aplicación de la métrica TOC (Fuente: elaboración propia).

Responsabilidad: luego de aplicar la métrica se obtuvieron resultados satisfactorios que reflejan una responsabilidad baja con un valor del 60%.

Complejidad de Implementación: después de haber realizado la medición de la métrica, arrojó también resultados positivos ya que la complejidad de las clases es baja en un 60 %.

Reutilización: Los valores de la Figura 5 muestran un nivel alto de reutilización de las clases, 60%.

A partir del análisis de los resultados mostrados en la Figura 5, se concluye que los atributos de calidad del diseño evaluados en las clases de la herramienta propuesta, a través de la métrica TOC, se comportan de forma satisfactoria. Las clases del diseño presentan una alta reutilización, así como baja complejidad y responsabilidad.

3.4 Pruebas

En todo proceso de desarrollo de software es indispensable la realización de pruebas para garantizar el buen funcionamiento y la calidad del producto final. Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de pruebas, al realizarse de manera continua, garantizan que los errores sean detectados en un corto plazo de tiempo y se corrijan de manera más sencilla, permitiendo asegurar el éxito del producto. Existen diferentes niveles de pruebas, entre ellos se destacan las pruebas de unidad y las de aceptación.

3.4.1 Pruebas de unidad

Al desarrollar un nuevo software la primera etapa a considerar en la realización de pruebas, es el nivel de unidad, también llamadas pruebas modulares, pues permiten determinar si un módulo del programa está listo y correctamente terminado. Estas no se deben confundir con las pruebas informales que realiza el programador mientras está desarrollando *(Oré, 2009).* Como parte de las pruebas de unidad aplicadas a la herramienta obtenida, se utilizaron los métodos de caja blanca y caja negra.

3.4.1.1 Método de caja blanca

El método de caja blanca, garantiza que se ejerciten por lo menos una vez todos los caminos independientes del código, así como la ejecución de todos los bucles en sus límites operacionales y todas las decisiones lógicas en las vertientes verdaderas y falsas *(Pressman, 2003).* Para aplicar este método a todas las funcionalidades de la herramienta, se utiliza la técnica de camino básico. A continuación, se toma como ejemplo la funcionalidad *detectarAmbiguedad*, perteneciente a la clase *GestorAnalisis* para explicar el desarrollo de la técnica. La selección de esta funcionalidad se realiza teniendo en cuenta su complejidad, la cual responde a uno de los principales requisitos funcionales de la solución. La Figura 6 muestra el fragmento de código de la funcionalidad seleccionada.

```
public LinkedList<Ambiguedad> detectarAmbiguedad(String texto, int[] cantidad_oraciones) throws IOException {

    LinkedList<Ambiguedad> ambiguedades = new LinkedList<>();
    Sintactico sintactico = new Sintactico(texto, ambiguedades);
    ASTDocumento documento = sintactico.documento();
    List<ASTParrafo> parrafos = documento.getParrafos();
    for (ASTParrafo parrafo : parrafos) {
        List<ASTOracion> oraciones = parrafo.getOraciones();
        for (ASTOracion oracion : oraciones) {
            for (int i = 0; i < listaAnalizadores.size(); i++) {
            listaAnalizadores.get(i).detectarAmbiguedad(oracion, ambiguedades);

            }

        }
    }
    cantidad_oraciones[0] = documento.cantidadOraciones();
    return ambiguedades;
}
```

Figura 6. Código de la funcionalidad *detectarAmbiguedad* (Fuente: elaboración propia).

A continuación, se describen los pasos para aplicar la técnica de camino básico.

1. **Confeccionar el grafo de flujo:** usando el código de la Figura 6, se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:
   - Nodos: son círculos que representan una o más sentencias procedimentales.
   - Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
   - Regiones: son las áreas delimitadas por aristas y nodos.

En la Figura 7 se muestra el grafo de flujo obtenido.

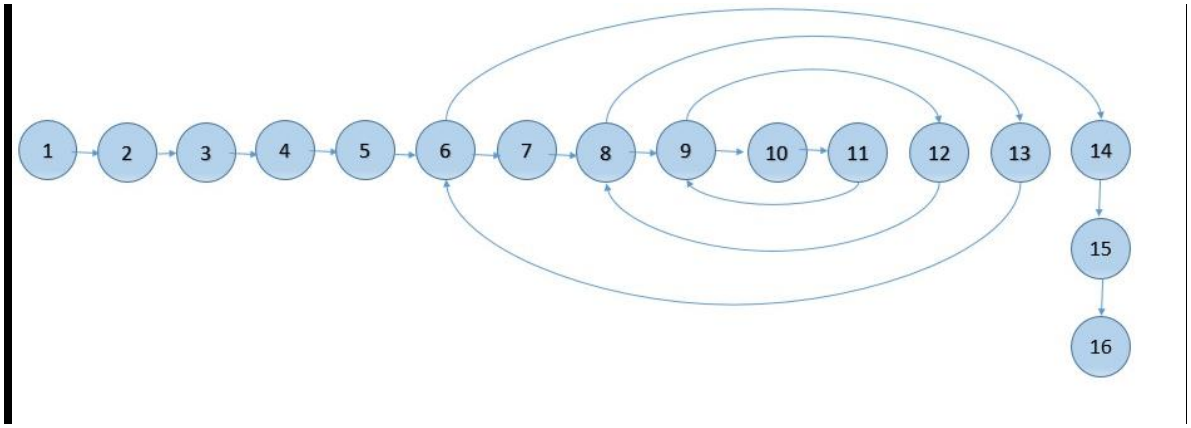

Figura 7. Grafo del camino básico del código de la funcionalidad *detectarAmbiguedad* (Fuente: elaboración propia).

2. **Calcular la complejidad ciclomática:** proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de caminos de un programa. La complejidad ciclomática se calcula a través de la siguiente fórmula:
   - V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos.

   V (G) = A-N+2 = 18-16+2 = 4

3. **Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) brinda el número de caminos linealmente independientes de la estructura de control del programa, definiéndose 4 caminos:

   Camino básico #1: 1-2-3-4-5-6-14-15-16
   Camino básico #2: 1-2-3-4-5-6-7-8-13-6-14-15-16
   Camino básico #3: 1-2-3-4-5-6-7-8-9-12-8-13-6-14-15-16
   Camino básico #4: 1-2-3-4-5-6-7-8-9-10-11-9-12-8-13-6-14-15-16

4. **Obtención de casos de pruebas (CP):** cada camino independiente representa un caso de prueba a realizar. En este caso se obtienen 4 caminos básicos, por tanto, se hace necesario la confección de igual número de CP para aplicar las pruebas a esta funcionalidad, donde los datos introducidos en el CP indican las sentencias vinculadas a cada nodo del camino. A continuación, en la tabla 10, se muestra el CP diseñado para el camino básico # 4:

Tabla 10. Caso de prueba para el camino básico #4.

Caso de prueba: Camino básico #4	
Entrada	El texto que va a ser analizado.
Resultados esperados	Una lista con las ambigüedades presentes en el texto.
Condiciones	Debe haber un párrafo con al menos una oración ambigua y debe haber algún analizador implementado en la lista de analizadores.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no presenta ciclos infinitos y no existe código innecesario en el sistema desarrollado.

3.4.1.2 Método de caja negra

El método de caja negra se aplica con el propósito de comprobar las funcionales del software, a través de la técnica de partición equivalente y valores límites, utilizando CP, con el objetivo de encontrar errores de funciones incorrectas, ausentes o de interfaz. A continuación, en la tabla 11 se muestra el CP definido para la HU “Importar documento”.

Tabla 11. Caso de prueba de la HU “Importar documento”

Escenario	**Descripción**	**Documento**	**Respuesta del sistema**
EC 1.1 Importación correcta.	Importar texto de un documento con una extensión correcta.	V Formato de documento *.doc, *.docx, *.odt y *.pdf.	Muestra el texto.
		I	

EC 1.2: Importación incorrecta.	Importar texto de un documento con una extensión incorrecta.	Solo admite el formato de documento *.doc, *.docx, *.odt y *.pdf.	El archivo seleccionado no tiene una extensión válida.

Con el objetivo de comprobar que las funcionalidades de la herramienta se ejecutan correctamente y responden a las necesidades del cliente, se realizaron pruebas funcionales por parte de los especialistas del grupo de Calidad del Centro de Gobierno Electrónico. Las pruebas se realizaron en dos iteraciones. En la primera se detectaron un total de 9 No Conformidades (NC), clasificadas en 5 de funcionalidad, 2 de redacción, 1 ortografía y 1 de diseño de interfaz, al finalizar la iteración todas las NC quedaron resueltas. En la segunda iteración los resultados fueron satisfactorios, obteniéndose cero NC, generándose así el Acta de Liberación Interna de Productos de Software (ver Anexo 3). La figura 8 ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta los tipos de NC identificadas (funcionalidad, ortografía, redacción y diseño de interfaz).

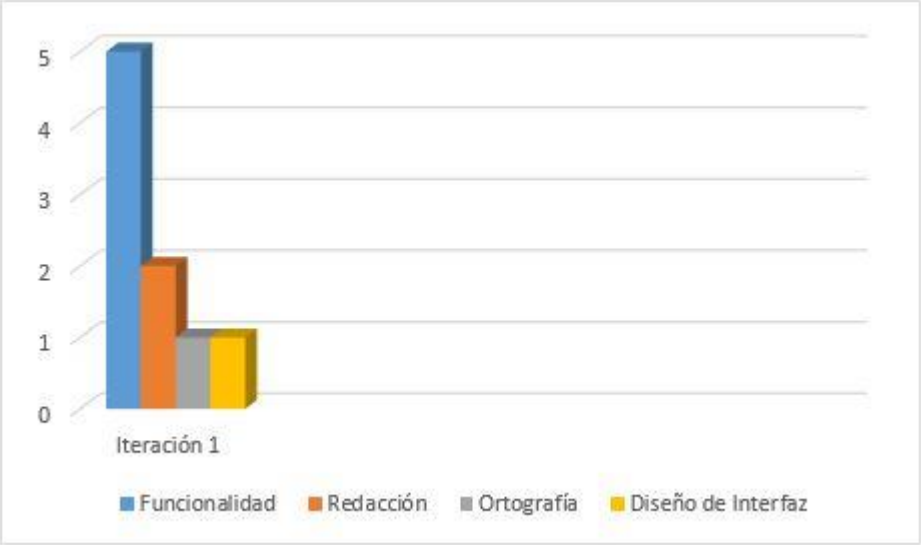

Figura 8. No conformidades detectadas al aplicar el método de caja negra (Fuente: elaboración propia).

3.4.2 Pruebas de aceptación

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o algún especialista funcional que haya tenido participación en la definición de los requisitos funcionales. Estas pruebas van dirigidas a determinar cómo el sistema cumple con los requisitos y satisface las

necesidades del cliente. Está considerada como la fase final del proceso, para crear un producto confiable y apropiado para su uso (Pressman, 2010).

Para la aplicación de estas pruebas, se confeccionaron CP de aceptación por HU. A continuación, en la tabla 12 se muestra el diseño del CP de aceptación de la HU “Realizar desambiguación sintáctica”.

Tabla 12. Caso de prueba de Aceptación de la HU “Realizar desambiguación sintáctica”.

<table>
<tr><td colspan="3">Caso de prueba de aceptación</td></tr>
<tr><td colspan="2">Código de caso de prueba: 02</td><td>Nombre historia de usuario: Realizar desambiguación sintáctica.</td></tr>
<tr><td colspan="3">Nombre de la persona que realiza el caso de prueba: José Wilfredo de la Paz Valdés</td></tr>
<tr><td colspan="3">Descripción de la prueba: revisar a través de la herramienta el correcto funcionamiento del RF: Realizar desambiguación sintáctica.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: se debe tener un texto previamente importado en la herramienta.</td></tr>
<tr><td colspan="2">Entrada/Pasos de ejecución</td><td rowspan="2">Resultados esperados:</td></tr>
<tr><td>Acción:</td><td>Entrada:</td></tr>
<tr><td>Se selecciona una de las siguientes opciones: “Procesar texto”</td><td>Texto.</td><td>El sistema es capaz de obtener el árbol sintáctico con una representación no ambigua de las oraciones del texto analizado.</td></tr>
<tr><td colspan="3">Evaluación de prueba: Satisfactoria</td></tr>
</table>

Se realizó un encuentro con la MSc. Yarina Amoroso Fernández, presidenta de la Sociedad Cubana de Derecho e Informática, con el objetivo de revisar la herramienta, teniendo en cuenta los CP definidos. Los resultados obtenidos fueron satisfactorios, avalados por la especialista entrevistada como un aporte importante para el desarrollo de la informática jurídica en Cuba. Al finalizar el encuentro se generó el Acta de Aceptación del Producto (ver Anexo 4).

3.5 Aporte de la investigación

La investigación realizada se originó a partir de la siguiente problemática: la presencia de ambigüedad sintáctica en los textos de la Legislación Cubana, hace que estos se caractericen por la existencia de dificultades lingüísticas que posibilitan la ocurrencia de ambigüedad en la interpretación y comprensión de los términos, provocando inconsistencia en el análisis de los contenidos y por tanto una baja compresión de los textos. Con la implementación de la *Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana,* se reducen cada uno de los problemas lingüísticos que originaron el presente trabajo, cumpliendo, además con el objetivo general de la investigación. A continuación, se demuestra a través de un ejemplo cómo la herramienta desarrollada responde a lo anterior.

Ejemplo para validar el cumplimiento del objetivo general de la investigación a través de la herramienta desarrollada:

Teniendo en cuenta la siguiente porción del texto de la Constitución de la República de Cuba, se demuestra cómo la herramienta es capaz de identificar las posibles ambigüedades presentes en el mismo y brindar una propuesta de desambiguación al usuario. La figura 9 muestra el análisis realizado por la herramienta sobre el texto antes mencionado.

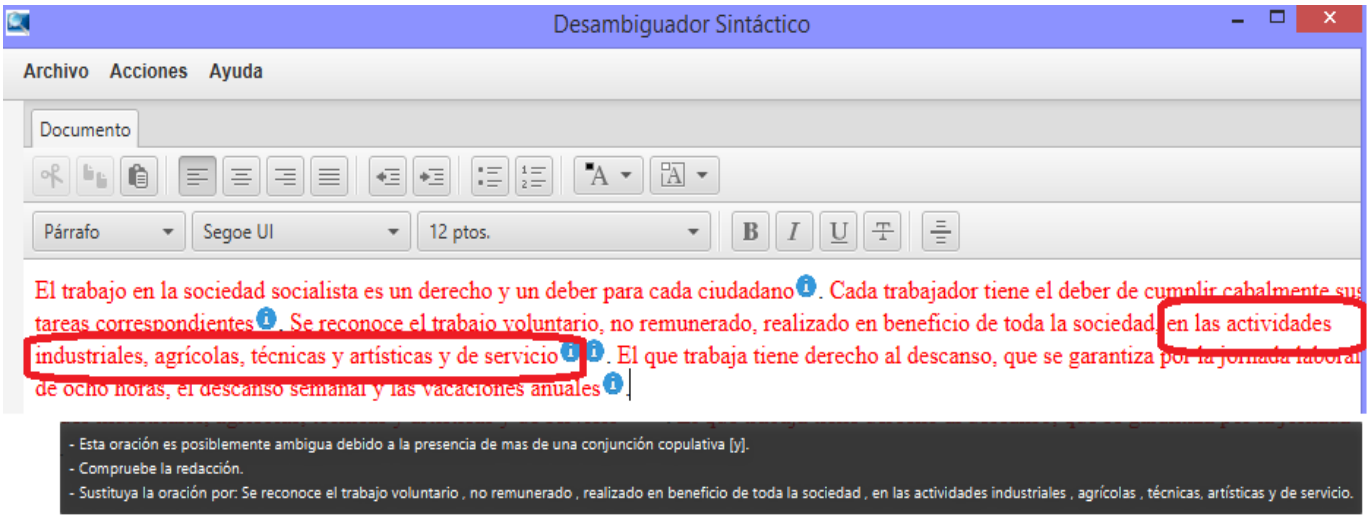

Figura 9. Análisis del texto ambiguo a través de la herramienta (Fuente: elaboración propia).

En la imagen se observa cómo en la parte del texto donde se plantea *"en las actividades industriales, agrícolas, técnicas y artísticas y de servicios"*, existe ambigüedad coordinativa copulativa causada por el uso de la conjunción *y, de forma* consecutiva. Mostrándose cómo la herramienta brinda la información necesaria al usuario, para que este modifique el texto ambiguo. En la figura 10 se observa el texto procesado por la herramienta, después de eliminada la ambigüedad señalada en la figura 9.

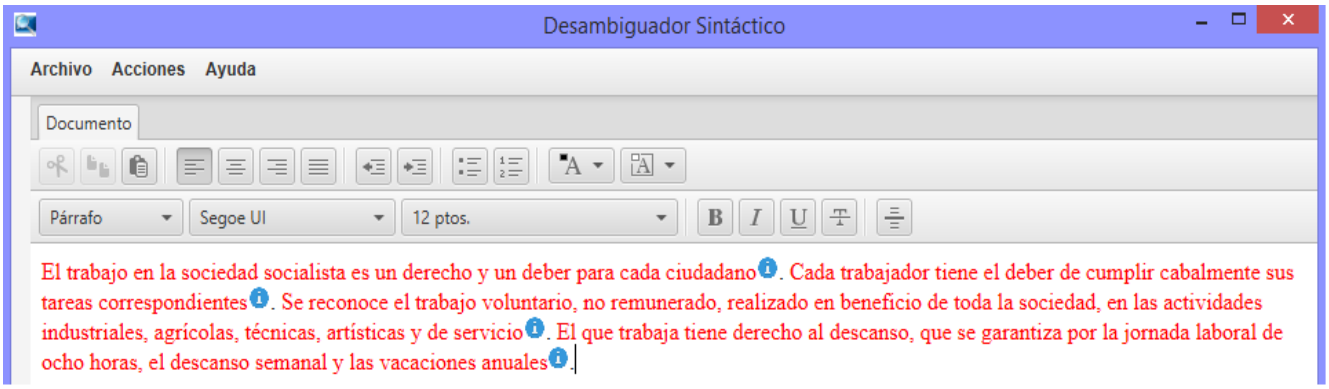

Figura 10. Análisis del texto desambiguado a través de la herramienta (Fuente: elaboración propia).

El análisis realizado anteriormente demuestra cómo la herramienta es capaz de reducir la ambigüedad en un texto, contribuyendo a la mejor compresión del mismo.

3.6 Conclusiones parciales

- Las técnicas de validación de requisitos aplicadas demostraron que los requisitos descritos a través de HU están en correspondencia con las necesidades del cliente.

- La aplicación de métricas de validación del diseño, demostró que las clases poseen una baja responsabilidad, complejidad de implementación y alta reutilización del diseño propuesto.

- La aplicación de métodos de caja negra y caja blanca, demostró la correcta implementación y ejecución del código, así como las funcionalidades de la herramienta desarrollada.

CONCLUSIONES GENERALES

- El resultado del estudio de las categorías sintácticas que originan ambigüedad estructural en un texto y los métodos de desambiguación, demostraron la necesidad de definir reglas heurísticas, que respondan a la utilización en el desarrollo de la solución propuesta de un método de desambiguación basado en inteligencia artificial.

- La herramienta informática desarrollada es capaz de identificar las ambigüedades sintácticas presentes en un texto de la legislación cubana y brindar una propuesta de desambiguación al usuario.

- La aplicación de técnicas de validación de requisitos, métricas de validación del diseño, pruebas unitarias y de aceptación a la solución propuesta, certifican la obtención de un software funcional que responde a los requerimientos del cliente y atributos de calidad del diseño, avalado en cada caso por las actas de liberación y aceptación emitidas.

- El resultado del trabajo contribuye al fortalecimiento del desarrollo de la informática jurídica en Cuba, constituyendo un resultado dentro del Grupo de Investigación de Informática Jurídica del Centro de Gobierno Electrónico.

RECOMENDACIONES

- Aumentar el número de reglas heurísticas en el método propuesto de forma tal que brinde otras propuestas de desambiguación sintáctica.
- Incorporar a la herramienta nuevas funcionalidades que permitan la desambiguación léxica y semántica.

BIBLIOGRAFÍA REFERENCIADA

Bisbal, Empar, y otros. 2003. 3LB-SAT : una herramienta de anotación semántica. [En línea] 2003. http://rua.ua.es/dspace/handle/10045/1510.

Class, Peter. 2004. Tutorial de C++: o el diario de Peter Class. [En línea] 2004. http://es.tldp.org/Manuales-LuCAS/doc-tutorial-c++/doc-tutorial-c++.pdf.

Eclipse. 2015. Eclipse. [En línea] 2015. http://www.eclipse.org/gmt/mofscript/.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

Letelier, Patricio y Penadés, Mª Carmen. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm..

Netbeans.org. 2012. Bienvenido a NetBeans y www.netbeans.org. *NetBeans.* [En línea] 2012. http://netbeans.org/index_es.html.

NLTK.org. 2014. [En línea] 2014. http://www.nltk.org.

Pressman, Roger S. 2002. *Ingeniería del Software. Un Enfoque Práctico. Quinta Edición.* España : McGraw-Hill, 2002.

ProyectosAgiles.org. 2014. Qué es SCRUM/ proyectoÁgiles. [En línea] 2014. http://www.proyectosagiles.org/que-es-scrum.

Ramos, Sulema Torres. 2009. Optimización global de coherencia en la desambiguación del sentido de las palabras. [En línea] 2009. http://www.gelbukh.com/thesis/Sulema%20Torres%20Ramos%20-%20PhD.pdf.

Rodríguez, Jesús Javier. 2003. Introducción a la programación: teoría y práctica. s.l. : Editorial Club Universitario, 2003, 9788484542742.

Software-talk.org. 2012. Netbeans vs Eclipse: An IDE comparison. *Software Talk.* [En línea] 2012. http://software-talk.org/blog/2012/01/netbeans-vs-eclipse-an-ide-comparison/.

Sommerville, Ian. 2005. *Ingeniería del software. Séptima Edición.* Madrid. España : Pearson Educación. S. A., 2005. 84-7829-074-5.

Zapata, Carlos, Palomino, Karla y Rosero, Roberto. 2007. Portal de Revistas UN. *Un método para la desambiguación sintáctica de tipo coordinativo y preposicional.* [En línea] 2007. http://www.revistas.unal.edu.co/index.php/dyna/article/view/1764/11536.

Zapata, Javier. 2013. Niveles de prueba del software. *Niveles de prueba del software.* 2013.

ANEXOS

Anexo 1: Diagrama de clases del diseño

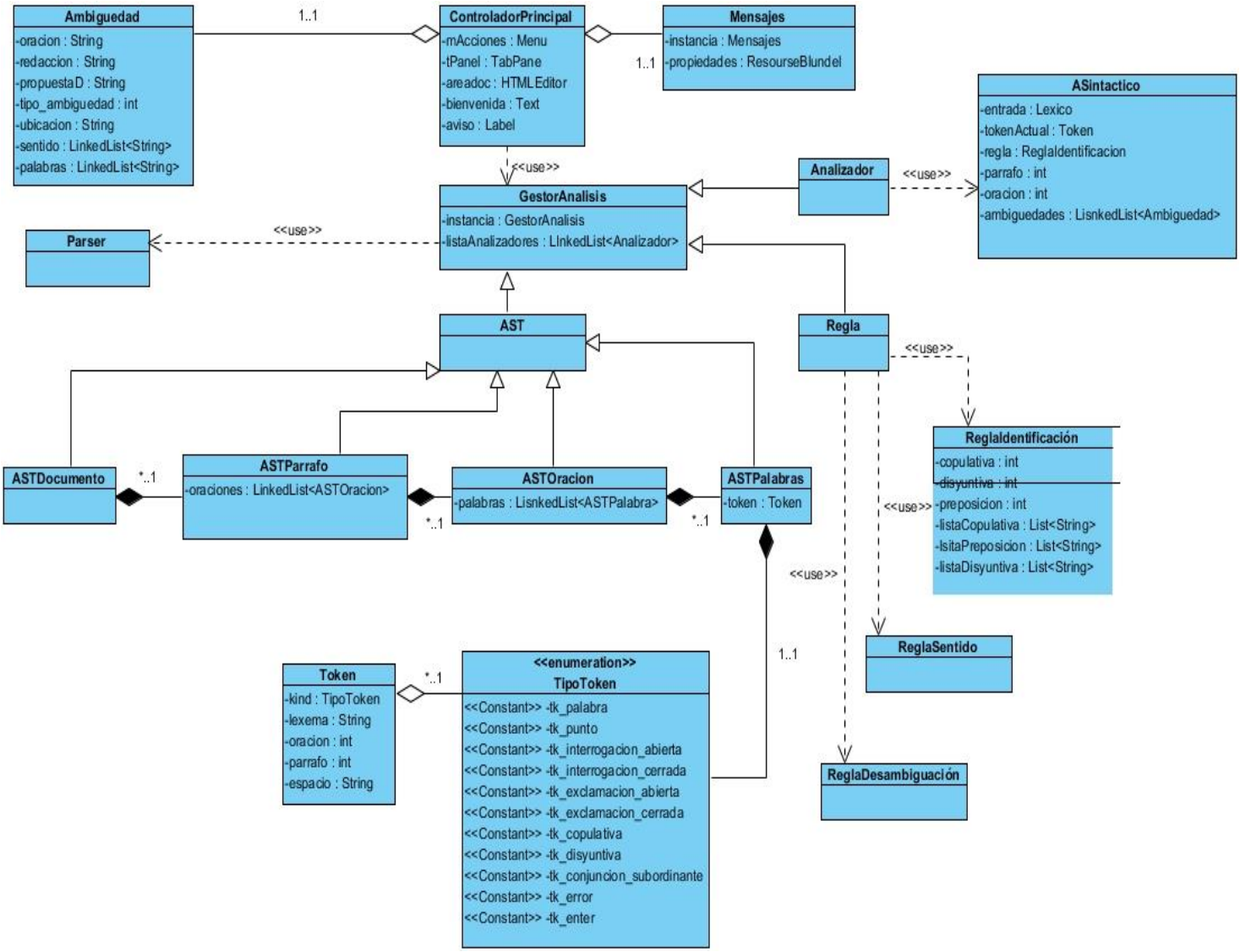

Figura 11. Diagrama de clases del diseño de la herramienta (Fuente: elaboración propia).

Anexo 2: Acta de Aceptación de los requisitos.

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

En cumplimiento del desarrollo de la Tesis: **Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana.** Se hace entrega de los productos que se relacionan a continuación:

- Especificación de Requisitos de Software
- Historias de Usuarios

La Parte Cliente, luego de haber revisado los productos de trabajo determina que los mismos cumplen con los estándares establecidos para el diseño y redacción de estos artefactos, quedando de esta forma aceptados.

Entrega	**Recibe**
Nombre y apellidos: José Wilfredo De La Paz Valdés	**Nombre y apellidos:** MSc. Yarina Amoroso Fernández
Cargo: Estudiante	**Cargo:** Presidenta de la Sociedad Cubana de Derecho e Informática

Fecha: 11/03/2016

Anexo 3: Acta de Liberación Interna de Productos de Software.

FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 28/06/2016

Emitida a favor de: Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana.

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana.	*1.0*	*0*	*2*	*Evaluación dinámica* *Pruebas de Funcionalidad*	*28/06/2016*

Ing. Yordanis Garcia Leiva
Asesor de Calidad CEGEL

Jose Wilfredo de la Paz Valdez
Autor

Ing. Jorlen Machado Suastegui
Responsable de la liberación

Centro de Gobierno Electrónico

1

Anexo 4: Acta de aceptación de la herramienta por parte del cliente.

Acta de aceptación

En cumplimiento del desarrollo de la Tesis: **Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana.** Se hace entrega del producto:

- Herramienta informática para la desambiguación sintáctica de textos de la Legislación Cubana.

La Parte Cliente, luego de haber revisado el software, considera que la solución cumple con cada uno de los requisitos que originaron su desarrollo y constituye un aporte al desarrollo de la Informática Jurídica en Cuba. Conforme a lo anterior se expresa la aceptación del mismo.

Entrega	**Recibe**
Nombre y apellidos: José Wilfredo De La Paz Valdés	**Nombre y apellidos:** MSc. Yarina Amoroso Fernández
Cargo: Estudiante	**Cargo:** Presidenta de la Sociedad Cubana de Derecho e Informática

Fecha: 28/06/2016