UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Reingeniería de un Software Para Reconocimiento de Imágenes

Memoria para optar al título de Ingeniero Civil en Computación

Felipe Antonio Krauss Benavente

Profesora Guía:
Sra. Nancy Hitschfeld-Kahler

Miembros de la Comisión
Don Alexandre Bergel
Don Benjamín Bustos

Santiago de Chile
01/04/2010

Resumen

El presente trabajo plantea la revisión de un software para el “Reconocimiento de bordes en imágenes aplicado a anillos de árboles”. Dicho producto logra satisfacer sus metas funcionales pero adolece de problemas en su diseño, lo cual provoca que la extensión del mismo se vuelva laboriosa e incluso inviable. Entre sus funcionalidades están la de aplicar distintos filtros a las imágenes, generar una malla inicial, aplicar un algoritmo de mejora en conjunto con los criterios de refinamiento y selección de un punto, y seleccionar puntos y segmentos que pueden formar parte de los anillos de los árboles.

El objetivo de esta memoria es realizar un rediseño y posterior re implementación de este software de reconocimiento de anillos de árboles, para lograr una extensión de sus funcionalidades actuales en lo que respecta al uso de polígonos como geometría inicial, la posibilidad de incorporar nuevos criterios para la mejora y selección del punto, la posibilidad de incorporar nuevos algoritmos de mejoramiento como también para la creación de la malla inicial o para la malla final. Para lo anterior se propone un proceso de desarrollo soportado en análisis y diseño por contrato.

El resultado más relevante del presente trabajo es la obtención de un producto que permite la generación automática de mallas geométricas tanto para imágenes como para geometrías poligonales convexas, que además es fácilmente extensible en los aspectos mencionados.

Dedicatoria y Agradecimientos

Esta memoria está dedicada a mi madre, su luz sigue entre nosotros.

Quisiera agradecer aquellos docentes y maestros que han guiado mis pasos para finalizar este ciclo, muy en especial a la Sra. Nancy Hitschfeld-Kahler por su orientación, apoyo y comprensión durante la elaboración de este trabajo.

También resulta imposible dejar de mencionar a Angélica y Magali por toda la paciencia y ayuda prestada durante todo el proceso de especialidad.

Finalmente agradecer a mi señora, Rayen Kelün, por su enorme soporte durante esta jornada, además de su apoyo y amor incondicional.

Índice General

Lista de Figuras

Lista de Tablas

1 INTRODUCCIÓN

"***Diseñar software orientado al objeto es difícil, y diseñar software reusable orientado al objeto es más difícil***" [1]

La ingeniería de software es una disciplina reciente, si la comparamos con ingenierías tales como la de construcción de caminos o edificaciones, u otras ciencias como la medicina. De hecho, la primera computadora completamente programable es la Z3 [2] y fue terminada en mayo de 1942, previo a esta, la computación era un área meramente teórica. En estos casi setenta años se han realizado grandes progresos, particularmente en el análisis, diseño y construcción de sistemas.

En la implementación de productos informáticos es vital comprender y documentar los requisitos que el sistema debe cumplir. Para un correcto entendimiento de las diferentes problemáticas que plantea la construcción de una solución en particular se hace necesario realizar un análisis de los requisitos o solicitudes que definen el producto.

En el caso de tecnologías orientadas al objeto, es donde se introduce el "**Object Oriented Análisis"** o sus siglas **OOA**. El **OOA** busca establecer el "modelo del problema" y las "responsabilidades del sistema" [3]. Entre ambos establecen las características que el software debe cumplir. Si bien, en el **OOA** se debe establecer un nivel detallado de compresión de los requisitos, no es su responsabilidad la definición completa del producto a construir. A este proceso se le denomina diseño [4].

En el diseño de software se modela el sistema y se encuentra su forma, entre otras cosas su arquitectura [4]. Esta forma dependerá de cuales han sido las decisiones tomadas durante el proceso y qué políticas de construcción se han seguido. Si bien el análisis nos entrega un modelo conceptual, la implementación detallada de éste puede variar mucho por diferentes factores. Entre dichos factores encontramos la arquitectura tecnológica de un proyecto, necesidades de

rendimiento de la aplicación o la misma experiencia de los participantes involucrados.

Una herramienta para la elaboración de un buen diseño son los “Patrones de Diseño”. Una definición es: “*En el mundo del software, un patrón de diseño es una manifestación tangible de una organización de la memoria tribal*” [5]. Finalmente los patrones de diseño actúan como recetas, las cuales contienen soluciones generales para problemas conocidos. Son estos patrones los que son utilizados por los arquitectos de software, tal cual lo hace un chef al modificar y mezclar sus recetas de cocina para cambiar detalles en el sabor o presentación de un plato, para elaborar diseños de software de mayor calidad.

Precisamente es este punto sobre lo que trató esta memoria, aplicar buenas prácticas de diseño de software para mejorar y depurar un producto ya existente.

El producto modificado fue un software que permite la generación y refinación de mallas geométricas. Para ello toma como punto de partida un archivo de imagen sobre la cual elabora una primera malla y luego esta es refinada según las instrucciones del usuario.

En su concepción original el producto estaba pensado para refinar cualquier malla vinculada a una geometría. Dicha geometría pudiera tener cualquier origen; ya sea desde una imagen, una polilínea u otra representación. Además se pensó que los procesos asociados a la depuración o refinamiento fueran independientes de la representación de origen.

Era parte vital de este trabajo lograr un nuevo diseño. Este debe ser extensible y robusto, logrando obtener un software que permita cumplir con los conceptos originales.

Adicionalmente se logra realizar la extensión de la implementación de forma tal de incluir no solo imágenes sino también geometrías poligonales como fuente de información para las mallas geométricas.

Finalmente, el nuevo diseño presenta un nivel de documentación el cual permita tomar y extender el producto en trabajos futuros.

1.1 MOTIVACIÓN

1.1.1 FAMILIA DE SOFTWARE

En el paper [6] se describe una “familia de software” para mallas geométricas. La finalidad de dicha familia es proveer de una herramienta que permita la generación de mallas. Estas últimas son elaboradas en un proceso de refinamiento el cual involucra el uso de una serie de componentes y la interacción entre estas.

El diseño propuesto en [6] contempla el uso de las siguientes clases para la representación de las mallas geométricas y sus propiedades:

- **Mesh**: Malla compuesta por un conjunto de triángulos.
- **Vertex**: Vértice de un triángulo.
- **Edge**: Lado de un triángulo.
- **Triangle**: Triángulo, representado por tres vértices o tres lados.
- **Geometry**: Geometría que describe el dominio sobre el que se genera una malla. Por ejemplo puede ser un polígono simple.
- **Region**: Región, representa una parte de una geometría.

Para la manipulación de las mallas se usan las siguientes componentes:

- **InitialMeshAlgorithm**: Algoritmo para la generación de malla inicial.

- **LeppBasedAlgorithm**: Algoritmo para refinar o mejorar una malla.
- **FinalMeshAlgorithm**: Algoritmo para generar malla final.
- **Criterion**: Criterio para controlar el refinamiento o mejoramiento

El proceso normal de ejecución de un generador de mallas se puede representar por el diagrama de Secuencia de la Figura 1.En él se muestra el circuito completo partiendo de la representación de una geometría hasta la generación de una malla final, incluyendo la aplicación de los diversos algoritmos.

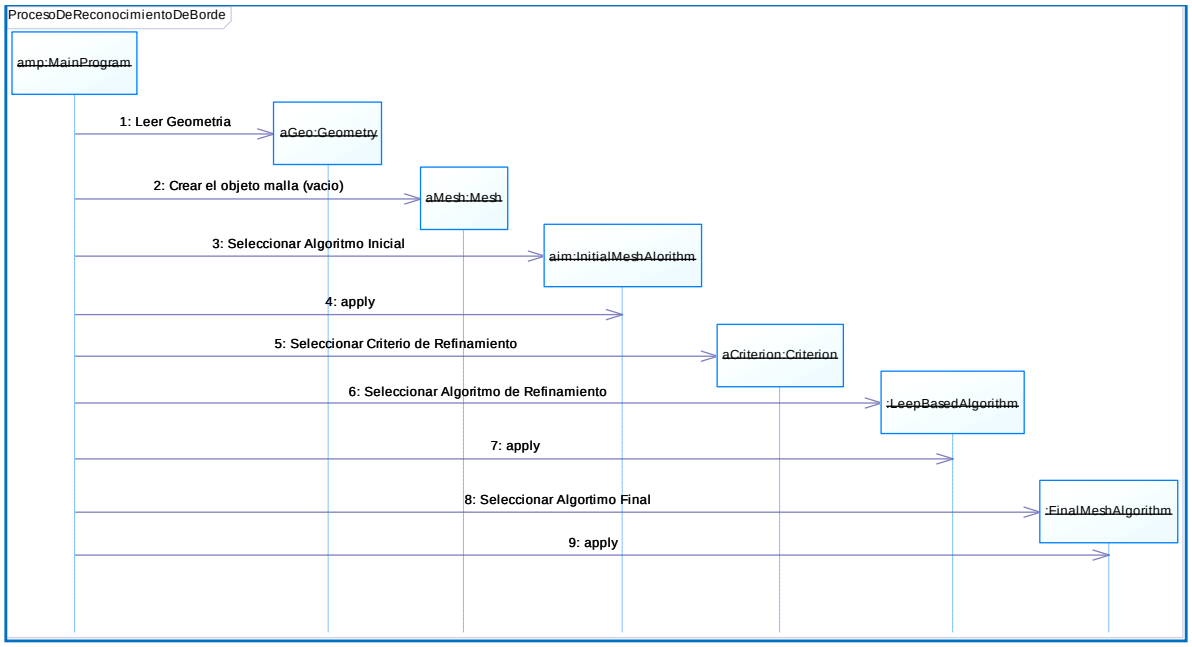

FIGURA 1 : GENERACIÓN DE MALLA

El proceso consiste en la aplicación de los siguientes pasos:

1. Obtención de una Geometría, esto puede ser realizado desde una imagen cuya geometría es el rectángulo mínimo que la rodea o ser leída desde un archivo.

2. Instanciar un objeto malla, el cual se encuentra vacío[1].
3. Seleccionar el algoritmo de malla inicial.
4. Aplicar el algoritmo de 3 sobre la geometría de 1 para generar una malla inicial. Dicha malla será almacenada en 2 para su posterior proceso de refinado.
5. Refinar
    a. Seleccionar un Criterio de refinamiento
    b. Seleccionar un algoritmo de Refinamiento.
    c. Aplicar el algoritmo de (b) usando el criterio (a). De esta forma los triángulos de la malla se dividen en triángulos más pequeños, hasta que el criterio es cumplido para todos los triángulos de la malla.
    d. Así se obtiene una malla refinada o mejorada.
    e. De existir algún criterio y/o algoritmo pendiente de aplicar se vuelve a "Refinar".
6. Seleccionar algoritmo de Malla Final.
7. Aplicar el algoritmo 6 a la malla refinada.

[1] En términos de programación es reservar memoria para el objeto.

1.1.2 SOFTWARE ACTUAL

Durante el año 2008 Pablo Aguilar realizó su memoria [7] y tesis [8] implementando un software para el reconocimiento de anillos de árboles. En este software se usan algoritmos de generación de mallas[2] para ayudar en el proceso de reconocimiento de anillos. El trabajo realizado estuvo fuertemente basado en la familia de software [6].

El producto obtenido permite aplicar el algoritmo de refinamiento de la malla pero posee ciertas limitantes. Una de ellas es la imposibilidad de generar mallas desde de una geometría poligonal; por lo tanto siempre se debe generar a partir de una imagen.

La forma de operar del producto es la que se indica:

- Solicita al usuario que seleccione una imagen y carga su información en la componente “treeangles.visibleMesh.PictureVisibleMesh” la cual representa la malla, pero al mismo tiempo almacena información de la geometría.
- Permite al usuario seleccionar filtros para la imagen, de forma de alterar la geometría de la misma.
- Permite generar una malla desde la geometría existente seleccionado alguna de las formas de criterio y selección de punto.

Desde un punto de vista general las componentes principales del sistema se vinculan como se muestra en la Figura 2, en donde GUI es la interfaz gráfica que permite manipular las imágenes para obtener las mallas geométricas.

[2] Discretizaciones espaciales.

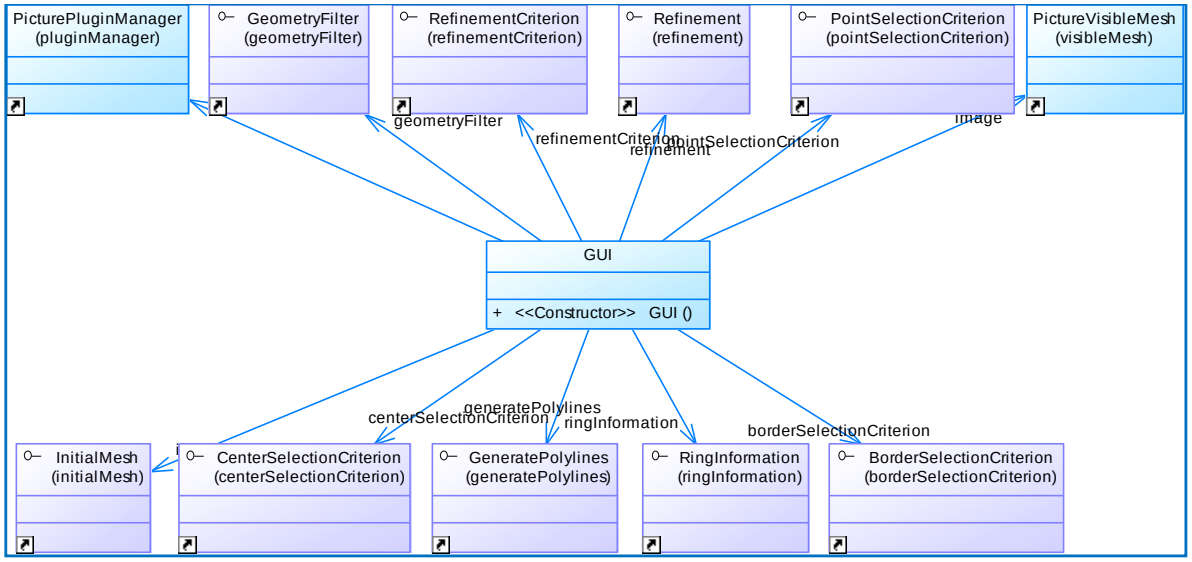

FIGURA 2 : COMPONENTES DEL SOFTWARE INICIAL

En la Tabla 1 se presenta las principales agrupaciones o paquetes y sus respectivas responsabilidades dentro del software.

Paquete	Responsabilidad
treeangles.borderSelectionCriterion	Responsable de implementar los algoritmos para la selección de los bordes de una polilínea.
treeangles.centerSelectionCriterion	Responsable de la selección del punto centro de un anillo en el reconocimiento de los mismos.
treeangles.fileFilter	Contiene las componentes gráficas responsables de filtrar el tipo de archivo que se desea cargar para iniciar el proceso de refinamiento.
treeangles.generatePolylines	Conjunto de componentes utilizadas para la generación de polilíneas en anillos de árboles.
treeangles.geometry	Paquete responsable de definir la componente “Geometría” de la familia de software.
treeangles.geometryFilter	Conjunto de filtros que permiten modificar los tonos de una imagen. Son utilizados para aplicar cierta norma a

	los valores del tono para trabajar con escalas que resalten las propiedades más importantes y minimicen el ruido de la imagen.
treeangles.initialMesh	Paquete responsable de generar mallas a partir de una geometría.
treeangles.mesh	Paquete responsable de definir la componente “Malla” de la familia de software.
treeangles.pluginManager	Componente responsable del manejo de los posibles algoritmos y filtros a ser utilizados por la interfaz gráfica.
treeangles.pointSelectionCriterion	Paquete responsable de la selección del punto por el cual se dividirá un triángulo que no cumpla con el criterio de refinamiento.
treeangles.polyline	Contiene las componentes responsables de la representación de las polilíneas.
treeangles.refinement	Paquete responsable de implementar los algoritmos de refinamiento o mejora de la familia de software.
treeangles.refinementCriterion	Paquete responsable de implementar los criterios de refinamiento de la familia de software.
treeangles.region	Contiene las componentes representación de los diferentes tipos de “Región” asociados a la familia de software.
treeangles.ringInformation	Conjunto de componentes que elaboran la representación de anillos de árboles.
treeangles.visibleMesh	Paquete responsable de realizar el dibujo de la polilínea sobre la imagen a ser depurada.

TABLA 1 : PAQUETES DE SOFTWARE

De los paquetes antes descritos se debe trabajar en la reingeniería de cinco de ellos, pues son los responsables de contener los algoritmos y criterios de refinamiento. Dichos paquetes son:

- treeangles.geometryFilter
- treeangles.initialMesh
- treeangles.pointSelectionCriterion
- treeangles.refinement
- treeangles.refinementCriterion

1.1.3 REDISEÑO Y EXTENSIÓN

En la ejecución del proceso para la obtención y refinado de una malla se escogen diversos algoritmos y criterios. Dichas decisiones son tomadas por un usuario final, por lo tanto no son predecibles y las combinaciones dependen de los requerimientos de la aplicación para la cual se está generando la malla. Esto indica que conservar una estructura de selección de criterios fija, no se condice con la extensibilidad del producto.

La necesidad central de la problemática es tener un modelo extensible que permita intercambiar indistintamente los algoritmos y criterios para mallas aplicados al reconocimiento de bordes de una imagen o para simplemente generar una malla dada una geometría de un dominio cualquiera. Lo cual hace pensar bastante en la filosofía de implementación de un "**framework**" [9].

Un "**framework**" es una herramienta que simplifica el proceso de construcción de software, entregando una serie de necesidades básicas cubiertas y permitiendo extender funcionalidad para la implementación de los requisitos del sistema.

Un "**framework**" está compuesto por "**Cold Spot**" y "**Hot Spot**". Los primeros son aquellas zonas de código las cuales no cambiarán y siempre serán parte de la instalación del "**framework**", en ellas se tienen condensado el funcionamiento inalterable y que rara vez tiene que ver con las necesidades del producto a construir. Por otro lado, los "**Hot Spot**" son aquellas piezas intercambiables y que serán construidas o configuradas dependiendo de las necesidades propias del producto, normalmente éstas son las piezas a ser desarrolladas por quienes implementan el producto "**framework**".

La filosofía de los "**Cold/Hot Spot**" permite "enchufar" y "desenchufar" distintas componentes de software, inclusive en tiempo de ejecución, sin que esto afecte la configuración del programa. De esta manera, es posible incluir nuevas componentes sin afectar las existentes. El cómo se implementa dicha

potencialidad es responsabilidad de la configuración del “**framework**” y de sus “**Cold Spot**”.

Si bien, en el ámbito de la presente memoria no se desea construir un “**framework**” para el reconocimiento de bordes en imágenes, sí se desea poder generar un mecanismo de extensibilidad del producto existente. Es por ello que se plantea el uso del mecanismo de “**Cold/Hot Spot**” para la representación de algoritmos y criterios, de manera que desarrollos actuales y futuros puedan incluir nuevas implementaciones sin variar mayormente el producto final de este trabajo.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Realizar una reingeniería del software para reconocimiento de anillo de árboles en imágenes, el cual fue obtenido como resultado de [8], con el fin de mejorar su extensibilidad y reusabilidad.

1.2.2 OBJETIVOS ESPECÍFICOS

Rediseñar la actual herramienta de software para lograr los siguientes objetivos:

- Encontrar los patrones de diseño que permitan darle mayor flexibilidad a la implementación la familia propuesta en [6]
- Aplicar los patrones de diseño encontrados en el punto anterior en el rediseño de la herramienta construida en [8].
- Modificar las estructuras de datos para permitir trabajar directo con geometrías y/o mallas geométricas sin necesidad de leer una imagen.
- Diseñar e implementar los "**Hot Spot**" que permitan utilizar diferentes algoritmos en las distintas etapas del proceso de generación de una malla:
    - Generación de una malla inicial.
    - Generación de una malla refinada / mejorada.
    - Generación de una malla final.
- Diseñar e implementar los "**Hot Spot**" para los criterios y regiones ya existentes.
- Búsqueda de algoritmos de código abierto que se puedan incorporar y adaptar a las distintas etapas del proceso de generación de mallas, construyendo sus respectivos "**Hot Spot**".

1.3 METODOLOGÍA

El software original construido en [8] utilizaba Java para su construcción y como ambiente de trabajo NetBeans. Se desea mantener el lenguaje de programación pero migrar el proyecto al entorno de desarrollo a Eclipse.

Como herramientas fundamentales para el soporte del rediseño se han utilizado los patrones de diseño y diseño por contrato [10], con el soporte de ambas se logró dar una mejor visibilidad y consistencia de las componentes principales.

En el proceso de desarrollo se abordó un mecanismo incremental, en el cual el primer ciclo se logró tener todas las componentes definidas pero sin operación, en los ciclos dos y tres se incorporó y depuro la funcionalidad de las diversas piezas de software.

1.4 PLAN DE TRABAJO

El plan de trabajo para la completa ejecución de la memoria queda resumido por el cronograma presentado en la Figura 3. Para una mejor comprensión se han incorporado los objetivos de cada una de las grandes tareas y una breve descripción de lo que se realizó en ellas.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**CC69F**	**115 días**	**lun 13-07-09**	**vie 18-12-09**
2	**Revisión del Diseño Inicial**	**15 días**	**lun 13-07-09**	**vie 31-07-09**
3	Diseño por Contrato de Familia de Software	5 días	lun 13-07-09	vie 17-07-09
4	Revisión de Patrones de Diseño a Implementar	5 días	lun 20-07-09	vie 24-07-09
5	Evaluación de Avance	5 días	lun 27-07-09	vie 31-07-09
6	**Revisión Implementación Existente**	**15 días**	**lun 03-08-09**	**vie 21-08-09**
7	Revisión de adecuación al diseño original	5 días	lun 03-08-09	vie 07-08-09
8	Mejoras posibles a la forma de implementación	5 días	lun 10-08-09	vie 14-08-09
9	Evaluación de Avance	5 días	lun 17-08-09	vie 21-08-09
10	**Extensión de Funcionalidades**	**15 días**	**lun 24-08-09**	**vie 11-09-09**
11	Busqueda de nuevos algoritmos e implementaciones	10 días	lun 24-08-09	vie 04-09-09
12	Evauación de Avance	5 días	lun 07-09-09	vie 11-09-09
13	**Rediseño de Software**	**15 días**	**lun 14-09-09**	**vie 02-10-09**
14	Diseño y especificación del Software	10 días	lun 14-09-09	vie 25-09-09
15	Evauación de Avance	5 días	lun 28-09-09	vie 02-10-09
16	**Implementación**	**40 días**	**lun 05-10-09**	**vie 27-11-09**
17	Ciclo A	15 días	lun 05-10-09	vie 23-10-09
18	Revisión A	5 días	lun 26-10-09	vie 30-10-09
19	Ciclo B	15 días	lun 02-11-09	vie 20-11-09
20	Revisión B	5 días	lun 23-11-09	vie 27-11-09
21	**Pruebas**	**15 días**	**lun 30-11-09**	**vie 18-12-09**
22	Pruebas funcionales	5 días	lun 30-11-09	vie 04-12-09
23	Verificación v/s Software Anterior	5 días	lun 07-12-09	vie 11-12-09
24	Verificación de metas propias	5 días	lun 14-12-09	vie 18-12-09

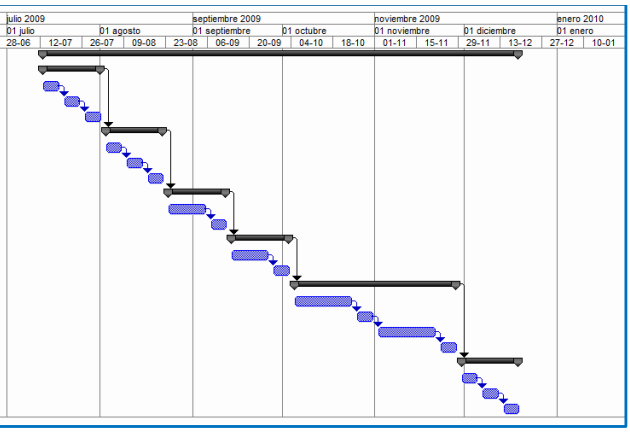

FIGURA 3 : PLAN DE TRABAJO

1.4.1 REVISIÓN DEL DISEÑO INICIAL

En este punto se realizó un estudio íntegro de la familia de software [6]. El resultado de este trabajo permitirá identificar los "**Cold Spot**", "**Hot Spot**" y patrones de diseño que usa y que complementen a la familia.

Cómo metodología de trabajo se utilizó diseño por contrato [10] con el cual se expresará el comportamiento que se desea obtener. Estos contratos serán los que guiarán los procesos de revisión de todo el trabajo.

1.4.2 Revisión Implementación Existente

Se elaboró un completo análisis al diseño e implementación de [8] aplicando los conceptos de la tarea anterior. Esto permitió identificar las zonas de la aplicación a mejorar.

1.4.3 Extensión de Funcionalidades

Se realizó una búsqueda de algoritmos de libre disposición para la generación de mallas iniciales, el proceso de refinamiento y la lectura de formatos de geometrías poligonales.

1.4.4 Rediseño de Software

En este punto se, ejecutó el destilado de las tres tareas anteriores y se consolidaron en un nuevo diseño. El cual incluyó las características antes abordadas y también las nuevas cualidades buscadas, como lo son obtener la inclusión de los Hot Spot.

1.4.5 Implementación

Esta actividad consiste en la construcción y verificación de las piezas de software destinadas a plasmar el diseño final. La tarea de implementación contó con dos ciclos de construcción y prueba.

1.4.6 Pruebas

En esta última actividad se evaluó el nuevo diseño y su implementación usando las métricas estándares definidas durante el análisis y diseño. Adicionalmente se compararon los resultados con los contratos obtenidos en el inicio.

2 TRABAJO RELACIONADO

A continuación se entrega una descripción de la familia de software [6] que inspira el producto a rediseñar. Ella permitió establecer el marco de referencia de los objetivos y mejoras a implementar.

El estudio de la familia de software descrita en [6], se realizó con un doble enfoque. Primero se efectuó un análisis orientado a detectar el listado de las componentes más relevante, el cual incluyo una descripción de sus respectivos comportamientos y estados. Posteriormente se buscaron los patrones de diseño utilizados en la familia. Como resultado se logró una revisión completa de la estructura estática y el comportamiento dinámico de las principales componentes de la familia.

2.1 COMPONENTES

A continuación se presenta una descripción de las componentes que integran la familia de software, esta ha sido descompuesta en las piezas de software responsables de la representación de la malla geométrica y de aquellas orientadas a la generación de la misma.

2.1.1 REPRESENTACIÓN DE LA MALLA

La malla geométrica es el centro del estudio, pues es ella la que se desea generar y/o depurar. La malla se encuentra representada por conjunto de triángulos en un espacio de dos dimensiones. Asimismo los triángulos están representados por sus vértices y lados. En la Figura 4 se puede apreciar la relación entre las componentes principales que permiten representar la malla.

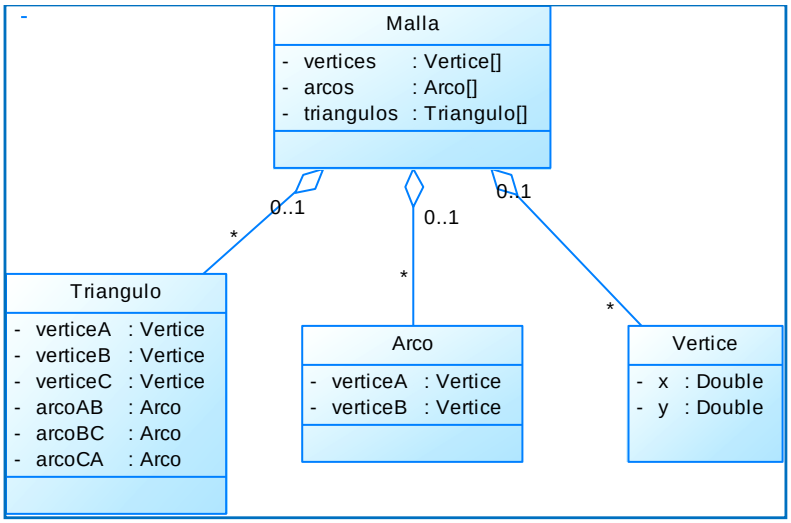

FIGURA 4 : COMPONENTES ASOCAIDAS A MALLA

La componente Malla representa a las mallas geométricas, ella se encuentra definida por tres conjuntos, uno de triángulos, otro de arcos y finalmente uno de vértices. Es sobre esta componente donde se realizaran los procesos.

La representación de un los triángulos está dado por la componente homónima. Una forma de definir un triángulo es por sus tres vértices (A, B, C); de ellos se deducen los tres segmentos y ángulos. En este caso los segmentos se definen, no se deducen, para mejor soporte del procesamiento de la mallas.

Los arcos quedan definidos por dos puntos en el espacio. Para dichos puntos se ocupa la componente Vértice, la cual almacena las coordenadas x e y del plano.

Las componentes antes descritas no poseen comportamiento relevante para la familia de software.

2.1.2 ALGORITMOS, CRITERIOS Y REGIÓN

Los algoritmos representan los procesos utilizados durante la generación de la malla. En cambio los criterios son las componentes que permiten decidir si se continúa o no con la operación de refinamiento. Finalmente la región representa el lugar geométrico donde se refinará.

Es en esta parte de la familia donde se pueden realizar las extensiones para agregar nuevos comportamientos. Esto se logra por la implementación de nuevos algoritmos, criterios y regiones. Por lo tanto se convierte en un candidato a “Hot Spot”.

Si bien, el número de componentes totales es mayor al aquí expuesto, es posible entender la funcionalidad completa únicamente describiendo las clases abstractas de la familia. No obstante lo anterior en el diagrama de la Figura 6 se incluye todas las componentes asociadas a algoritmos y criterios, por otro lado la Figura 5 muestra el diagrama de las clases que representan regiones.

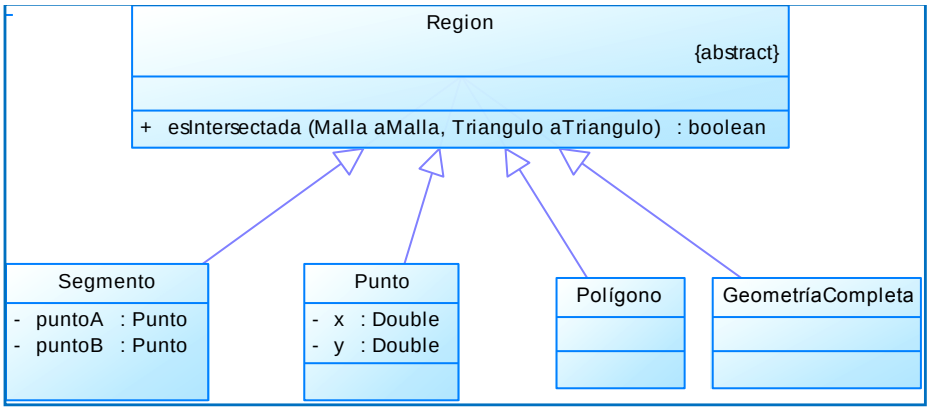

FIGURA 5 : COMPONENTES ASOCIADAS A REGIÓN

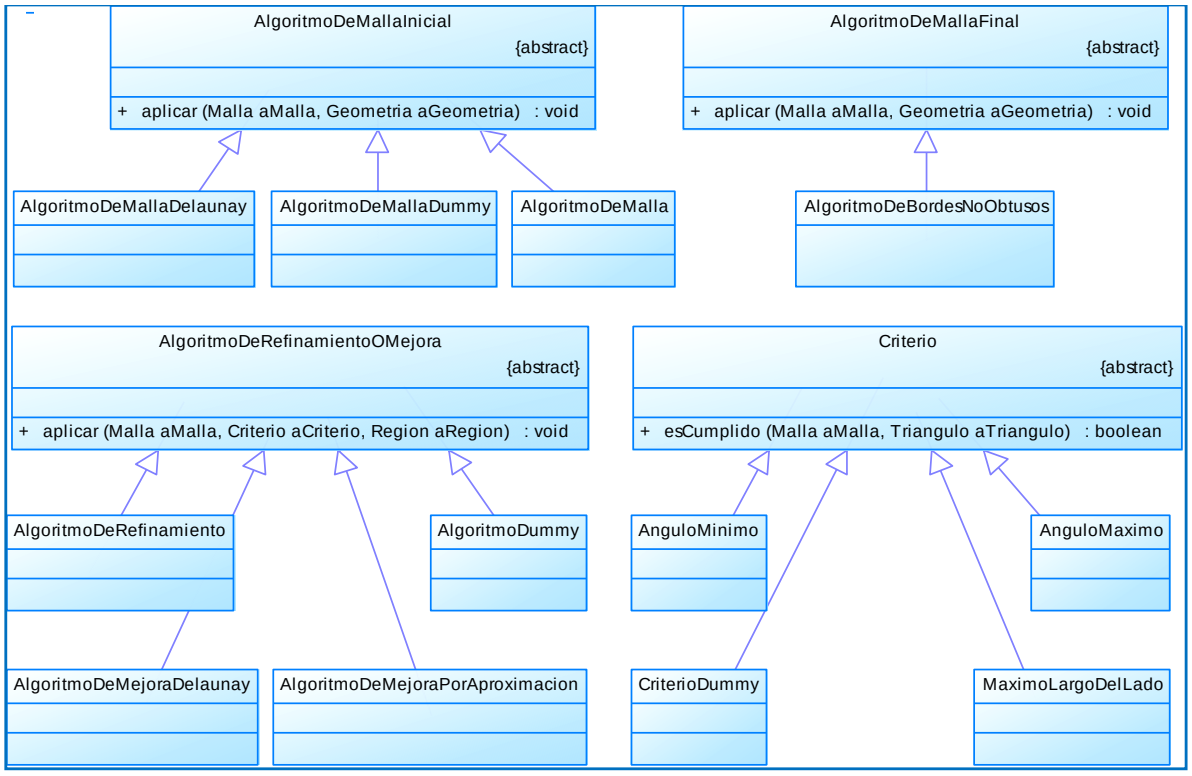

FIGURA 6 : COMPONENTES PARA LA GENERACIÓN DE MALLA

Por un lado se tiene la componente abstracta Región, la cual es responsable de representar cualquier forma o geometría bidimensional. Las componentes Segmento, Punto, Polígono y Geometría Completa son especializaciones de Geometría y permiten definir geometrías más específicas.

La componente Algoritmo De Malla Inicial permite generar una malla inicial. Es el primer paso en el proceso de generación de la malla geométrica. A diferencia de Algoritmo De Malla Final, la cual es el último paso.

El Algoritmo De Refinamiento o Mejora asume el proceso de la depuración de la malla que se está procesando, para ello se apoya en la componente Criterio y su capacidad de decidir si un triángulo en particular cumple con ciertas condiciones con respecto a la malla.

En la Tabla 2 se expresan los métodos principales de las componentes antes descritas

Componente	Método	Descripción
Región	boolean esIntersectada(Malla, Triángulo):	Revisa si la "Región" es intersecada por el "Triángulo", de ser así devuelve el valor booleano "verdadero" en caso contrario "falso"
Algoritmo de Malla Inicial	void aplicar(Malla, Geometría)	Genera una malla inicial
Algoritmo de Malla Final	void aplicar(Malla, Geometría)	Genera una malla final.
Algoritmo De Refinamiento o Mejora	void aplicar(Malla, Geometría, Criterio)	Depura la malla revisando la geometría.
Criterio	boolean esCumplido(Malla, Triángulo)	Revisa si el Triángulo cumple el criterio para la Malla solicitada.

TABLA 2 : MÉTODOS PRINCIPALES DE LA FAMILIA DE SOFTWARE

2.2 PATRONES DE DISEÑO

En la familia de software se plantea el uso de varios patrones de diseño de manera explícita. Aquí se encuentra una descripción de los patrones y su aplicación.

2.2.1 STRATEGY

El patrón “Strategy” permite definir una familia de algoritmos, encapsulado cada uno, y haciéndolos intercambiables.

Los criterios y algoritmos son encapsulados bajo el patrón “Strategy”, de esta forma se permite que cada uno de los algoritmos o del criterio posean implementaciones diferentes e intercambiables.

2.2.2 ITERATOR

El patrón “Iterator” permite acceso a los elementos de un objeto agregado de forma secuencial sin exponer su representación subyacente.

Parte del proceso de depuración consiste en verificar que cada triángulo de la malla actual que intersecta la región especificada cumpla con el criterio que se está evaluando. En caso de no cumplirlo se le aplicara el Algoritmo de Depuración o Refinamiento, para lo cual se necesita hacer un recorrido secuencial sobre la totalidad de Triángulos de la Malla.

3 TRABAJO REALIZADO

En este capítulo se expone el trabajo que fue realizado en la presente memoria. Las tareas son descritas según el orden en que fueron enfrentadas: análisis preliminar, estudio de implementación existente, rediseño, implementación y pruebas.

3.1 ANÁLISIS PRELIMINAR

Antes de comenzar el trabajo de rediseño se realizó un análisis inicial orientado a establecer métricas de la calidad para la validación del diseño. En dicha labor se consideró que el presente trabajo es un producto intermedio, por lo cual debería apuntar a la extensibilidad. Además se considera la depuración del producto y los objetivos propios de la memoria. Con lo anterior se establecieron las siguientes métricas para evaluar el resultado final del producto.

1. La geometría a ser utilizada en el proceso de refinamiento deben poder ser obtenidas desde un archivo especificado como una imagen o un archivo de texto que describe un dominio poligonal.
2. El proceso de refinamiento se debe entender como una secuencia de pasos independientes, los cuales son ejecutables por separado. De esta forma la malla geométrica generada en un paso puede ser un resultado en sí.
3. La incorporación de nuevos criterios o algoritmos debe ser realizada con el mínimo esfuerzo. Esto debiera ser posible sin necesidad de alterar ninguna componente ya existente, ni tampoco conocer la implementación de las demás piezas de software.
4. La implementación actual no maneja regiones específicas, por tanto se pretende agregar al código actual de la manera descrita en la familia.

Estos cuatro puntos son las metas técnicas que el producto debe cumplir. En el proceso de pruebas se realizaron las revisiones de cada una de ellas permitiendo confirmar la completa cobertura de las mismas.

3.2 REVISIÓN DEL DISEÑO INICIAL

La familia de software presenta un diseño inicial el cual sirve como guía para el rediseño del producto. En esta parte del análisis se estableció cuáles serían las principales funcionalidades a ser implementadas y qué se esperaba de cada una de ellas.

El resultado de esta revisión fue la descomposición del proceso de generación de una malla en cuatro sub procesos. A cada uno se le identificó y se estableció una meta para ser cumplida dentro alcance de la presente memoria.

3.2.1 OBTENCIÓN DE GEOMETRÍA INICIAL

Al iniciar el proceso se debe contar con una geometría, la cual es obtenida desde un archivo.

Se establece como meta ser capaz de mantener la funcionalidad de leer archivos de imágenes y extender el proceso a archivos de texto que describen geometrías poligonales.

3.2.2 OBTENCIÓN DE MALLA INICIAL

A partir de una geometría se debe poder generar automáticamente una malla inicial.

Se establece como meta poder generar mallas para geometrías provenientes de polígonos convexos e imágenes.

3.2.3 Refinar una malla geométrica

Utilizando un algoritmo de refinamiento y un criterio se debe realizar un proceso de mejora. Este proceso alterará la malla inicial y provocará que todos los triángulos que pertenezcan a la malla cumplan con el criterio. Para ello el algoritmo necesita poder acceder a la malla, un criterio y una región. Dependiendo del criterio usado, se usara la imagen de input o no.

Se establece como meta que todos los algoritmos y criterios sean aplicables a cualquier geometría, no importando su tipo.

3.2.4 Obtención de una Malla Final

Como último paso del proceso de generación automática de una malla se aplica un algoritmo para la obtención de una malla final, ya sea aplicando un post proceso de mejoramiento o para generar información adicional.

Se establece como meta tener al menos un algoritmo de malla final.

3.3 Estudio de la implementación existente

Parte focal del desarrollo de la presente memoria consistía en modificar el diseño de la implementación efectuada en [8]. Como etapa central de dicho proceso se realizó una comparación entre el software obtenido en [8] y el diseño planteado en la familia de software descrita en [6]. Dicha comparación se divide en la revisión de la implementación de las componentes básicas de la familia y la utilización de los patrones de diseño respectivos.

3.3.1 Revisión de Componentes

En el presente análisis se evaluó la similitud de la implementación de las componentes básicas de la familia. Para ello se verificó que se encuentren presentes los métodos mínimos requeridos y que las firmas coincidan.

3.3.1.1 COMPONENTES QUE SE ENCENTRAN PRESENTADAS EN LA IMPLEMENTACIÓN

En la Tabla 3 se indica cuales componentes de la familia inicial cumplen con el diseño original y que pieza de software la representa.

Componente	Clase java que la representante
Triángulo	treeangles.mesh.Triangle
Arco	treeangles.mesh.Edge
Vértice	treeangles.mesh. Point3D
Malla	treeangles.mesh.Mesh

TABLA 3 : COMPONENTES V/S CLASES JAVA

3.3.1.2 COMPONENTES QUE NO SE ENCUENTRAN BIEN REPRESENTADAS

En el proceso de análisis del producto se encontró una serie de componentes que no respetaban el espíritu del diseño de la familia de software. En algunos casos se incluyeron mejoras al diseño, en otras ocasiones no se encontraban presentes como parte de la implementación y como tercer escenario algunas de ellas adolecían de vicios de implementación.

3.3.1.2.1 REGIÓN

Componente representada por la interfaz “treeangles.region.Region”.

El método “esIntersectada(Malla, Triángulo)” se encuentra definido por “intersects(Triangle t)”. Donde la firma ha sido cambiada y la “Malla” no es entregada como parámetro. Lo anterior implica que “Región” conoce la “Malla” a la cual se está haciendo referencia, conllevando un vicio en el diseño.

3.3.1.2.2 SEGMENTO

Componente no implementada.

3.3.1.2.3 PUNTO

Componente no implementada.

3.3.1.2.4 GEOMETRÍA COMPLETA

Componente representada por medio de “treeangles.region. Rectangle”.

El método “esIntersectada(Malla, Triángulo)” se encuentra definido por “intersects(Triangle t)”. Incurre en el mismo vicio que “Región”.

3.3.1.2.5 ALGORITMO DE MALLA INICIAL

Componente representada por la interfaz “treeangles.initialMesh.InitialMesh”.

Si bien el método “aplicar” se encuentra presente, su firma ha sido cambiada. En vez de utilizar una Malla y una Geometría, solo tiene la primera. Para tener acceso a la Geometría esta se encuentra implementada como una variable de instancia. Este es un error en el diseño de la actual implementación, pues Geometría no corresponde a una variable de estado del algoritmo.

3.3.1.2.6 ALGORITMO DE MALLA FINAL

Esta componente no se encuentra explicita en la implementación.

3.3.1.2.7 ALGORITMO DE REFINAMIENTO O MEJORA

Componente representada por la interfaz “treeangles.refinement. Refinement”.

El refinamiento es efectuado utilizando una malla geométrica, un criterio de refinamiento y un criterio de sección de punto. Esta última componente incorpora una nueva dimensión a la solución, la cual no está cubierta por la familia.

Adicionalmente el acceso a la geometría es por una variable de instancia. Lo cual es un error consistente con “Algoritmo de Malla Inicial”.

3.3.1.2.8 CRITERIO

Componente representada por la interfaz “treeangles.refinementCriterion. RefinementCriterion”.

El método “esCumplido” esta implementado con exactitud por “meetsCriterion”.

Adicionalmente existe la interfaz “treeangles.refinementCriterion.PictureRefinementCriterion” la cual es una especialización de “treeangles.refinementCriterion. RefinementCriterion”. Esta interfaz no significa ningún aporte o especialización de Criterio. Su utilidad es la de de permitir discriminar algoritmos para tipos específicos de representación de

malla. Esto no debe ser así, cualquier criterio debe ser aplicable no importando la implementación particular de la malla a trabajar.

Es importante mencionar que existe la interfaz “treeangles.pointSelectionCriterion. PointSelectionCriterion” la cual representa otra forma de “criterio”, el de selección de punto. Dicho criterio no está incluido en la familia de software original. Su representación sigue las mismas normas de la componente “Criterio”.

Así como para el criterio de refinamiento existe un vicio de diseño en la especialización, también se tiene el mismo vicio con la componente “treeangles.pointSelectionCriterion.PicturePointSelectionCriterion”. Esto debe ser independiente si lo aplicamos sobre geometrías específicas como PSLG o como imágenes (Picture).

3.3.2 Revisión de los Patrones de Diseño

En el caso de “Strategy” es utilizado no sólo para los algoritmos existentes, sino también para la representación de los criterios de selección de punto. Con lo anterior se cumple a cabalidad el uso del patrón.

En el caso de “Iterator” esta aplicado al recorrido que realizan los diferentes algoritmos sobre la malla.

3.4 Rediseño de software

Como estrategia de rediseño se optó por volver a implementar todas las componentes. Esta decisión fue tomada una vez que el diagnóstico de la implementación existente reportara los vicios expuestos en el punto anterior. Al volver a codificar las componentes desde cero se logró forzar el correcto uso de los patrones de diseño, liberando el producto de las limitantes y generando un software que cumple con las metas antes establecidas.

El rediseño fue trabajado en cuatro etapas; en la primera de ellas se extendió el diseño de la familia de software para incorporar aspectos detectados en la implementación existente. Posteriormente se definieron las componentes principales y sus responsabilidades. Como tercera etapa se establecieron los contratos que guardarían el correcto cumplimiento del comportamiento esperado. Finalmente se incorporó el código asociado a los criterios de selección de punto y de refinamiento, debiendo codificarse íntegramente las piezas asociadas al algoritmo de mejora de Delanuay, malla inicial y malla final; además de crear las componentes asocias a las geometrías PSLG.

3.4.1 EXTENSIÓN DEL DISEÑO

Un elemento no cubierto por la familia de software en el proceso de refinamiento es la selección del punto para la separación de un triángulo en tres nuevos triángulos. En el producto realizado por Pablo Aguilar se identificó esta necesidad, La cual fue plasmada en el uso de criterios tales como el ortocentro del triángulo y el punto que presentara la mayor diferencia respecto al tono promedio del triángulo en cuestión.

Para incluir dicha brecha se estableció una nueva componente en la familia de software. Esta pieza es un nuevo tipo de criterio responsable de seleccionar el punto en el proceso de refinamiento.

3.4.2 PRINCIPALES COMPONENTES

En el nuevo diseño se estableció una división orientada a separar los elementos según su rol en el proceso de refinamiento. En un primer grupo se establecieron aquellos que son responsables de la representación de la malla geométrica. Como segundo grupo se tienen las componentes responsables de almacenar los datos de las geometrías, incluyendo sus especializaciones. En una tercera agrupación quedaron las clases responsables de la representación de las regiones. Finalmente en un cuarto gran paquete quedaron las responsables del

comportamiento orientado al refinamiento, los cuales se pueden apreciar en Figura 7. Dicha distribución se revisa con más detalle a continuación.

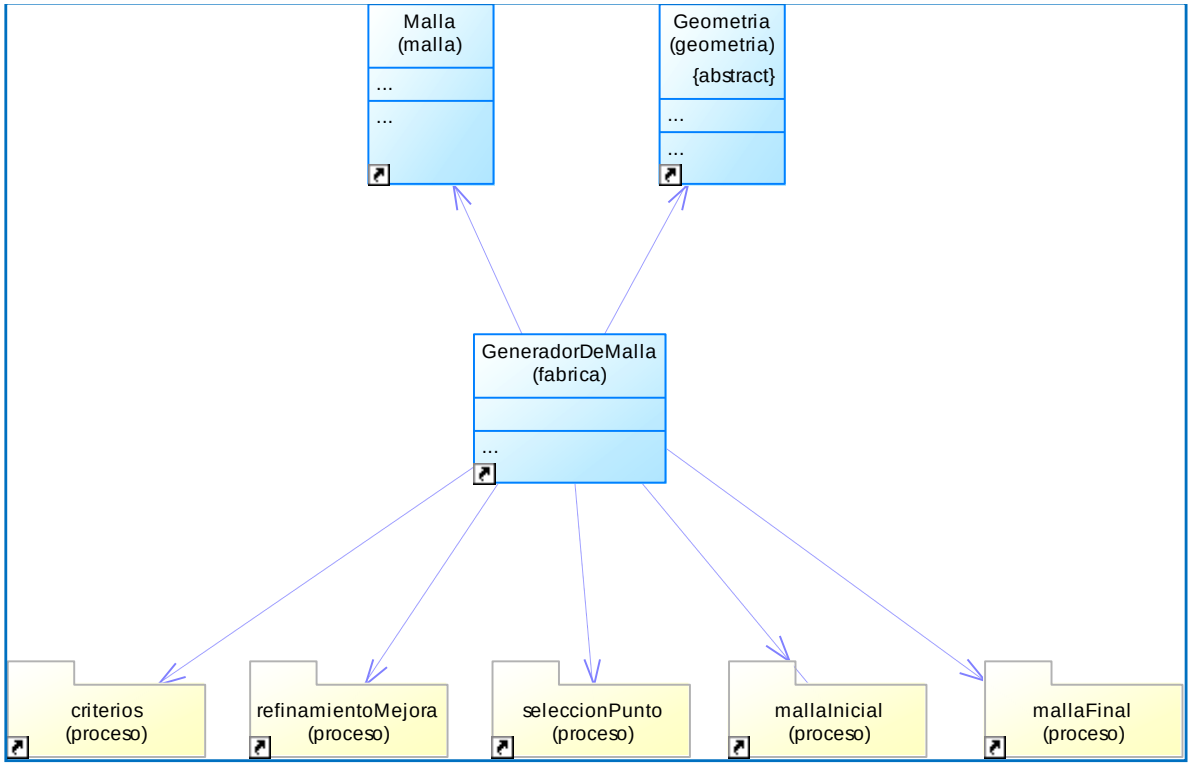

FIGURA 7 : COMPONENTES PRINCIPALES

3.4.2.1 MALLA

Las componentes de este grupo están orientadas a representar la malla geométrica que se va a estudiar.

Una malla geométrica está definida por un conjunto de triángulos. Un triángulo se puede representar por sus tres vértices, así como también por los tres arcos que lo componen. En el caso de este análisis ambas representaciones resultan útiles por lo cual se utilizan de manera sincronizada.

Por lo tanto, podemos ver una malla como el conjunto de sus triángulos, arcos y vértices. Los últimos dos conjuntos son siempre deducibles del primero Tal cual como se aprecia en la Figura 8.

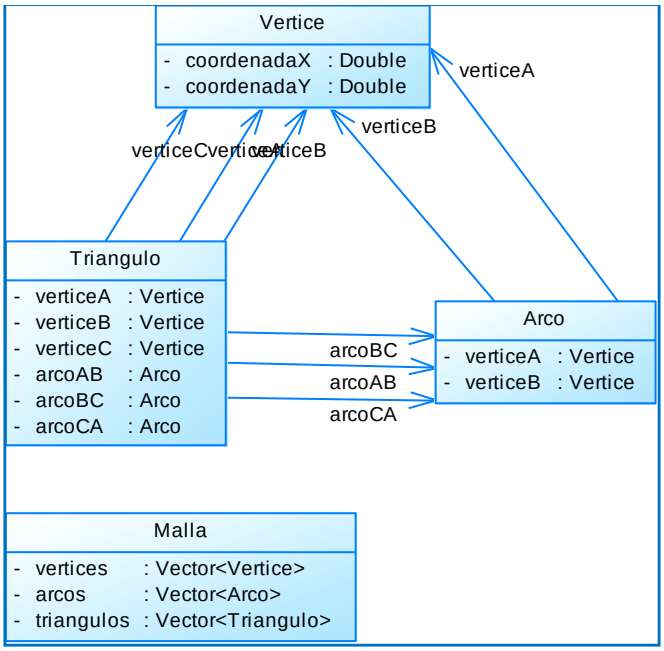

FIGURA 8 : PAQUETE MALLA

3.4.2.2 GEOMETRÍA

La geometría es el punto de partida del proceso de trabajo, dado que desde ella se genera una Malla inicial. La obtención de una geometría dependerá de qué tipo de origen se esté utilizando, por lo tanto resulta natural encapsular el instanciado del estado del objeto en su constructor.

Para lograr obtener geometrías de diversas fuentes se realizarán especializaciones de la clase “Geometría”, dejando así la libertad de incorporar nuevas fuentes de información sin alterar el proceso de generación de una malla.

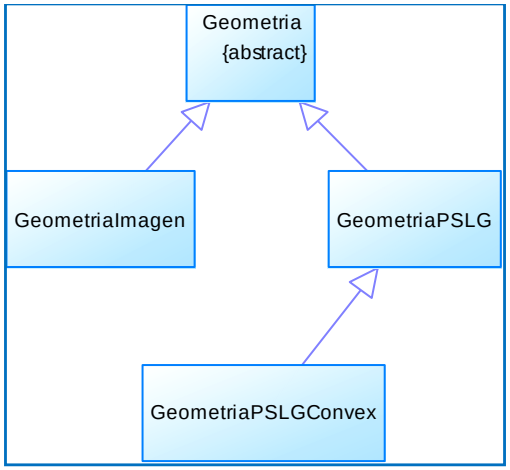

FIGURA 9 : PAQUETE GEOMETRÍA

3.4.2.3 REGIÓN

La región es el lugar geométrico sobre el cual se realiza el proceso de refinamiento. Si bien fueron definidas las regiones “Polígono” y “Punto”, no fueron parte del alcance del presente trabajo y todos los procesos se han realizado sobre la “Geometría Completa”.

Lo anterior significa que al aplicar un algoritmo de mejora en el proceso siempre se realizará sobre toda la geometría que originó la malla inicial, dejando la posibilidad que trabajos futuros puedan extender el software para realizar procesos reducidos a una región particular, como lo es un cuadrado o un circulo.

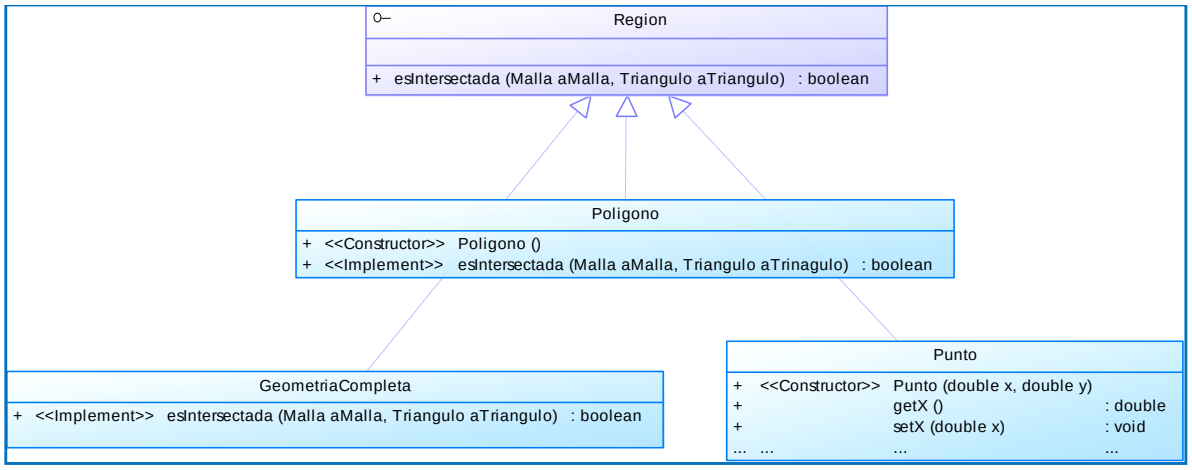

FIGURA 10 : PAQUETE REGIÓN

3.4.2.4 PROCESO

Las componentes definidas en “proceso” representan las principales interfaces a ser implementadas para el refinamiento de mallas geométricas.

La componente “Generador de Malla” encapsula los procesos asociados a cada paso del algoritmo de refinamiento.

“Malla Inicial” representa a todos los algoritmos que permiten generar una nueva malla geométrica desde una geometría particular. Por su parte “Malla Final” encapsula los algoritmos que permitan realizar un proceso de generación de información final, estadísticas o mejoras específicas de una malla

“Refinamiento Mejora” encapsula algoritmos que permiten realizar un paso de refinamiento de una malla. Para lo cual revisa todos los triángulos de malla en cuestión y revisa que se encuentren dentro de la región a mejorar. De ser así prueba que el triángulo cumpla con las condiciones de “Criterio”. Para todos los triángulos que no cumplan se selecciona un punto de su interior utilizando

“Selección Punto” y se divide el triángulo. Esto es revisado por las reglas propias de cada refinamiento.

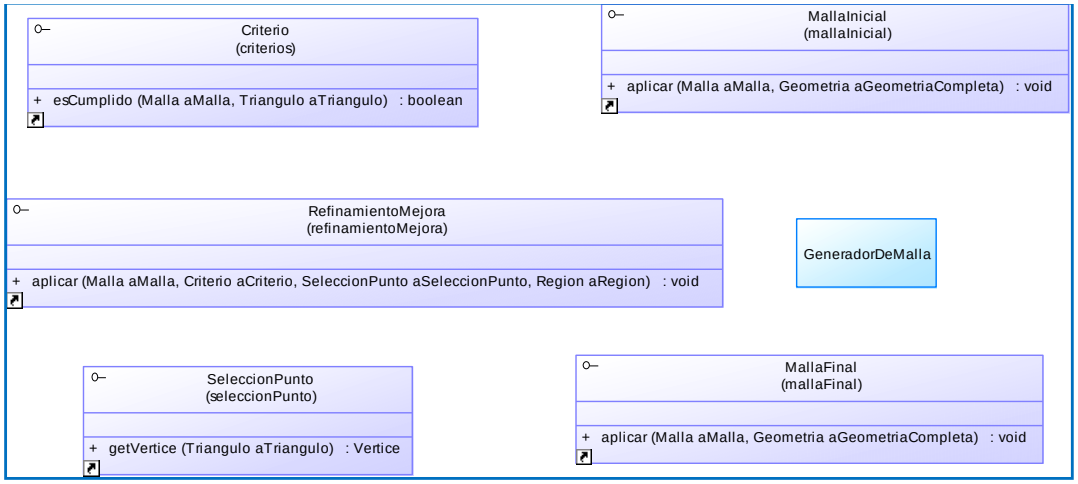

FIGURA 11 : PAQUETE PROCESO

3.4.3 DISEÑO POR CONTRATO

Como parte de las buenas prácticas de diseño se ha establecido el uso de la metodología de “diseño por contrato” [10]. Dicha herramienta permitió verificar el correcto funcionamiento de las componentes claves del nuevo diseño.

Para lograr lo anterior se identificó una serie de invariantes los cuales fueron implementados durante la etapa de construcción. Adicionalmente se procedió a identificar los contratos destinados a resguardar el comportamiento de los métodos principales del proceso de generación de las mallas geométricas.

3.4.3.1 INVARIANTES

En Tabla 4 se expresa el listado de los invariantes de las componentes principales.

Componente	Invariante
Malla	this.vertices.size() – this.arcos().size() + this.numeroDeCaras() = 1
Triángulo	this.verticeA <> this.verticeB this.verticeB <> this.verticeC this.verticeC <> this.verticeA this.verticeA <> null this.verticeB <> null this.verticeC <> null this.verticeA no pertenece this.arcoBC this.verticeB no pertenece this.arcoAB this.verticeC no pertenece this.arcoAB this.area()>0
Arco	this.verticeA <> null this.verticeB <> null this.verticeA <> this.verticeB this.largo()>0
Vértice	this.x <> nulo this.y <> nulo

TABLA 4 : INVARIANTES

3.4.3.2 CONTRATOS

La Tabla 5 representa los contratos identificados

Componente	Método	Pre Condiciones	Post Condiciones
AlgoritmoDeMallaInicial	aplicar	aMalla <> nulo aGemoetria <> nulo	aMalla <> nulo aMalla contiene toda aGeometria
AlgoritmoDeMallaFinal	aplicar	aMalla <> nulo aGemetria <> nulo	aMalla <> nulo aMalla contiene toda aGeometria
AlgoritmoDeRefinamientoOMejora	aplicar	aMalla <> nulo aCriterio <> nulo aSeleccionPu	Para todo Triangulo en la aMalla haber verificado aCriterio.esCumplido(aMalla,Triangulo)

		nto<> null aRegion <> nulo	
Criterio	esCumplido	aMalla <> nulo aTriangulo <> nulo	aTriangulo revisado dentro de aMalla

TABLA 5 : CONTRATOS PRINCIPALES

3.5 IMPLEMENTACIÓN

Una vez establecido el diseño a ser implementado se crearon todas las componentes y se efectuó el proceso de adaptar los algoritmos embebidos en el código existente a las nuevas estructuras. Durante este proceso se aprovechó de implementar documentación del código fuente bajo estándar javadoc a todas las componentes del software.

Posteriormente se procedió a buscar bibliotecas para la construcción de los contratos. Después de un proceso de revisión de varias alternativas se acotaron las opciones a dos posibilidades: JContrator [3]y C4J[4], ambas soluciones de código abierto y hospedadas en Sourceforge. JContrator posee una comunidad activa la cual la está soportando a diferencia de C4J que se encuentra al final de su ciclo de vida. Como ventajas C4J permite establecer los contratos por medio de clases externas a las componentes, independizando el contrato de quien es vigilado. Otro punto que hizo pesar la balanza de C4J por sobre JContrator es su integración directa con el IDE Eclipse. Adicionalmente C4J permite activar y desactivar la validación sin necesidad de alterar el código, a diferencia de JContrator que requiere proceso de pre compilación. Con lo cual la opción que fue seleccionada para la implementación de los contratos es C4J. Una vez tomada esta decisión se procedió a implementar las componentes destinadas a albergar los contratos e invariantes descritos en el punto “Diseño por contrato”.

3.5.1 EXTENSIÓN DE FUNCIONALIDADES

Como gran extensión a la funcionalidad original se implementó el manejo de geometrías poligonales como geometría. Para ello se extendió la funcionalidad de cargar geometrías iniciales desde archivos con extensión “poly”. Además se incluyeron las componentes orientadas a realizar la especialización de la clase geometría en sus variaciones PSLG y “Convex PSLG”. Por otro lado, se incorporó

[3] Ver sitio web: http://jcontractor.sourceforge.net

[4] Ver sitio web: http://c4j.sourceforge.net/

la posibilidad de crear mallas iniciales en función de los nuevos tipos de geometría.

La generación de geometrías desde geometrías poligonales fue realizada en base a implementaciones de algoritmos de código abierto, estos fueron encapsulados utilizando el patrón de diseño “Bridge”. Para ello se creó una interfaz “GeometriaPSLGPuente” responsable de la creación de la malla, la cual es invocada por “GeneradorDeMalla” cuando se intenta crear una geometría desde un archivo tipo “poly” como se puede ver en la Figura 12. Adicionalmente se habilitó la componente “GeometriaPSLGConvexDelanuayPuente” la cual utiliza las clases de código abierto para la generación de la malla inicial. De encontrarse más implementaciones que permitan crear mallas estas pueden incorporarse como “puentes” nuevos y quedar disponibles para su uso. La selección del puente en particular a ser utilizado que determinado por el estado del objeto de tipo ParametroDTO.

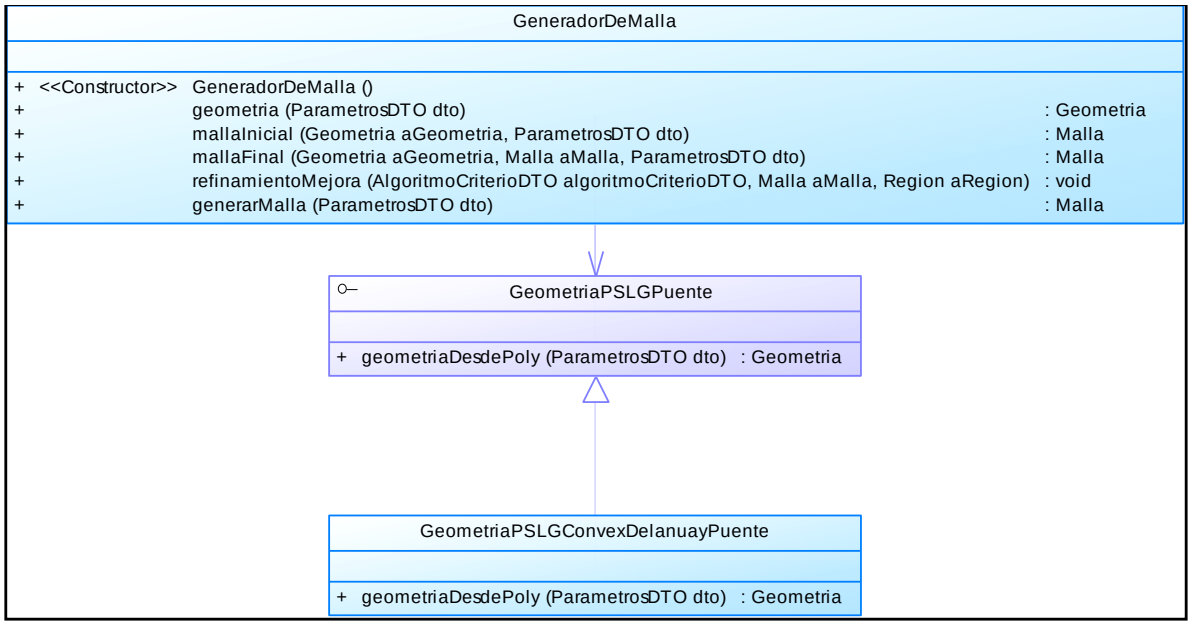

FIGURA 12 : PATRÓN BRIDGE

3.6 PRUEBAS

Las pruebas fueron separadas según sus características de funcionales y no funcionales.

3.6.1 PRUEBAS FUNCIONALES

Las pruebas funcionales están divididas en tres grandes grupos. Las primeras están orientadas a la obtención de la geometría. Estas se han dividido en las de las geometrías procedentes de imágenes y las de polilíneas.

Como segundo bloque se han realizado revisiones a la generación de la malla inicial, las cuales se han concentrado en la malla proveniente desde polilíneas.

El tercer grupo se refiere al refinamiento de la malla. Aquí se han realizado pruebas unitarias de cada componente y pruebas cruzadas mezclando los diversos algoritmos y criterios, tanto de selección de punto como de refinamiento.

3.6.1.1 OBTENCIÓN DE LA GEOMETRÍA

Como parte del proceso inicial se debía ser capaz de obtener una geometría para ser trabajado. En el caso de las imágenes esto se obtiene de manera directa desde las propiedades de la misma, pero para las polilíneas era requerido poder leer el contenido del archivo en formato “poly” para su representación para los procesos de refinamiento.

Aquí se presentan imágenes que son el resultado de pruebas asociadas a las geometrías del icoságono y la geometría con forma de “coma”, la primera consta de un total de 20 puntos y la segunda de 48.

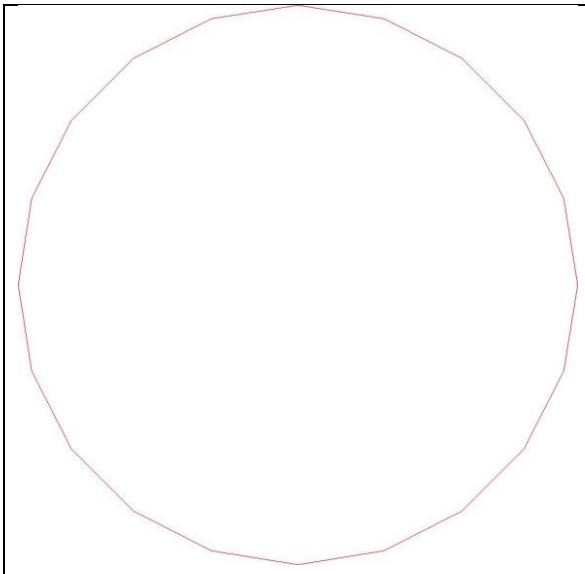

FIGURA 13 : GEOMETRÍA DEL ICOSÁGONO

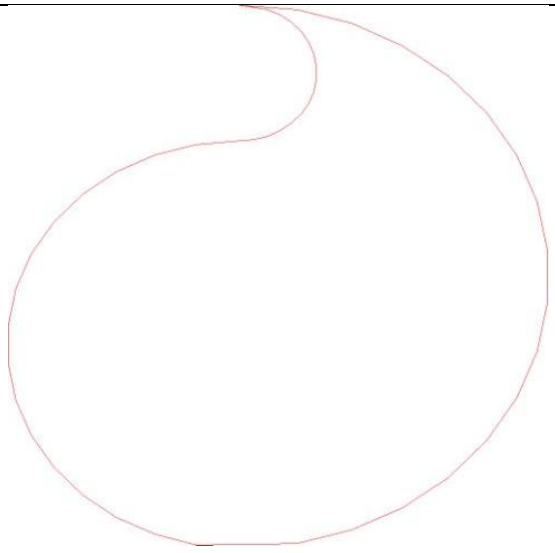

FIGURA 14 : GEOMETRÍA DE LA COMA

3.6.1.2 GENERACIÓN DE MALLA INICIAL

En el caso de las geometrías de origen desde imagen simplemente se mantuvo la malla inicial la cual consiste en una triangulación de Delanuay simple que es el rectángulo de la imagen dividido por la diagonal del mismo.

Por el contrario, la generación de mallas iniciales para las polilíneas es un punto que debió ser probado. Para esto se utilizó un algoritmo de código abierto, por lo cual se probó su adaptación más que la forma de la funcionalidad en sí.

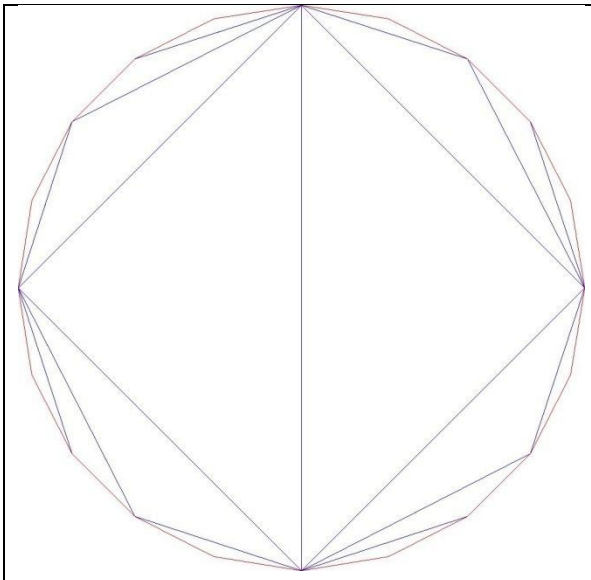

FIGURA 15 : MALLA INICIAL DEL ICOSÁGONO

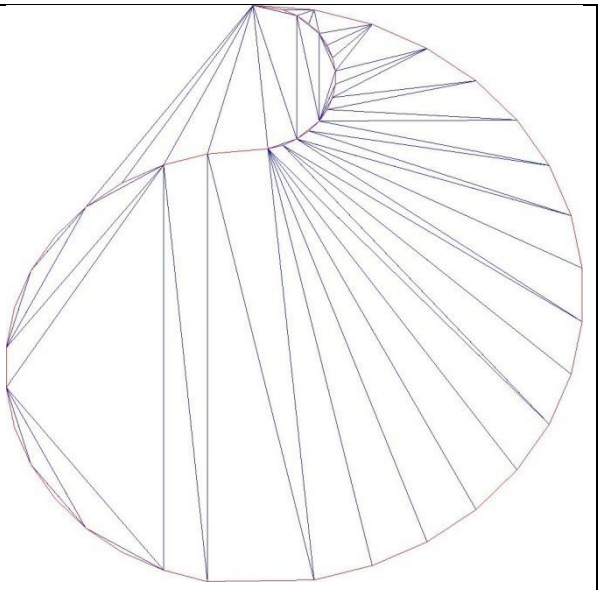

FIGURA 16 : MALLA INICIAL DE LA COMA

3.6.1.3 REFINAMIENTO DE MALLA

El último ítem a probar consistió en la depuración de mallas geométricas bajo diversos criterios.

3.6.1.3.1 PRUEBA A

Para la geometría icoságono se utilizó un doble proceso de refinamiento.

Primero se utilizó una triangulación de Delanuay con criterio de largo máximo del lado con un valor de 100, luego se procedió con un criterio de área máxima utilizando un valor 500. En ambos casos se utilizó el criterio de selección del ortocentro del triángulo. A nivel numérico los resultados se exponen en la Tabla 6

Triángulos Totales	106
Arcos Totales	318
Vértices Totales	76
Segundos de Proceso	37,4

TABLA 6 : RESULTADOS PRUEBA A

Aquí se presentan gráficamente los resultados de cada una de las etapas. En la Figura 17 se ve como el proceso queda después de terminar la primera iteración, en cambio en la Figura 18 se ve el proceso de refinamiento completo.

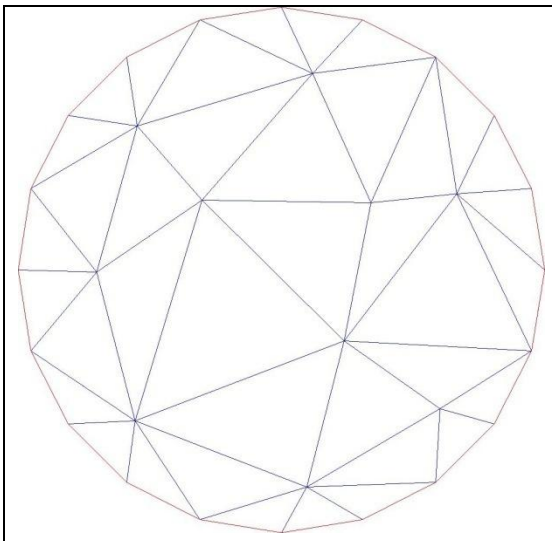

FIGURA 17 : PRIMERA ITERACIÓN

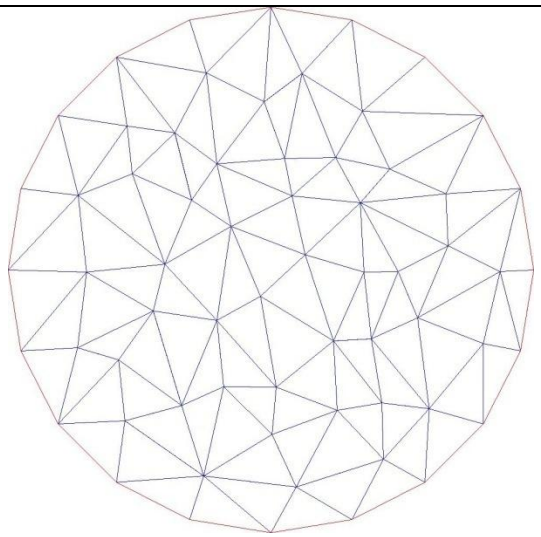

FIGURA 18 : SEGUNDA ITERACIÓN

3.6.1.3.2 PRUEBA B

Para la geometría de la “comma” se aplicó un refinamiento de Delanuay con un criterio de selección del ortocentro del triángulo y el área máxima con un valor asociado de 0,5. En Tabla 7 se muestran los resultados obtenidos.

Triángulos Totales	174
Arcos Totales	522
Vértices Totales	186
Segundos de Proceso	74

TABLA 7 : RESULTADOS PRUEBA B

En la Figura 19 se aprecia el proceso de refinamiento concluido.

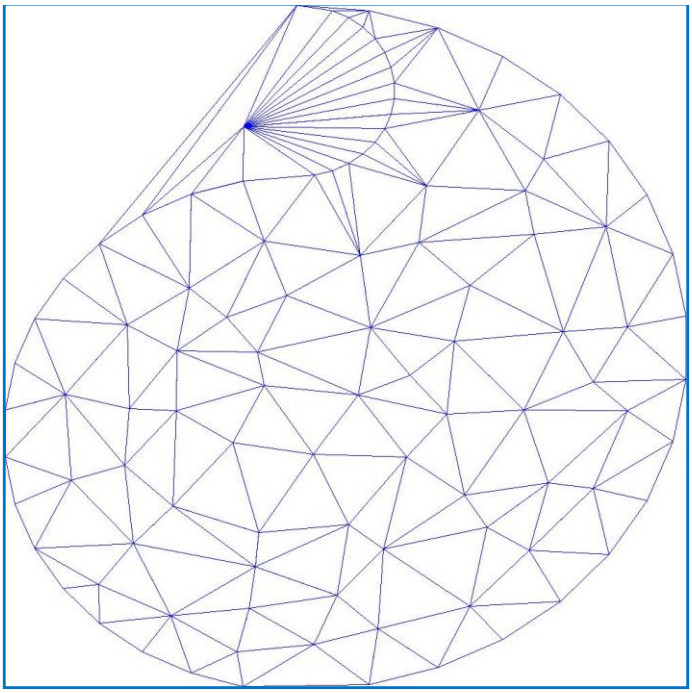

FIGURA 19 : MALLA GENERADA

3.6.1.3.3 PRUEBA C

En la prueba con la imagen “C” se procedió a utilizar el nuevo filtro de escala de grises para la creación de la geometría. En el refinamiento se aplicó el criterio mixto de área máxima e intensidad utilizando valores de 300 y 150 respectivamente, para la selección del punto se utilizó el criterio de mayor error. Al final del proceso se obtuvieron los resultados descritos en Tabla 8.

Triángulos Totales	1464
Arcos Totales	4392
Vértices Totales	735
Segundos de Proceso	**16556**
Horas de Proceso	4.6

TABLA 8 : RESULTADOS PRUEBA C

En la Figura 20 se aprecia la imagen incial, por el contrario en la Figura 21 se observa como queda la malla geométrica superpuesta a la imagen.

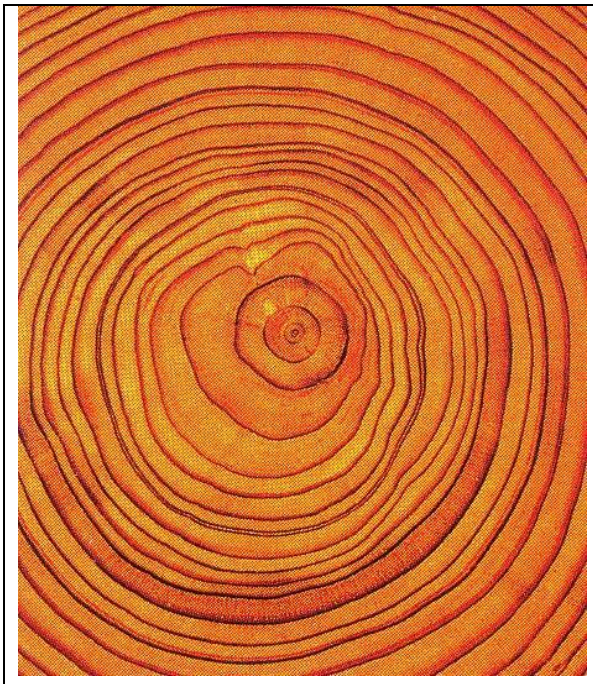

FIGURA 20 : IMÁGEN “C”

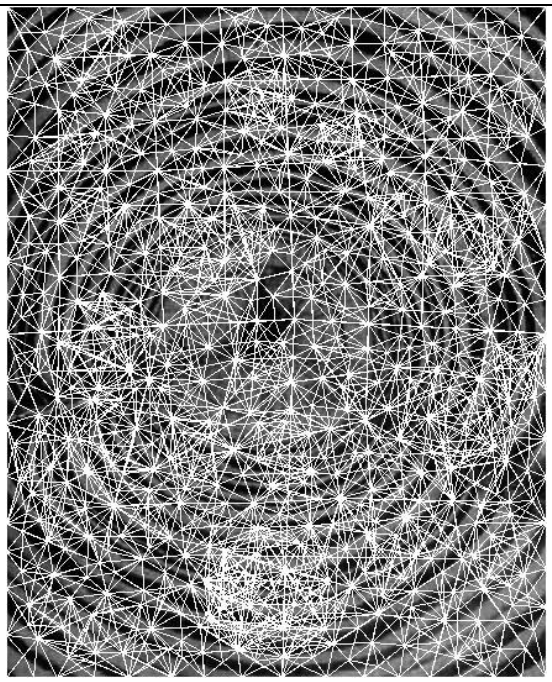

FIGURA 21 : IMÁGEN “C” CON SU MALLA

3.6.1.3.4 PRUEBA D

En la prueba con la imagen “D” se realizó una mezcla de filtros, primero aplicándose uno Gaussiano y posteriormente uno de Ecualización. En el refinamiento se aplicó el criterio mixto de rea máxima con un valor de 520 y de diferencia de intensidad de los pixeles con un valor de asociado 50 y como criterio se usó la selección del punto de mayor error. Los resultados del proceso se aprecian en la Tabla 9.

Triángulos Totales	666
Arcos Totales	1998
Vértices Totales	336
Segundos de Proceso	2102317
Minutos de Proceso	35

TABLA 9 : RESULTADOS PRUEBA D

Las imágenes que a continuación se muestran corresponden al proceso inicial, el resultado después de los filtros y finalmente la triangulación obtenida.

FIGURA 22 : IMAGÉN “D”

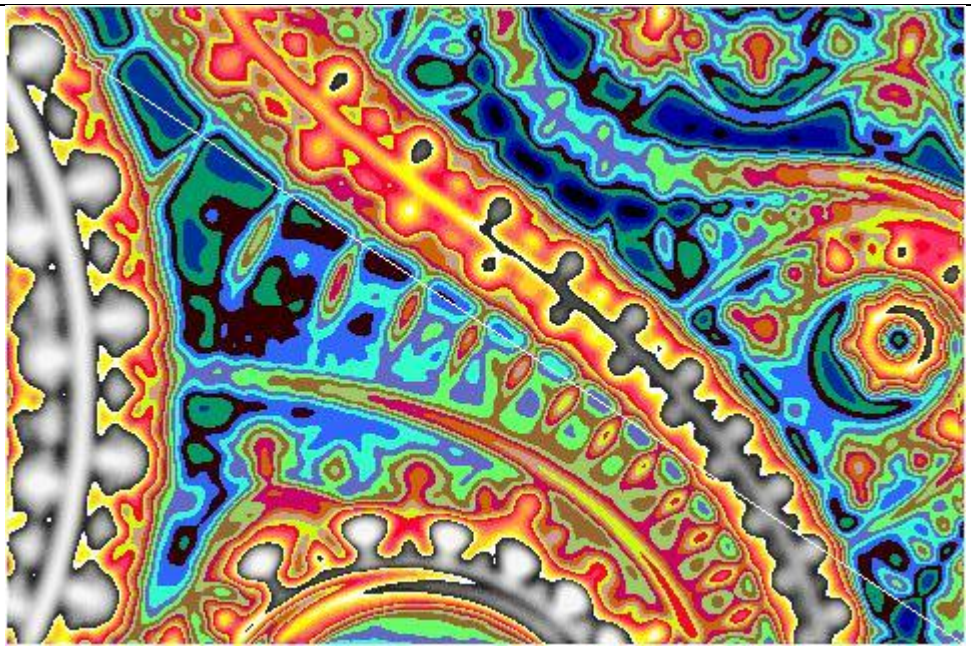

FIGURA 23 : IMÁGEN "D" DESPUES DE LOS FILTROS

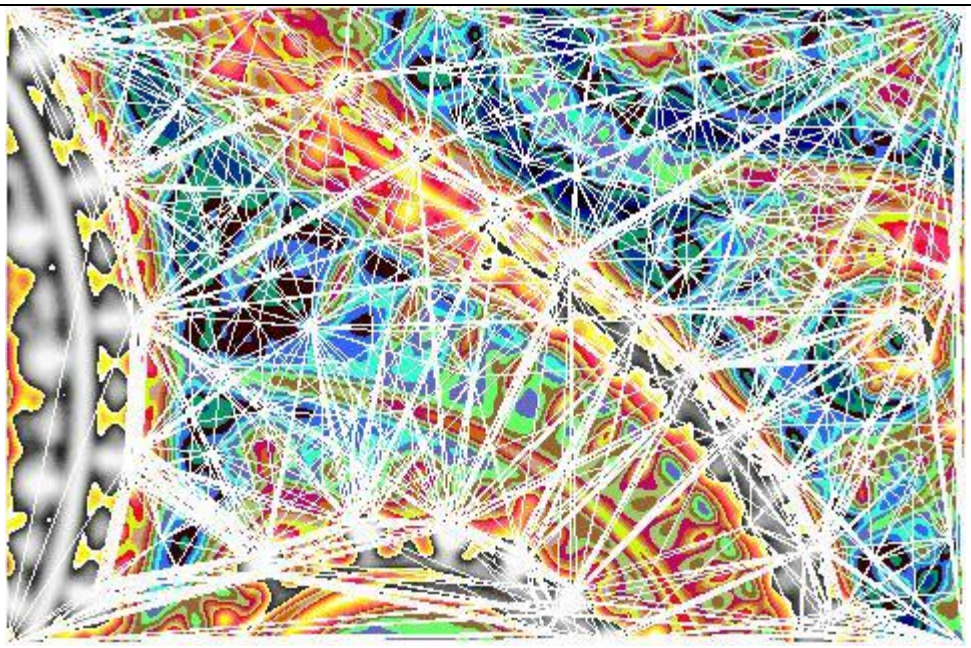

FIGURA 24 : IMÁGEN "D" CON SU MALLA

3.6.2 Pruebas no Funcionales

En el análisis preliminar se establecieron tres métricas a ser cumplidas como parte del nuevo diseño, aquí se presentan los resultados de las mismas.

3.6.2.1 Trabajo desde archivos de imágenes o Geometrías PSLG

La métrica establecida fue “las geometrías a ser utilizadas en el proceso de refinamiento deben poder ser obtenidas desde archivos, ya sean éstos de imágenes o archivos de texto”.

Para el correcto cumplimiento se efectuó una serie de acciones. Primero se desacopló la representación de la geometría de la imagen, conviviendo geometrías asociadas a polilíneas e imágenes indistintamente. Segundo se generó código para la lectura de archivos de extensión “poly”.

3.6.2.2 Proceso de Generación de una Malla en Pasos o Etapas

El proceso de generación de una malla se debe entender como una secuencia de pasos independientes, los cuales son ejecutables por separado. De esta forma, la malla geométrica obtenida de un paso puede ser un resultado en sí.

Para lograr este punto se diseñó el software como una biblioteca, la cual posee dentro de su API los métodos:

- geometría: Permite instanciar una geometría desde uno de los tipos de archivos reconocidos como fuente válida.
- mallaInicial: Para una geometría particular genera una malla inicial. refinamientoMejora
- mallaFinal: Para una malla y geometría en particular genera una malla final.
- refinamientoMejora: Permite depurar una Región de una Malla aplicando ciertos algoritmos y criterios.

Al tener esta API es posible realizar el proceso de refinamiento de manera iterativa repitiendo cuantas veces se requiera el método “refinamientoMejora”.

Aquí se incluye un ejemplo en donde a una geometría cargada desde el archivo “geometría.poly” se le aplican dos proceso de refinamiento, uno por el largo del lado y el otro por área máxima.

```
ParametrosDTO dto = new ParametrosDTO();

dto.geomtriaArchivoEsImagen = false;

dto.geomtriaArchivoEsPoly = true;

dto.geometriaRutaArchivo = " geometría.poly";

dto.geometriaPSLGFabrica = new GeometriaPSLGConvexDelanuayBridge();


dto.mallaInicial = new AlgoritmoDeMallaPSLGConvex();

dto.mallaFinal = new AlgoritmoDummyDeMallaFinal();


{

AlgoritmoCriterioDTO algoritmoCriterioDTO = new AlgoritmoCriterioDTO();

//

algoritmoCriterioDTO.criterio = new CriterioMaximoLargoDelLado(100);
```

```
algoritmoCriterioDTO.seleccionPunto = new SeleccionPuntoBaricentro();
algoritmoCriterioDTO.refinamientoMejora = new AlgoritmoDeMejoraDelaunay();
//
dto.algoritmosCriterios.add(algoritmoCriterioDTO);
}
{
AlgoritmoCriterioDTO algoritmoCriterioDTO = new AlgoritmoCriterioDTO();
//
algoritmoCriterioDTO.criterio = new CriterioAreaMaxima(500);
algoritmoCriterioDTO.seleccionPunto = new SeleccionPuntoBaricentro();
algoritmoCriterioDTO.refinamientoMejora = new AlgoritmoDeMejoraDelaunay();
//
dto.algoritmosCriterios.add(algoritmoCriterioDTO);
//
}


GeneradorDeMalla aRefinar = new GeneradorDeMalla();
// Instanciar la Geometría
Geometria aGeometria = aRefinar.geometria(dto);
// Malla Inicial
Malla aMalla = aRefinar.mallaInicial(aGeometria,dto);
// Instanciar geometria Completa
Region aRegion = new GeometriaCompleta();
// Refinar
```

```
dto = filtros(caso,dto,aGeometria);

//

for (AlgoritmoCriterioDTO algoritmoCriterioDTO : dto.algoritmosCriterios) {

        aRefinar.refinamientoMejora(algoritmoCriterioDTO, aMalla, aRegion);

}

//

aRefinar.mallaFinal(aGeometria, aMalla, dto);
```

3.6.2.3 INCORPORACIÓN DE NUEVOS CRITERIOS O ALGORITMOS

La incorporación de nuevos criterios o algoritmos debe ser realizada con el mínimo esfuerzo. Esto debiera ser posible sin necesidad de alterar ninguna componente ya existente, ni tampoco conocer la implementación de las demás piezas de software.

Para cumplir con esto en la fase de diseño se elaboró una serie de Hot Spot, los cuales quedan representados por las componentes:

- Criterio: Permite definir si un triángulo debe o no ser depurado
- RefinamientoMejora: Proceso que recorre los triángulos de la geometría y selecciona aquello que no cumplen con el criterio, modificando la malla por medio de la creación de un nuevo punto y la mantención de las propiedades de la misma.
- SeleccionPunto: Mecanismo que permite seleccionar el nuevo punto del triángulo para ser incorporado en la malla.
- ImagenFiltro: Proceso de alteración de las propiedades del tono de la imagen.
- MallaInicial: Algoritmo para generar la malla inicial del proceso de refinamiento.

- MallaFinal: Algoritmo para generar la malla final del proceso de refinamiento.

Una vez terminada la creación de los criterios ya existentes se incluyeron algunos Hot Spot nuevos:

- SeleccionPunto: Se incorporó el baricentro y ortocentro como criterios de selección de punto.
- ImagenFiltro: Se incorporó un filtro que transforma los tonos de la imagen a escala de grises.
- GeometriaPSLGFabrica: Se creó una componente que permite crear mallas iniciales para geometrías definidas como PSLG convexas.

4 CONCLUSIONES

En el desarrollo de la presente memoria se abordó el objetivo de mejorar un software de automatización de generación de mallas geométricas, para lo cual se revisó desde el diseño original del producto hasta los detalles de la implementación final.

Como meta del proceso de rediseño se buscaron e identificaron los patrones de diseño que gobiernan el comportamiento del producto. El primero de ellos es Strategy el cual permite aislar las algoritmos y criterios en componentes que puedan ser utilizados en el proceso de generación de la malla. Iterator es utilizado para recorrer las diversas etapas en la generación de una malla. Finalmente Bridge se ha utilizado para conectar la nueva aplicación con componentes de código abierto para el manejo de PSLG.

Otro objetivo logrado fue la modificación del diseño de software para que este perdiera acoplamiento entre los conceptos de geometría e imagen. Para ello se aisló el proceso de carga de la geometría y se utilizó la clase abstracta “Geometría” para representar cualquier posible origen. Lo anterior permite que los Hot Spot destinados al refinamiento y mejora sean utilizados indistintamente en cualquier geometría, como por ejemplo el “Algoritmo de Mejora de Delaunay”.

La principal mejora del diseño es su reenfoque a los “Hot Spot”. El nuevo software contiene interfaces las cuales permiten extender cualquiera de las componentes claves del producto. De esta forma es posible contar con la posibilidad de crear nuevos algoritmos o criterios para el proceso de depuración o mejora de la malla. Así mismo la inclusión de componentes para los procesos de “Generar Malla Inicial” y “Generar Malla Final”.

Se han elaborado todos los Hot Spot tendientes a cubrir los criterios de refinamiento, criterio de selección de punto, algoritmos y filtros de imagen que ya se encontraban presentes en la implementación existente.

En el proceso de generación automática de la malla geométrica se requiere una geometría como punto de partida, un objetivo logrado consiste en la posibilidad de que dicha geometría sea cargada desde un archivo con una imagen o un dominio poligonal. Más aún, se ha habilitado un Hot Spot que permite en el futuro tener nuevas formas de cargar geometrías iniciales, así es posible incluir diversos nuevos formatos de geometría sin que esto necesariamente requiera alterar las piezas vinculadas a los procesos de refinamiento o mejora de la malla.

4.1 TRABAJO FUTURO

Una mejora requerida en el futuro es la posibilidad de trabajar con geometrías asociadas a PSLG no convexas. El actual algoritmo de malla inicial para PSLG convexas es utilizable con polígonos no convexos pero la malla resultante puede contener desviaciones no deseables.

En el presente trabajo se ha incluido el formato “poly” como nuevo origen de geometría, es esperable poder contar con otros formatos de archivos asociados a PSLG.

El nuevo software se ha concentrado en el rediseño del producto lo cual ha permitido al obtención de una biblioteca para la generación automática de mallas geométricas. Se ha excluido del proceso una interfaz gráfica que permita explotar las nuevas potencialidades del diseño, por lo cual, este se considera un punto vital de mejora.

Una extensión esperable es la posibilidad de acotar el proceso de mejora a una región que no sea la geometría completa. Para ello se requiere la implementación de nuevas especializaciones de la componente Región.

5 BIBLIOGRAFÍA

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of reusable object oriented sotware*. Addison-Wesley, 1995.

[2] Wikipedia. Wikipedia Z3. [Online]. http://es.wikipedia.org/wiki/Z3

[3] P. Coad and E. Yourdon, *Object-Oriented Análisis*. New Jersey, U.S.A.: Prentice-Hall, Inc., 1991.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, *El proceso Unificado de Desarrollo de Software*. Madrid: Pearson Educación S.A., 2000.

[5] D. Alur, J. Crupi, and D. Malks, *Core J2EE Patterns Best Practices and Design Strategies*. San Antonio Road: Prentice Hall, 2001.

[6] M. C. Bastarrica and N. Hitschfeld-Kahler, "Designing a Product Family of Meshing Tools," *Advances in Engineering Software*, no. 37, pp. 1-20, 2005.

[7] P. Aguilar Vergara, *Reconocimiento de Bordes en Imágenes Aplicado a Anillos de Árboles*. Santiago, Chile: Departamento de Ciencias de la Computación, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, 2008.

[8] P. Aguliar Vergara, *Reconocimiento Automático de Bordes en Imágenes Aplicado a Anillos de Árboles*. Santiago, Chile: Departamento de Ciencias de la Computación, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, 2008.

[9] Wikipedia. Wikipedia Software Framework. [Online]. http://en.wikipedia.org/wiki/Software_framework

[10] M. Bertrand, "Applying 'Design by Contract'," *Computer*, vol. 25, no. 10, pp. 40-51, Oct. 1992.