**Universidad de las Ciencias Informáticas**

**Facultad 5**

# MÓDULO DE TRANSFORMACIÓN A ENTIDADES 2D

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

**Autor:** José Ángel Lores Estrada

**Tutor:** Msc. Marvyn Amado Márquez.

**Co-tutor:** Ing. Fabio Casado Ruiz.

# DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con el mismo.

Para que así conste firmo la presente a los 28 días del mes de junio del año 2012.

_____ _____

**Autor: José Angel Lores Estrada** **Tutor: MsC. Marvyn Amado Márquez**

_____-

**Co-tutor: Ing. Fabio Casado Ruiz**

# AGRADECIMIENTOS:

*A esos dioses gracias a los cuales soy la mejor persona que puedo ser, gracias a los cuales estoy aquí, y he podido llegar a ser el profesional en que hoy me veo convertido, gracias a los cuales he labrado mi camino de victorias. A los seres que más amo en este mundo, y por los cuales vivo. A mis padres María y Ángel. A Ale, el mejor hermano del mundo.*

*A la memoria de mi abuela, que no logró ver su primer nieto ingeniero, y de Rosita, que fue en vida para mí como mi segunda madre.*

*A Griselita y Luis Raúl, mi salida de socorro en muchas situaciones extremas.*

*A mis amigos desde primero, que me demostraron que son poquitos los amigos de verdad: Yanet, Leya, Betty, José Carlos, Karel, Luanner y Jairo, y a los que se fueron sumando que me dejaron la misma enseñanza: Javier, Miguel Ángel, Adrián, Diomne, Daylén y Dianne. A Annabel, más que mi amiga, mi psicóloga, mi confidente, mi consejera, mi hermana.*

*A la personita especial que sabe que le debo la tranquilidad cuando el estrés me dominaba, que le agradezco los buenos ratos, las risas y hasta los llantos, y que para poder resumirlo todo, le agradezco su paciencia, y su compañía por tanto tiempo.*

*A mis mejores colegas, los de fiestas, tomaderas, café, playa y buenos y divertidos ratos: Leo, Manuel, Arian, Danilo, Gendor, Sasha y Fernando. A mi mano siempre salvadora y esa voz con su consejo y su frase siempre precisa: Yuneiry.*

*A mis profes, sin cuyo conocimiento y paciencia hoy no sería nadie: Dayrén, Zaida, Zoraida, Yaima, Ileana, Reina, Raisel, Brenda, Lida y Yeyli. A mi tutor, sin cuyo apoyo no hubiera podido.*

# DEDICATORIA:

*A los dueños de mis sueños, escultores de mi carácter, forjadores de mis triunfos y guías de mi existir. A mis padres que con su apoyo incondicional, su confianza incluso en los peores momentos, su constancia, entrega, esfuerzo y gran sacrificio, me han traído hasta este lugar. A ellos va dedicado mi trabajo, y no solo eso, a ellos va dedicada mi vida.*

# RESUMEN:

El proyecto “Centro de Desarrollo y Simulación de Estructuras Mecánicas” (CDSEM), del Centro de Informática Industrial (CEDIN), se encuentra involucrado en la creación de la herramienta de diseño e ingeniería asistidos por computadora “GALBA-CAD”, para el apoyo en la creación de taladros de perforación de petróleo. Dentro de las funcionalidades que la herramienta debe poseer, se encuentra la aplicación a las distintas geometrías en 2 dimensiones (2D) existentes en su modelador geométrico de las transformaciones geométricas lineales afines: rotación, traslación y escalado.

La herramienta GALBA-CAD no posee las funcionalidades previamente mencionadas, lo que trae como consecuencia que no se puedan crear modelos a partir de primitivas 2D, por no contar con una forma de modificar su posición y tamaño, haciendo imposible combinarlas. Por tanto, surge la necesidad de dotar a la herramienta de estas funcionalidades, objetivo que se pretende lograr con la realización del presente trabajo.

Se realizó un estudio sobre las tecnologías utilizadas con estos fines, y las variantes para llevarlas a la práctica. De esta investigación se tomaron las mejores prácticas y conceptos encontrados, utilizándolos como punto de partida para el desarrollo de una solución.

Se obtuvo un módulo que fue incorporado a la herramienta GALBA-CAD, encargado de aplicar las transformaciones geométricas antes mencionadas, cumpliendo las expectativas del cliente.

**Palabras clave:** escalado, modelador geométrico, primitivas 2D, rotación, transformaciones, traslación.

# ABSTRACT:

The project "Centre for Development and Simulation of Mechanical Structures" (CDSEM), belonging to the “Center for Industrial Computing” (CEDIN), is involved in the creation of a computer aided design and engineering tool, called "Galba-CAD" for giving support in creating oil drilling rigs. Among the features that the tool should have, is the application of geometric transformations: rotation, translation and scaling, to the different geometries in the sketcher.

Galba-CAD tool does not have the aforementioned features, which results in that will not be possible to create models from 2D primitives, because the user don´t have a way to change its position and size, making it impossible to combine them. Thus, arises the necessity to provide the tool with these capabilities, objective to be achieved with the completion of this work.

A study on the technologies used for these purposes, and variants for implementing them was made, taking from this research the best practices and concepts found, and using them as a starting point for developing a solution.

It was obtained a module which was incorporated to Galba-CAD tool, responsible for implementing geometric transformations mentioned above, fulfilling customer expectations.

**Keywords:** 2D primitives, rotation, scaling, sketcher, transformations, translation

# ÍNDICE:

# ÍNDICE DE FIGURAS



# ÍNDICE DE TABLAS

# INTRODUCCIÓN:

La automatización de los procesos industriales, ha dado lugar a un notable avance de la industria. Esto ha sido posible debido a la conjunción de varios factores; entre ellos: las nuevas tecnologías en el campo mecánico, la introducción de las computadoras en la industria, y el control y la regulación de sistemas y procesos. De todos ellos, la incorporación de las computadoras en la producción podría catalogarse como el elemento fundamental que permite llevar a cabo la automatización integral de los procesos industriales. La aparición de la microelectrónica y los microprocesadores, ha facilitado el desarrollo de técnicas de control complejas, la robotización, la implementación de sistemas de gobierno y la planificación.

Se ha evidenciado en los últimos años una dependencia creciente del uso de modelos bidimensionales (2D) y tridimensionales (3D) obtenidos mediante computadoras, para la modelación y solución de problemas industriales. El uso de estos modelos ha llegado a convertirse en imprescindible para un buen desempeño profesional en ramas específicas tales como la arquitectura; para la creación de planos y vistas virtuales de edificios, la cartografía; donde se utilizan para la digitalización de planos, las ciencias empresariales y económicas, la escultura, la ingeniería mecánica e industrial y la animación. En todos los casos la visualización de esos modelos requiere de elementos básicos como son:

- Un sistema gráfico.
- La creación de entidades geométricas (líneas, círculos, puntos y arcos).
- La aplicación de transformaciones geométricas a estas entidades.

Con la finalidad de obtener estos modelos, que respondan a las crecientes necesidades de la industria e ingeniería modernas, han surgido disímiles tecnologías y herramientas que facilitan el trabajo a los desarrolladores. Se ha apreciado entre ellas, un especial auge de las conocidas como tecnologías CAD/CAE/CAM, que por sus siglas en inglés equivalen a "***C****omputer* ***A****ided* ***D****esign*", "***C****omputer* ***A****ided* ***E****ngineering*" y "***C****omputer* ***A****ided* ***M****anufacturing*", o lo que es lo mismo; tecnologías de diseño, ingeniería y fabricación asistidos por computadora.

El CAD atiende prioritariamente aquellas tareas exclusivas del diseño, tales como el dibujo técnico y la documentación del mismo, pero normalmente permite realizar otras tareas complementarias relacionadas principalmente con la presentación y el análisis del diseño realizado [1]. El término CAE engloba el conjunto de herramientas informáticas que permiten analizar y simular el comportamiento del producto diseñado [2], y por último el término CAM se refiere al uso de computadoras para ayudar en todas las fases de la fabricación de un producto, incluyendo la planificación del proceso y la producción, mecanizado, calendarización, administración y control de calidad, con una intervención mínima del operario [3].

La introducción en la industria cubana de estas tecnologías converge hacia una gestión integral en todos los procesos y actividades desarrolladas dentro de la empresa, mediante un único sistema informático. Esto facilita el diseño y desarrollo de nuevos productos e implica la disminución de la separación entre el diseño y la fabricación. La Universidad de las Ciencias Informáticas, como vanguardia de nuestro país en la explotación y puesta en funcionamiento de las más modernas tecnologías en beneficio de la sociedad, igualmente ha incursionado en el uso de este tipo de tecnologías y herramientas. En este sentido, el proyecto CDSEM, perteneciente al CEDIN, de la Facultad 5, se encuentra en proceso de desarrollo de un proyecto conjunto con la empresa venezolana PDVSA y la Industria Chino-Venezolana de Taladros (ICVT), para el apoyo a la creación de taladros de perforación de pozos de petróleo, con su punto culminante en la entrega de una herramienta actualmente en desarrollo, nombrada “GALBA-CAD”.

Todas las herramientas CAD tienen en común la utilización de una interfaz (modelador geométrico 2D o 3D) donde son creados y visualizados los gráficos e imágenes que serán utilizados por los usuarios finales para elaborar sus diseños. De los módulos que componen un modelador geométrico, aquel donde se realiza el trabajo de creación y modificación de entidades 2D, es conocido como *“sketcher”.* La obtención de estos gráficos e imágenes mediante computadoras trae consigo gran cantidad de ventajas, entre las cuales destaca la facilidad con que sobre ellos se puedan realizar transformaciones para hacerlos en apariencia y funcionalidad más fieles a la realidad, y para permitir combinarlos para dar lugar a diseños más complejos. De esta forma el cliente que use software CAD para obtener sus gráficos puede, en dependencia de sus necesidades, realizar operaciones específicas que le permitan alcanzar un objetivo determinado; por ejemplo: un cartógrafo puede modificar a su antojo la escala de los mapas con los cuales trabaja, un arquitecto obtener diferentes panorámicas de las obras que diseña, un gerente o un

estadista modificar las escalas de las gráficas que utiliza, o un animador modificar a su antojo la posición de un personaje. Todos estos cambios se pueden realizar de una forma relativamente sencilla, debido a que toda imagen gráfica se codifica y almacena en la computadora en forma de números, los cuales son fácilmente modificables.

A la hora de hacer la selección por parte del equipo del proyecto, de las tecnologías y herramientas a utilizar para desarrollar GALBA-CAD, se tuvo en cuenta que existen numerosas variantes y plataformas para desarrollar aplicaciones CAD/CAE/CAM. De todas ellas, a petición del cliente, se tomó como punto de referencia para el diseño de la interfaz gráfica y funcionalidad de la herramienta, su homóloga privativa Autodesk® Inventor®. Existen novedosas opciones dentro del mundo del software libre que pueden ser utilizadas con tales fines, destacándose entre ellas una plataforma muy fácil de usar y que permite el desarrollo rápido de aplicaciones sofisticadas de diseño, conocida como OCAF (*Open CASCADE Application Framework*), basada como su nombre lo indica en la tecnología OpenCASCADE, considerada entre las más poderosas para el desarrollo de aplicaciones CAD/CAE/CAM, y que presenta un núcleo de modelado 3D basado en bibliotecas de objetos reutilizables en C++; además de un conjunto amplio de herramientas de desarrollo [4]. Es por ello que luego de valoradas las diferentes opciones con que se contaba, se concluyó en la selección de la tecnología OCAF.

La herramienta “GALBA-CAD”, posee un modelador geométrico sobre el cuál se realiza todo el trabajo de diseño de las piezas del taladro. Estas piezas serán visualizadas en el mismo como combinaciones de entidades 2D, las cuales deben poder ser transformadas en cualquier momento por el diseñador para la construcción de dichas piezas. Las transformaciones geométricas a elementos 2D forman una parte importante en la creación y manipulación de imágenes y objetos en computación gráfica, pues permiten alterar de una forma uniforme toda la imagen y hacerlo siempre a conveniencia del usuario.

Una transformación geométrica, es una función matemática que transforma un punto o conjunto de puntos en otro punto o conjunto, dentro del sistema de referencia global de la escena. Las transformaciones dan la posibilidad al diseñador de alterar de forma uniforme toda la imagen y permiten cambiar la posición, orientación y tamaño de los objetos. Dentro de las transformaciones lineales afines más importantes abordadas en el presente trabajo se encuentran: la traslación, la rotación y escalado. Se

aprecia además cómo pueden ser expresadas de una forma sencilla mediante multiplicaciones de matrices. [5]

La herramienta “GALBA-CAD”, carece de un módulo que contenga implementadas las funcionalidades necesarias para realizar dichas transformaciones a las entidades 2D. Sin estas funcionalidades no sería posible, como exige el usuario, poder modificar en todo momento la apariencia y posición de los objetos 2D dentro del *“sketcher”* del modelador geométrico, sin tener que volver a modelar o componer de nuevo todo el objeto, tampoco definir un objeto en cualquier parte de la escena, mover o rotar dichos objetos para tener otra perspectiva de la misma cuando esto sea necesario, ni sería posible para los mecánicos (usuarios finales de la herramienta) ajustar primitivas de acuerdo a su posición, rotación o escalado para conformar el diseño de una estructura mecánica, por lo que en gran medida la herramienta no cumpliría con los requisitos funcionales bajo los cuales fue concebida, y acarrearía el descontento del cliente.

Es por ello que se evidencia en este caso el siguiente **problema a resolver:**

¿Cómo aplicar transformaciones a los elementos 2D presentes en el Modelador Geométrico 2D de la herramienta GALBA-CAD?

El **objeto de estudio** se orienta alrededor de la aplicación de transformaciones geométricas a elementos 2D, mientras que el **campo de acción** se enmarca en las transformaciones lineales afines a entidades 2D en un modelador geométrico utilizando OpenCASCADE.

Con el objetivo de dar solución al problema, se plantea como **objetivo general:**

Desarrollar un módulo para aplicar transformaciones lineales afines a los elementos 2D presentes en el *“sketcher”* del modelador geométrico de la herramienta GALBA-CAD.

Para darle solución a este problema, se plantearon los siguientes **Objetivos Específicos**:

1. Realizar un estudio para la elaboración del marco teórico y el estado del arte inicial referente a los temas.

2. Elaborar análisis y diseño del módulo.

3. Implementar el módulo.

4. Validar la solución propuesta, comparando los resultados obtenidos con los obtenidos en la herramienta Autodesk® Inventor®.

Los métodos de investigación científica y los procedimientos utilizados para darle cumplimiento a las tareas planteadas fueron:

**1- Métodos teóricos:**

a) **El método histórico:** para conocer la evolución y desarrollo de los sistemas CAD/CAE/CAM, identificando las características generales y esenciales de su funcionamiento que pudieran ser de utilidad.

b) **El método lógico:**

El método lógico utilizado fue el **método sistémico**, para lograr que todos los elementos de la concepción del modelo formen un todo que funcione de manera integrada a partir del estudio de las funcionalidades del elemento individual, y la adecuación de sus relaciones con los restantes (relaciones entre CAD/CAE/CAM – OpenCASCADE – OCAF – modeladores geométricos – elementos 2D y transformaciones geométricas a esos elementos, que converjan en el resultado final como un todo).

**2. Métodos Empíricos:**

a) **El método de la observación científica y la medición** se emplean para la identificación del problema, el estudio objetivo del estado del arte, el diagnóstico y la evaluación de la aplicación.

b) **El método coloquial** se emplea para la presentación y discusión de los resultados en sesiones científicas (Seminarios de Tesis, Jornada Científica, Pre-defensa y Defensa).

**3. Procedimientos Científicos:**

a) **El procedimiento de análisis** se emplea para la descomposición del objeto de estudio en partes y propiedades para el estudio de sus relaciones y componentes. Se complementa con la síntesis en la

sistematización de los conocimientos mediante el entendimiento y descubrimiento de las relaciones esenciales y características generales entre las partes previamente analizadas.

b) **El procedimiento de la abstracción** se emplea para la comprensión del objeto de estudio. [6]

Como **resultado esperado** de este trabajo, se pretende lograr la incorporación del módulo de transformaciones lineales afines de entidades 2D a la herramienta GALBA-CAD.

El presente trabajo tiene la siguiente estructura: resumen, introducción, cuatro capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, glosario de términos y anexos.

En el **Capítulo 1**: se hace un estudio acerca de las herramientas CAD/CAE/CAM y su puesta en práctica en proyectos reales a través del uso de la tecnología OpenCASCADE, y más específicamente de su *framework* OCAF. Se presentan los conceptos y definiciones que ayudan al desarrollo y comprensión de la solución propuesta. Además se hace un estudio de las transformaciones geométricas que se le pueden aplicar a entidades 2D dentro del modelador geométrico de una herramienta desarrollada mediante el uso de OCAF. Por último, se justifica el porqué de la selección de OpenCASCADE y no de tecnologías homólogas para la realización del trabajo.

En el **Capítulo 2:** se describe y desglosa la solución desde el punto de vista de la ingeniería de software, describiendo el ciclo seguido desde el modelo de dominio hasta la etapa de diseño. Además se mencionan y seleccionan las herramientas que se van a emplear en la realización del trabajo.

En el **Capítulo 3**: se brindan los resultados de la etapa de diseño e implementación y se realiza una descripción de las clases diseñadas y se brinda una panorámica de los algoritmos implementados, su funcionamiento y la descripción de las clases y operaciones pertenecientes a OpenCASCADE que fueron utilizadas.

En el **Capítulo 4:** se dan los resultados de las pruebas realizadas a la aplicación, y de detectarse se muestran las principales dificultades y sus correcciones.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo son presentados los conceptos y definiciones que ayudan en el desarrollo y comprensión de la solución propuesta. Se describen las principales características de las herramientas CAD/CAE/CAM, la tecnología OpenCASCADE, su *framework* OCAF, los modeladores geométricos 2D y las transformaciones geométricas a elementos 2D. Además, tomando estas descripciones como punto de partida, se justifica el porqué de la selección de las tecnologías que se van a emplear en la realización del trabajo.

## 1.1 Herramientas CAD/CAE/CAM

Hace poco más de 30 años los diseños industriales producidos alrededor del mundo eran realizados con lápiz o tinta sobre un papel como un dibujo común. Los cambios menores o correcciones debían hacerse borrando la parte incorrecta del diseño, y redibujando; mientras que para realizar cambios mayores debían reelaborarse completamente, en especial si los cambios de un diseño afectaban otros diseños. En dichos casos también se requería la capacidad de identificar la manera en que un cambio repercutía en otros diseños. En ocasiones surge la problemática de la no existencia de métodos para predecir el comportamiento de los productos finales de la industria. Muchas veces luego de fabricado el producto, son descubiertas deficiencias en su diseño que dan al traste con la imposibilidad de su puesta en explotación.

Ante la necesidad de dar solución a estas problemáticas, surgen los sistemas CAD/CAE/CAM. Esta tecnología podría descomponerse en numerosas disciplinas pero normalmente, abarca el diseño gráfico, el manejo de bases de datos para el diseño y la fabricación, control numérico de máquinas herramientas, robótica y visión computarizada. Actualmente estos sistemas están conectados a los sistemas de gestión y producción de tal forma que ya desde la fase de diseño se puede saber el coste del producto final y controlar los *stocks* de componentes y materiales para su fabricación.

El concepto de “diseño asistido por computadora” representa el conjunto de aplicaciones informáticas que permiten a un diseñador “definir” el producto a fabricar. Gracias a estas tecnologías se consigue una mayor productividad en el trazado de planos, integración con otras etapas del diseño, mayor flexibilidad,

mayor facilidad de modificación del diseño, ayuda a la estandarización, disminución de revisiones y mayor control del proceso de diseño [7]. Para realizar “ingeniería asistida por computadora”, se dispone de programas que permiten calcular cómo va a comportarse la pieza en realidad, en aspectos tan diversos como deformaciones, resistencias, características térmicas y vibraciones. Se puede comprobar, por ejemplo, si una pieza metálica será capaz de resistir altas temperaturas y si los cables de un puente colgante serán capaces de soportar el tablero del mismo y el peso de los carros que transitarán por encima de él [8].

La ingeniería CAM hace referencia concretamente a aquellos sistemas informáticos que ayudan a generar los programas de control numérico (CN) necesarios para fabricar las piezas en máquinas con control numérico computarizado (CNC). A partir de la información de la geometría de la pieza, del tipo de operación deseada, de la herramienta escogida y de las condiciones de corte definidas, el sistema calcula las trayectorias de la herramienta para conseguir el mecanizado correcto, y a través de un postprocesado genera los correspondientes programas de CN con la codificación especifica del CNC donde se ejecutarán. En general, la información geométrica de la pieza proviene de un sistema CAD, que puede estar o no integrado con el sistema CAM. [7]

Debido a la necesidad de no correr riesgos en el proceso de diseño y fabricación de sus taladros de perforación de pozos de petróleo, para evitar así millonarias pérdidas económicas y demoras en el montaje de plantas y plataformas de extracción; y teniendo en cuenta las características de los sistemas CAD/CAE/CAM, la empresa conjunta PDVSA-ICVT tomó la decisión de solicitar la creación de un software propio de diseño e ingeniería asistidos por computadora, que resolviera sus problemas y le brindara tranquilidad en el proceso de fabricación de los taladros. De esta forma surge el convenio entre dicha empresa y la Universidad de Ciencias Informáticas.

Para una mejor comprensión de las características específicas de cada una de estas tecnologías se brinda a continuación una breve descripción de ellas.

### 1.1.1 Tecnologías y herramientas para el desarrollo CAD/CAE/CAM

Entre la gran variedad de tecnologías y herramientas para el trabajo con CAD/CAE/CAM, muchas de ellas se han vuelto más populares y han brindado un número mayor de prestaciones que otras, lo que ha

introducido en el mercado un nivel competitivo que lleva en todo momento a buscar mejoras e innovación, que se traducen en mayor funcionalidad y facilidad de uso de muchas de estas herramientas que hoy en día tienen un funcionamiento muy potente e intuitivo. Las versiones más populares son aquellas que corren bajo el sistema operativo Windows, pero también existen nuevas y novedosas variantes bajo software libre.

#### 1.1.1.1 Programas CAD para el sistema operativo Windows.

Autodesk AutoCAD

Autodesk AutoCAD es una plataforma de diseño asistido por computadora para dibujo en dos y tres dimensiones. Actualmente es desarrollado y comercializado por la empresa Autodesk. Es un software reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos de edificios o la recreación de imágenes en 3D. Es uno de los programas más usados, elegido por arquitectos, ingenieros y diseñadores industriales. El programa gestiona una base de datos de entidades geométricas (puntos, líneas, arcos) con la que se puede operar a través de una pantalla gráfica en la que dichas entidades son mostradas, el llamado editor de dibujo o modelador geométrico. Además, procesa imágenes de tipo vectorial, aunque permite incorporar archivos de tipo fotográfico o mapa de bits, donde se dibujan figuras básicas o primitivas (líneas, arcos, rectángulos, textos), y mediante herramientas de edición se crean gráficos más complejos. [9]

Ashampoo 3D CAD Architecture

Ashampoo 3D CAD Architecture es una herramienta potente de diseño 2D y 3D, que cuenta con propiedades muy interesantes como son: diseño 3D, componentes inteligentes, diseño de planos en 2D/3D, exportación a imágenes de realidad virtual, y cálculo de materiales y costo para una determinada construcción. Posee características como: un asistente de inicio y asistente de proyectos, construcción libre de objetos en 3D, editor de superficies para el diseño individual de superficies, extensas funciones 2D, ayudas a la construcción y entradas, vistas 2D y 3D (también en paralelo) y extensos catálogos con objetos, materiales, texturas y símbolos [10].

Autodesk® Inventor®

El software de CAD en 3D, Autodesk® Inventor®, ofrece una gama completa y flexible de programas para diseño mecánico en 3D, simulación de productos, creación de herramientas, ingeniería a la carta y comunicación de diseños. Inventor lleva más allá de la tercera dimensión hacia prototipos digitales ya que permite producir un modelo 3D de gran precisión que puede ayudarlo a diseñar, visualizar y simular los productos antes de ser construidos. La creación de prototipos digitales con Inventor contribuye a que las compañías puedan diseñar mejores productos, reducir los costos de desarrollo y llegar al mercado más rápido [11].

#### 1.1.1.2 Programas libres para GNU/Linux.

Ninguna de las herramientas anteriores es tomada en cuenta a la hora de la selección por parte del cliente, pues este opta por contar con una herramienta concebida bajo software libre, condición que ninguna de ellas cumple. Estas herramientas son de licencia privativa, con altos costos en el mercado por el pago de dichas licencias, restricciones para su utilización, y un código fuente no reutilizable y desconocido, y no ajustado a las necesidades concretas de PDVSA e ICVT.

DraftSight

Es una especie de clon de AutoCAD con características interesantes, gratuito, pero tiene la desventaja de que solo trabaja en 2D. Está enfocado a estudiantes y profesionales y permite crear y editar archivos en formato AutoCAD (.dwg). La interfaz del programa es altamente funcional. Las herramientas están bien distribuidas, y se maximiza el área de dibujo mediante el empleo de desplegables, navegación por pestañas en algunas secciones y menús que pueden ocultarse [12]. Maneja las entidades básicas 2D como: arcos, círculos y líneas; permite insertar notas, y manipular las dimensiones de las entidades (longitud del arco, radio, diámetro, línea base) Es multiplataforma, por lo que también corre bajo los sistemas operativos Windows y MAC.

Qcad

Aplicación gratuita muy usada, enfocada al dibujo en dos dimensiones. Mediante ella se pueden crear vistas de piezas, planos de edificios y esquemas. Soporta capas de vistas, bloques, diversas fuentes de

texto, unidades métricas e imperiales, impresión a escala, herramientas de medición, bibliotecas con piezas predefinidas, y herramientas para construcción y modificación de puntos, líneas, círculos, elipses, polilíneas, textos, dimensiones, sombreados, rellenos e imágenes de trama [13]. Igualmente es multiplataforma, por lo que también corre bajo los sistemas operativos Windows y MAC.

LibreCAD

Es una propuesta gratuita de código abierto, que brinda las herramientas básicas necesarias para el trabajo, sin perderse en las complejas y sofisticadas funciones de toda herramienta CAD. Ofrece una herramienta de CAD 2D, derivada de QCad. El proyecto ha modificado algunas partes del código original para resolver problemas de licencias y se distribuye bajo GPLv2. Cuenta con una interfaz de usuario sencilla basada en las bibliotecas de Qt4. Dispone de un panel de herramientas dinámico muy completo, que esconde muchas más funciones de las que pueden apreciarse a simple vista. Tiene soporte para capas, funcionalidad esencial en este tipo de software. LibreCAD sólo aborda el dibujo 2D, es compatible con ficheros DXF, que importa muy bien, y CXF, pero no soporta ficheros DWG. Está disponible también para Microsoft Windows, Mac OS X y algunas de las principales distribuciones de GNU/Linux (Debian, Ubuntu, Fedora, Mandriva, Suse, entre otras). El programa está traducido a 20 idiomas, entre ellos el español, aunque parcialmente [14].

En todos los casos anteriores las herramientas fueron obviadas durante la selección por estar diseñadas solo para el trabajo en 2 dimensiones, mientras para el cliente es indispensable el trabajo tanto en entornos 2D como 3D.

Es por ello que el cliente tomó la decisión de solicitar la creación de una herramienta propia, diseñada bajo los paradigmas del software libre y 100% ajustada a sus objetivos específicos, la modelación y construcción de taladros de perforación petrolera.

Una vez llegado a un convenio con el equipo de desarrollo perteneciente al CEDIN, este se vio ante la necesidad de seleccionar una biblioteca que le proporcionara una base para el proceso de desarrollo y que contuviera la mayoría de las funcionalidades necesarias para dar cumplimiento a las peticiones del cliente.

## 1.2 Selección de tecnologías

El equipo de desarrollo, para hacer la selección de la tecnología que le serviría de base para la creación de la futura herramienta, valoró como opciones 2 de las más populares y potentes en esta rama en la actualidad: VTK y OpenCASCADE.

***VTK*** (Visualization Toolkit) es un conjunto de herramientas para el trabajo con gráficos en 3D, procesamiento de imágenes y visualización, de código abierto y disponible gratuitamente. Consiste en una biblioteca de clases en C++ y varias capas de interfaz. Fue desarrollada por "Kitware" encargado además de continuar ampliando el conjunto de herramientas, y que ofrece soporte profesional y servicios de consultoría relacionados con VTK. Soporta una amplia variedad de algoritmos de visualización (escalar, vectorial, tensorial, textura y métodos volumétricos), y avanzadas técnicas de modelado como: modelos implícitos, reducción de polígonos, suavizado de malla, corte y contorno. Cuenta con un extenso marco de visualización, y un conjunto de *widgets* de interacción 3D, compatibles con el funcionamiento en paralelo. Se integra con varias bases de datos de herramientas GUI como Qt y Tk. Es multiplataforma y funciona en Windows, Linux, Mac y Unix. También permite el ajuste automatizado del núcleo de C++ a Python, Java y Tcl, lo que permite a las aplicaciones VTK que puedan ser escritas también en esos lenguajes de programación [15].

***OpenCASCADE*** es una biblioteca especializada en creación de aplicaciones CAD, CAM, CAE, para el diseño y simulación de objetos y piezas enfocado a ingenieros y profesionales. Permite crear superficies sólidas en 3D, desarrollar aplicaciones y simular ensayos. En el epígrafe 1.3 se brinda una explicación detallada de esta tecnología que fundamenta su selección por parte del equipo de desarrollo para llevar a cabo el desarrollo de la herramienta.

No fue seleccionada VTK por parte del equipo de desarrollo por problemas de eficiencia. Al ser comparado su funcionamiento con el de OpenCASCADE hacía un uso mucho mayor de memoria y recursos en la PC.

## 1.3 OpenCASCADE y OCAF

Dentro de las variantes que han surgido para la implementación y trabajo con las tecnologías CAD/CAE/CAM en código libre, una de las más potentes, con mayor número de funcionalidades, y con mayor popularidad es la tecnología OpenCASCADE.

Open CASCADE (abreviatura de las siglas en inglés "*Computer Aided Software for Computer Aided Design and Engineering*"), desarrollada inicialmente a principios de los años 90 por "Matra Datavision", es una aplicación orientada a usuarios que se desenvuelven dentro del área del diseño, como ingenieros, arquitectos y diseñadores. Gracias a sus bondades es posible diseñar y modificar cualquier modelo bidimensional o tridimensional deseado con una facilidad asombrosa.

Cuenta con una licencia de funcionamiento GPL, lo que permite utilizarla de forma ilimitada en cualquier ordenador. Se comporta como un entorno de desarrollo integrado, orientado a la edición y el diseño de superficies y objetos bidimensionales y tridimensionales, utilizando para su diseño una gran cantidad de opciones de personalización que facilitan enormemente el modelado de estas figuras u objetos.

Cuenta con la capacidad de realizar análisis gráficos de cualquier superficie, lo que se traduce en estimar y simular los datos generados por las curvas de los gráficos de forma numérica y rápida, facilitando los estudios de ingeniería. Por si fuera poco, ofrece la posibilidad de importar y exportar figuras y posee un apartado de usuario simple y bastante amigable. Además viene con un paquete completo de efectos especiales y texturas que se pueden añadir a los diseños de modo que se puedan personalizar por completo [16].

La manera más sencilla de trabajar con OpenCASCADE es a través de su *framework* OCAF, que por sus siglas en inglés significa "*Open CASCADE Application Framework*". OCAF es una plataforma fácil de usar y que permite el desarrollo rápido de aplicaciones sofisticadas de diseño. Una aplicación típica desarrollada con OCAF puede tener un modelador geométrico en dos o tres dimensiones, herramientas de fabricación o de análisis, y aplicaciones de simulación o herramientas de ilustración. Cuenta con un sinnúmero de características que la convierten en una de las opciones más atractivas a la hora de hacer una selección de tecnología para el trabajo con CAD/CAE/CAM, y que justifican su selección en este caso, como son:

- Facilidad de diseño de la arquitectura de la aplicación.
- Definición de los componentes y la forma en que cooperan.
- Definición del modelo de datos capaz de soportar las funcionalidades requeridas.
- Estructuración del software en relación a:
    - Sincronizar la visualización con los datos. Los comandos de visualización de objetos tienen que actualizar la vista una vez que estos son llamados.
    - Soportar comandos para las operaciones “deshacer” y “rehacer” (Ctrl+Z). Esta función debe ser tomada en cuenta muy tempranamente en el proceso de diseño, y hecho así, funciona muy eficazmente.
- Permite la implementación de las funciones para salvar los datos. Si la aplicación tiene un ciclo de vida muy largo, la compatibilidad de los datos entre las versiones tiene que ser tratada.
- Facilidad para crear una interfaz de usuario.

Gracias a tener implementadas todas estas funciones, permite desarrollar aplicaciones en un tiempo bastante rápido, permitiendo que a la hora de desarrollar una aplicación, solo sea necesario concentrarse en las funcionalidades específicas, y no en la esencia del código que se maneja, factor decisivo para su selección final.

OpenCASCADE y OCAF utilizan como interfaz visual para el manejo de las entidades, una herramienta conocida como modelador geométrico, que por sus características hace que el resultado final resulte muy sencillo e intuitivo para el usuario.

## 1.4 Modeladores geométricos 2D

Los modeladores geométricos 2D son usados para generar geometrías y perfiles que serán usados posteriormente para la construcción de superficies y sólidos. Se basan en entidades geométricas vectoriales como puntos, líneas, arcos y polígonos, con las que se puede operar a través de una interfaz gráfica. Son una herramienta muy poderosa que permite modelar de una manera sencilla todo tipo de objetos. Estimulan la creatividad y la capacidad intelectual del usuario respecto al manejo de la geometría, e introducen y refuerzan el conocimiento de las tres dimensiones; de ahí la decisión de utilizarlos como interfaz visual para las herramientas CAD, y por supuesto para la herramienta GALBA-CAD.

Los modeladores pueden ser además de 3 dimensiones (3D), en cuyo caso su complejidad y por tanto sus potencialidades aumentan, y que cuentan con 3 módulos diferentes. De ellos los módulos que son comunes a los modeladores 2D son:

- ***"sketcher"*:** Con la finalidad de diseñar modelos mediante el empleo de primitivas 2D y herramientas de diseño con operaciones ágiles e intuitivas.
- **Dibujo técnico o plano:** Módulo que permite crear planos con las vistas de los modelos o ensamblajes de forma automática en muy poco tiempo.

Dichos módulos agrupan un conjunto de operaciones que facilitan el cumplimiento de sus propósitos, por ejemplo: *Fillet*, *Chamfer y Trim***.** Además se encuentran entre estas operaciones las transformaciones geométricas isométricas[1] y anamórficas[2], cuya explicación detallada se explica en el epígrafe siguiente.

Una característica indispensable que deben poseer las entidades generadas dentro del modelador geométrico 2D, es la posibilidad de aplicación sobre ellas de las diferentes transformaciones geométricas; para lograr así un mayor realismo y funcionalidad en la construcción mediante su combinación de las futuras superficies y sólidos.

## 1.5 Transformaciones a entidades 2D

Las entidades ubicadas dentro del *"sketcher"* del modelador geométrico 2D se definen mediante un conjunto de puntos. Las transformaciones no son más que procedimientos para calcular nuevas posiciones de esos puntos, con lo que cambia el tamaño y la orientación de la entidad que ellos conforman. Se puede resumir en que: consiste en transformar el sistema de coordenadas del objeto dentro del sistema de coordenadas del mundo. Las principales transformaciones geométricas lineales afines que se le pueden aplicar a estas entidades 2D son: traslación, rotación, escalado.

[1] Es aquella transformación geométrica que conserva las dimensiones y ángulos de las figuras a transformar. También son conocidas como movimientos.
[2] Una transformación es considerada anamórfica cuando cambia la forma de la figura original.

### 1.5.1 Representaciones de matriz y coordenadas homogéneas

En muy pocas ocasiones en problemas de la vida real, es enfrentada la necesidad de aplicar solamente una de estas transformaciones, sino que de forma general, se aplican como una secuencia continua de ellas para lograr un efecto determinado. Muchas de estas transformaciones se pueden llevar a cabo con la simple suma o multiplicación de vectores, mientras otras requieren la utilización de matrices.

Es por esta causa que a la hora de combinar más de una transformación, surge la necesidad de encontrar un estándar que permita realizarlo sin importar si se trabaja con vectores o matrices; por lo cual finalmente se utiliza el sistema de coordenadas homogéneas. Para expresar cualquier transformación bidimensional como una multiplicación de matriz, se representa cada posición de coordenadas cartesianas (x, y) con las 3 coordenadas homogéneas ($x_h$, $y_h$, h) donde:

$$x = \frac{x_h}{h}\,,\ y = \frac{y_h}{h}$$

Estas coordenadas homogéneas también pueden ser representadas de la forma (h*x, h*y, h). Se puede seleccionar el parámetro h como cualquier valor distinto de cero, pues en caso de asignarle ese valor las coordenadas “x” y “y” se indefinirían. Se toma como convenio utilizar siempre el valor h = 1, y de esta forma se representa cada posición bidimensional con las coordenadas homogéneas de la forma (x, y, 1). Expresar posiciones en coordenadas homogéneas da la posibilidad de representar todas las ecuaciones de transformación geométrica como multiplicaciones de matrices. Se representan las coordenadas con vectores de columnas de 3 elementos y las operaciones de transformación se expresan como matrices de 3x3 [17].

### 1.5.2 Traslación de entidades 2D

Se aplica una traslación en un objeto para cambiar su posición a lo largo de la trayectoria en una línea recta, de una dirección de coordenadas a otra. Para hacer esto se convierten los puntos bidimensionales que conforman la entidad, agregándole las distancias de traslación *tx* y *ty* a la posición de coordenadas original (x,y), para mover los puntos a una nueva posición (x’, y’). De esta forma la traslación queda definida mediante vectores de la siguiente manera:

$$x' = x + tx,\ y' = y + ty$$

El par de distancia de traslación (tx, ty) es conocido como vector de traslación o vector de cambio. La traslación es una transformación isométrica, que mueve los objetos sin deformarlos. Es decir, se traslada cada punto del objeto la misma distancia.

Para aplicar una traslación a un segmento de línea recta, se aplica la ecuación de transformación en cada uno de los extremos de la línea, y se vuelve a trazar la línea entre las nuevas posiciones de los extremos. Los polígonos se trasladan al sumar el vector de traslación a la posición de coordenadas de cada vértice que lo conforma, y se vuelve a generar el polígono con el nuevo conjunto de coordenadas de vértices, y las especificaciones actuales de los atributos [17].

La fórmula de la traslación llevada a la notación de coordenadas homogéneas queda de la forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

### 1.5.3 Rotación de entidades 2D

Se aplica una rotación bidimensional en un objeto 2D al cambiar su posición a lo largo de la trayectoria de una circunferencia en el plano xy. Para generar una rotación, se especifica un ángulo de rotación α y la posición ($x_r$, $y_r$) del punto de rotación (o punto pivote) en torno al cual se gira el objeto. Los valores positivos para el ángulo de rotación definen rotaciones en sentido opuesto a las manecillas del reloj alrededor del punto pivote, y los valores negativos giran los objetos en la dirección de las manecillas. Si se toma como pivote el origen de coordenadas, conocidas las coordenadas iniciales del punto a rotar ($x_r$, $y_r$), y el ángulo de rotación, la nueva posición del punto (x', y') luego de realizada la rotación, estará dada por las fórmulas:

$$x' = x_r \cos\alpha - y_r \, sen\,\alpha$$

$$y' = x_r \, sen\,\alpha + y_r \cos\alpha$$

Si el punto pivote no se encontrara en el origen de coordenadas, como casi siempre suele pasar, las ecuaciones cambiarían. Si se consideran ($x_c$ ,$y_c$) como las coordenadas del punto pivote, y (x, y) las coordenadas del punto a rotar, con α como el ángulo de rotación, las fórmulas quedan conformadas de la siguiente forma:

$$x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\,sen\,\alpha$$

$$y' = y_c + (x - x_c)\,sen\,\alpha + (y - y_c)\cos\alpha$$

Llevándola a la notación de coordenadas homogéneas, la fórmula queda de la forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \text{[17]}$$

### 1.5.4 Escalado de entidades 2D

Una transformación de escalado es aquella que altera el tamaño de un objeto. Esta operación se realiza tanto para líneas, arcos y circunferencias, como para polígonos; multiplicando los valores de coordenadas (x, y) de cada vértice por los factores de escalado $s_x$ y $s_y$ para producir las coordenadas transformadas (x', y'):

$$x' = x * s_x,\ y' = y * s_y$$

El factor de escalado $s_x$ escala objetos en la dirección de "x", mientras $s_y$ lo hace en la dirección del eje "y". Se pueden asignar valores numéricos positivos cualesquiera a los factores $s_x$ y $s_y$. Los valores menores que 1 reducen el tamaño de los objetos y los acercan al origen de coordenadas, mientras que los valores mayores que 1 producen una ampliación de los objetos a la vez que los alejan del centro del sistema cartesiano. Cuando se asigna el mismo valor a $s_x$ y $s_y$ se genera una escalado uniforme que mantiene las proporciones relativas de los objetos. Por otra parte cuando $s_x$ y $s_y$ tienen valores distintos, se obtiene un estiramiento o escalado diferencial. Si el factor de escalado toma un valor negativo se genera una simetría del objeto inicial, lo que no corresponde con el concepto de escalado

Se puede controlar la localización de un objeto escalado al seleccionar una posición llamada punto fijo, que debe permanecer sin cambios después de la transformación de escalado. Este punto fijo de coordenadas $(x_f, y_f)$ puede ser uno de los vértices, el centroide del objeto, o cualquier otra posición. Así, se escala un polígono con respecto al punto fijo al escalar la distancia desde cada vértice al punto fijo. Para un vértice con coordenadas (x, y), se calculan las coordenadas escaladas (x', y') como:

$$x' = x_f + (x - x_f)\, s_x \,,\; y' = y_f + (y - y_f)\, s_y$$

Llevándola a la notación de coordenadas homogéneas, la fórmula queda de la forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Para poder producir una secuencia de transformaciones con estas ecuaciones; por ejemplo escalado, seguido de rotación, y luego traslación; se deben calcular las coordenadas transformadas un paso a la vez. Primero se escalan las posiciones de coordenadas, después se giran estas coordenadas escaladas, y por último se trasladan estas coordenadas giradas [17].

Los algoritmos para llevar a cabo estas transformaciones pueden tornarse desde el punto de vista matemático bastante complejos, afortunadamente otra de las razones por las que se optó por el uso de la biblioteca OpenCASCADE, es porque la misma viene dotada de funcionalidades que ejecutan estas transformaciones, recayendo la complejidad en la forma en que dichas funcionalidades son utilizadas y adaptadas al entorno de la arquitectura bajo la cual fue concebida la herramienta GALBA-CAD. La forma en que los mismos fueron aplicados e integrados a la arquitectura del sistema en general para llegar a obtener el resultado esperado serán explicados y podrán comprendidos a los largo de los capítulos dedicados a la descripción de la solución y las características del sistema.

## 1.6 Conclusiones del capítulo:

Se analizaron los conceptos fundamentales relacionados con el campo del diseño, ingeniería y fabricación asistidos por computadora como vía de solución a las principales problemáticas de la industria moderna. Entre las diferentes formas de llevarlos a la práctica, se pudo realizar un estudio para finalmente seleccionar para el presente trabajo la que cuenta con mayor número de ventajas y potencialidades, la biblioteca OpenCASCADE y su *framework* de desarrollo OCAF, de los cuales se conocieron las principales características a tener en cuenta para la comprensión de la solución propuesta. Se sentaron las bases necesarias para llegar a la total comprensión del objeto de estudio y su campo de acción, se mostraron los principales procedimientos a seguir para aplicar transformaciones geométricas y se pudo comprender el uso que se le da a cada uno de los parámetros de entrada específicos de cada tipo de transformación.

# CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN

En este capítulo se abordará parte de lo referente a la solución propuesta, describiendo el ciclo seguido como parte de la ingeniería de software desde el modelo de dominio, pasando por los requisitos funcionales y no funcionales, hasta la descripción de los casos de uso derivados de dichos requisitos funcionales. Se hace referencia además a la selección de tecnologías, herramientas y metodologías que fue llevada a cabo para el presente trabajo.

## 2.1 Metodología de desarrollo de software

Todo proceso de desarrollo de software es riesgoso y difícil de controlar y, si no se sigue una metodología para controlar los flujos de trabajo, probablemente se obtendrán clientes insatisfechos con el resultado. Por tanto, uno de los principios de la universidad es, a la hora de desarrollar un software, seguir desde su concepción una metodología que garantice la ulterior satisfacción de los clientes. Una metodología de desarrollo de un proceso de software es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación a ser llevados a cabo por los desarrolladores de sistemas informáticos [18].

En el presente trabajo de diploma se optó para su utilización como metodología de desarrollo de software RUP (por sus siglas en inglés) o Proceso Unificado de Desarrollo de Software. Fue seleccionada por ser altamente robusta y estar preparada para llevar a cabo cualquier tipo de proyecto de desarrollo, y por haber sido la utilizada en todos los procesos de desarrollo a lo largo del plan de estudio. Por tener un diseño orientado a objetos se hace más fácil la comprensión a alto nivel para su posterior implementación. Dentro de sus principales características se encuentran:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean. El proceso de desarrollo de software avanza a través de una serie de flujos que parten de los casos de uso, se puede afirmar que estos proporcionan un hilo conductor y una guía para todo el proceso.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo y describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo

económicamente. La arquitectura debe diseñarse para que el software evolucione, no solo en su desarrollo inicial, sino también a lo largo de las futuras generaciones.

- **Iterativo e incremental:** RUP propone que cada proyecto se desarrolle en fases y que cada fase se desarrolle en iteraciones, donde cada iteración resulta en un incremento del proceso de desarrollo, lo cual se realiza de forma planificada y culmina con el cumplimiento del punto de control trazado en la fase [19].

## 2.2 Lenguaje de Modelado

Los lenguajes de modelado visual permiten especificar, construir, documentar y visualizar artefactos de un sistema de software. Se seleccionó para el presente trabajo la utilización del lenguaje UML. El mismo está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

La selección del lenguaje vino dada por ser un lenguaje hasta cierto punto universal, pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos [20].

## 2.3 Herramientas utilizadas para el desarrollo de la solución.

Para el desarrollo de la solución que se propone fue necesario el uso de un grupo de herramientas de desarrollo y de apoyo al proceso de desarrollo. Fue necesario un entorno de desarrollo, una herramienta de modelado y una SDK que brindara soporte al proceso de desarrollo entre otras.

Como herramienta de apoyo al proceso de desarrollo propuesto por RUP y herramienta de modelado se utilizó Visual Paradigm, herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite obtener todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar

documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML, presenta licencia gratuita y comercial y es fácil de instalar y actualizar, y compatible entre ediciones. Todas estas características propiciaron su selección [21].

Como entorno de desarrollo se optó por la utilización de Eclipse, equipado con un *plugin* que permite la comunicación y uso de las potencialidades de la plataforma Qt4. En ambos casos, además de por ser los utilizados por parte del equipo de desarrollo para la implementación de la herramienta GALBA-CAD; lo que obliga a su utilización para el desarrollo del módulo, la selección tuvo en cuenta las características y ventajas de ambos.

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Fue liberado originalmente bajo la "*Common Public License"*, pero después fue re-licenciado bajo la "*Eclipse Public License"*. La "*Free Software Foundation*" ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL). Eclipse dispone de un editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene soporte para pruebas unitarias, control de versiones con CVS, asistentes (*wizards*) para creación de proyectos, clases, pruebas y refactorización. Asimismo, a través de "*plugins*" libremente disponibles es posible añadir control de versiones con Subversion, e integración con Qt [22].

Qt es una biblioteca multiplataforma para desarrollar aplicaciones mediante el uso del lenguaje C++, las cuales pueden ser con o sin interfaz gráfica. Es utilizada en populares herramientas como Autodesk Maya, "*Dassault DraftSight*", "*Google Earth*", *KDE*, "*Adobe Photoshop*" y "*VLC media player*". Qt utiliza el lenguaje de programación C++ de forma nativa. Funciona en todas las principales plataformas, y tiene una amplia comunidad. Es distribuida bajo los términos de GNU LGPL, es software libre y de código abierto. Su herramienta de desarrollo QtDesigner es la utilizada para la creación de la interfaz visual de la herramienta GALBA-CAD. QtDesigner es un diseñador de interfaces gráficas de usuario multiplataforma. Permite la creación rápida de formularios con la tecnología de los *layouts* que facilitan la tediosa tarea de disponer componentes visuales en el formulario [23].

## 2.4 Modelo de dominio

El modelo del dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software [19].

Entre las clases conceptuales existen relaciones que describen el entorno. Dichas relaciones tienen lugar cuando el ingeniero mecánico utiliza dentro del modelador geométrico las herramientas de diseño, para crear o modificar una estructura. En la figura se muestra la descripción del dominio con el objetivo de facilitar la comprensión del sistema.

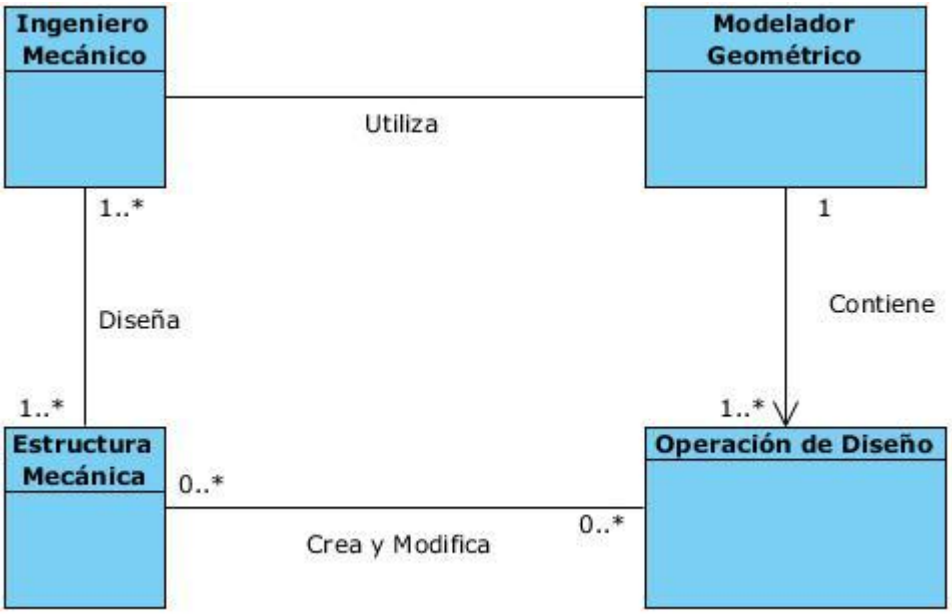


**Figura 1 Diagrama de clases del dominio**

### 2.4.1 Descripción de los procesos del negocio propuestos

En el Centro de Informática Industrial (CEDIN), dedicado a desarrollar productos y servicios informáticos de automatización industrial y computación gráfica, se desarrollan de forma simultánea varios proyectos productivos con fines específicos. Entre ellos, el proyecto “Centro de Desarrollo y Simulación de

Estructuras Mecánicas" (CDSEM) se especializa en la creación y emulación del funcionamiento mediante técnicas de computación gráfica de piezas y estructuras mecánicas. Este proyecto asumió, desde hace alrededor de 1 año, conjuntamente con la multinacional "Petróleos de Venezuela S.A" (PDVSA) y la "Industria Chino Venezolana de Taladros" (ICVT), la elaboración de una herramienta que fuera capaz de realizar el diseño y la simulación de fabricación y funcionamiento asistidos por computadora de los taladros de perforación petrolera utilizados por esa empresa.

La herramienta debe dar solución a un gran número de demandas que hacen los clientes, entre ellas que sea imitado el funcionamiento de la herramienta profesional *Autodesk*® *Inventor*® . Muchas de estas demandas ya han sido solucionadas por parte del equipo de desarrollo, y otras aún se encuentran en vías de solución. Para el logro de un elevado nivel de certeza y realismo en la elaboración de los bocetos de los taladros, la herramienta soportará diseños tanto en 2 como en 3 dimensiones, conformados en todos los casos por primitivas 2D, que pueden ser puntos, líneas, círculos, arcos de circunferencia o polígonos tanto regulares como irregulares. Una propiedad primordial que debe cumplir cualquier primitiva que sea utilizada en una herramienta de este tipo, es la posibilidad de que sobre ella puedan ser aplicadas transformaciones geométricas elementales, que garanticen al cliente en todo momento poder conformar a partir de ella los diseños de las piezas que desea modelar, convirtiéndose más que en un requisito, en una prioridad indispensable para el correcto funcionamiento del sistema. Cuando el usuario selecciona una de estas primitivas la aplicación debe brindarle la posibilidad de rotar, trasladar o escalar la misma con la dirección y longitud deseadas por el cliente. Una vez seleccionada la primitiva a transformar y el tipo de transformación a aplicar, la misma debe ser ejecutada y ser redibujada la primitiva en la nueva posición.

### 2.4.2 Definiciones de los principales conceptos.

**Ingeniero Mecánico:** Especialista en la rama de la mecánica que será el encargado del diseño de los prototipos de estructuras mecánicas, y que hará uso del módulo para aplicarle transformaciones geométricas a los mismos.

**Estructura Mecánica:** Son todas las primitivas que pueden ser creadas o modificadas por el ingeniero mecánico a partir de las operaciones de diseño para, en conjunción con otras similares dar lugar al diseño final del prototipo.

**Modelador Geométrico:** epígrafe **1.4**

**Herramienta de diseño:** Son herramientas con las que cuenta el modelador geométrico que permiten crear o modificar una estructura mecánica, es este caso por ejemplo: las de crear líneas, polígonos círculos y arcos de circunferencia, y las que permiten rotar, trasladar y escalar dichas estructuras.

## 2.5 Captura de Requerimientos

Se conoce como requerimiento a la condición o capacidad que debe exhibir o poseer un sistema o componente para satisfacer un contrato, y que se pueden clasificar en requisitos funcionales y no funcionales. A continuación se exponen los requerimientos funcionales y no funcionales, que fueron definidos para el presente trabajo:

### 2.5.1 Requerimientos no funcionales del sistema

Los requerimientos no funcionales son propiedades o cualidades que debe tener el producto. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable. Entre los requerimientos no funcionales del sistema propuesto se encuentran:

- **Apariencia o interfaz externa:**
  - El diseño visual del módulo está definido como un clon de la herramienta Autodesk® Inventor®.
  - Optimizado para una resolución de 1024x768.
- **Software:**
  - Sistema operativo: Linux, preferentemente Ubuntu 10.04 en su compilación CAELinux, con arquitectura de 64 bits.
  - Dependencias de los paquetes ftgl-dev y libxmu-dev.
- **Hardware:**

- Las estaciones de trabajo donde se utilice la herramienta deben poseer al menos como capacidad recomendada 1 GB de memoria RAM.

❖ **Usabilidad:**

- Para hacer uso del módulo dentro del sistema es necesario poseer conocimientos elementales de computación y haber trabajado previamente en alguna herramienta de diseño como 3ds Max o Autodesk® Inventor®.
- Distribución y etiquetado de opciones bien definidos dentro del modelador para facilitar la interacción al usuario y hacerlo intuitivo.

❖ **Soporte:**

- Sistema bajo plataforma de software libre.
- El usuario contará con un manual de usuario que le servirá para orientarse en la función que va a realizar sobre el sistema u otra tarea en general.

### 2.5.2 Requerimientos funcionales del sistema

Luego de desarrollado el modelo de dominio, y estudiados los procesos del negocio, se analiza qué debe hacer el sistema para darle cumplimiento a los objetivos planteados. Para ello se enumeran a través de requerimientos funcionales que son capacidades o condiciones que el sistema debe cumplir. De acuerdo con los objetivos planteados, los requerimientos funcionales del sistema propuesto son:

**RF1:** Realizar rotación.

**RF1.1:** Capturar la entidad activa (seleccionada por el usuario) a la que le será aplicada la transformación.

**RF1.2:** Introducir el centro y ángulo de rotación.

**RF1.3**: Aplicar la rotación tomando dicho centro como eje y rotando tantos grados como indique el ángulo de rotación.

**RF2:** Realizar traslación.

**RF2.1:** Capturar la entidad activa (seleccionada por el usuario) a la que le será aplicada la transformación.

**RF2.2:** Introducir el vector de traslación.

**RF2.3**: Aplicar la traslación según la longitud y sentido del vector introducido

**RF3:** Realizar escalado.

**RF3.1:** Capturar la entidad activa (seleccionada por el usuario) a la que le será aplicada la transformación.

**RF3.2:** Introducir el centro y valor de escalado.

**RF3.3**: Aplicar el escalado según el centro y distancia introducidos.

## 2.6 Modelo de casos de uso del sistema

Mediante la explotación de las facilidades que brinda el lenguaje UML, y luego de capturados los requisitos funcionales del sistema, estos se representan mediante un diagrama de casos de uso. Primero se debe tener claro cuáles son los actores que van a interactuar con el sistema, y los casos de uso que representarán todas las funcionalidades del sistema. En este caso particular quien usará el sistema y por tanto desempeñará el rol de actor del sistema, será el ingeniero mecánico.

| Actores | Justificación |
|---|---|
| Ingeniero Mecánico | Interactúa con el sistema durante la ejecución de las funcionalidades que este brinda. Carga las primitivas o entidades 2D que fueron diseñadas con anterioridad, o dibuja nuevas entidades, ordenando al sistema la aplicación de la transformación deseada a la entidad seleccionada. Posteriormente puede modificar la entidad resultante a conveniencia. |

**Tabla 1 Actores del sistema**

### 2.6.1 Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema (DCUS) representa gráficamente los casos de uso y su interacción con los actores.

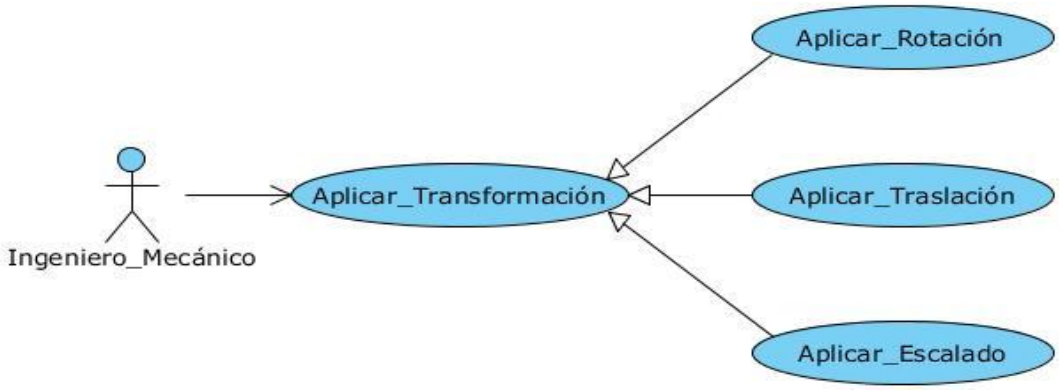


**Figura 2 Diagrama de Casos de Uso del sistema**

### 2.6.2 Descripción de los Casos de Uso del Sistema

| | |
|---|---|
| Caso de Uso: | Aplicar Transformación |
| Actores: | Ingeniero Mecánico |
| Propósito: | Aplicar una transformación geométrica a una entidad 2D. |
| Resumen: | Se inicia cuando el actor selecciona alguna de las opciones de aplicar transformaciones geométricas. Finaliza cuando es aplicada la transformación a la entidad seleccionada. |
| Precondiciones | Existe al menos una entidad 2D dibujada dentro del Modelador Geométrico |
| Flujo Normal de Eventos: | |
| Acción del Actor: | Respuesta del Sistema: |

| | |
|---|---|
| 1. Selecciona el ícono correspondiente a la transformación a aplicar.<br><br>2. Introduce los datos para aplicar la transformación | 1.1 Muestra una ventana que contiene las opciones para aplicar la transformación.<br><br>2.1 Captura los datos introducidos.<br>2.2 Aplica la transformación.<br>2.3 Muestra la entidad transformada. |
| Postcondiciones: | Se aplicó la transformación a la entidad. |
| Prioridad: | Crítica |

**Tabla 2 Descripción del Caso de Uso "Aplicar Transformación"**

| | |
|---|---|
| Caso de Uso: | Aplicar Traslación |
| Actores: | Ingeniero Mecánico |
| Propósito: | Trasladar una entidad 2D dentro del *"sketcher"*. |
| Resumen: | Se inicia cuando el actor selecciona la opción "*Move*" en el menú de opciones de transformación. Finaliza cuando es aplicada la transformación a la entidad seleccionada y según los parámetros introducidos. |
| Precondiciones | Existe al menos una entidad 2D dibujada dentro del Modelador Geométrico |
| Flujo Normal de Eventos: | |
| Acción del Actor: | Respuesta del Sistema: |
| 1. Escoge la opción "*Move*" | 1.1 Muestra una ventana con las opciones para realizar la |

| | |
|---|---|
| dentro del área de transformaciones. | traslación. |
| 2. Selecciona la opción "*Select*" y escoge una a una con el mouse la(s) entidad(es) que desea trasladar. | 2.1 Captura la(s) entidad(es) a trasladar. |
| 3. Selecciona la opción de introducir el punto de inicio de la traslación. | 3.1 Queda a la espera del evento click del mouse para capturar los puntos de inicio y fin de la traslación. |
| 4. Mediante eventos click del mouse, define los puntos de inicio y fin de la traslación. (Ver flujo alterno 1) | 4.1 La figura se traslada dentro del sistema de coordenadas de acuerdo al punto de inicio y el punto en el cual se encuentra el cursor. Se mantiene así hasta que se capture un segundo click, que será definido como punto final de la traslación. |
| 5. Presiona el botón "Done". | 5.1 Se muestra la entidad en su nueva posición. |
| Flujo Alterno de Eventos: | |
| 1.1 Selecciona la opción para trasladar una copia de la entidad y mediante eventos click del mouse selecciona los puntos de inicio y fin de la traslación. | 1.1.1 Captura los puntos de inicio y fin de la traslación y una copia de la figura se traslada dentro del sistema de coordenadas de acuerdo a los puntos definidos por el actor, manteniéndose la original en su posición inicial. |
| Postcondiciones: | Se trasladó la entidad y fue visualizada en su nueva posición. |
| Prioridad: | Crítica. |

**Tabla 3 Descripción del Caso de Uso "Aplicar Traslación"**

| Caso de Uso: | Aplicar Rotación |
|---|---|
| Actores: | Ingeniero Mecánico |
| Propósito: | Rotar una entidad 2D dentro del *"sketcher"*. |
| Resumen: | Se inicia cuando el actor selecciona la opción "*Rotate*" en el menú de opciones de transformación. Finaliza cuando es aplicada la rotación a la entidad seleccionada y según los parámetros introducidos. |
| Precondiciones | Existe al menos una entidad 2D dibujada dentro del Modelador Geométrico |

| Flujo Normal de Eventos: | |
|---|---|
| Acción del Actor: | Respuesta del Sistema: |
| 1. Escoge la opción "*Rotate*" dentro del área de transformaciones. | 1.1 Muestra una ventana con las opciones para realizar la rotación. |
| 2. Selecciona la opción "*Select*" y elige la(s) entidad(es) que desea rotar. | 2.1 Captura la(s) entidad(es) a rotar. |
| 3. Escoge la opción de seleccionar el punto eje de la rotación. | 3.1 Queda a la espera de que se introduzca el punto que será el centro de la rotación. |
| 4. Selecciona con el mouse el punto que desea como eje de rotación. | 4.1 Captura el punto que servirá como eje de rotación |
| 5. Introduce el ángulo de rotación deseado. (ver flujo alterno 1) | 5.1 La entidad rota de acuerdo al ángulo definido por el usuario en el área destinada para ello. |

| | |
|---|---|
| 6. Presiona el botón “Done” luego de insertado el ángulo deseado. | 6.1 Se muestra la entidad en su nueva posición. |
| Flujo Alterno de Eventos: | |
| 1.1 Selecciona la opción para hacer rotar una copia de la entidad e introduce el valor del ángulo.<br>1.2 Presiona el botón “Done” luego de obtenido el ángulo deseado. | 1.1.1 Rota una copia de la entidad de acuerdo al ángulo definido por el usuario.<br>1.2.1 Muestra una copia de la entidad en la nueva posición luego de efectuada la rotación y mantiene la original en su posición inicial. |
| Postcondiciones: | Se rotó la entidad y fue visualizada en su nueva posición. |
| Prioridad: | Crítica. |

**Tabla 4 Descripción del Caso de Uso “Aplicar_Rotación**

| | |
|---|---|
| Caso de Uso: | Aplicar Escalado |
| Actores: | Ingeniero Mecánico |
| Propósito: | Escalar una entidad 2D dentro del *“sketcher”*. |
| Resumen: | Se inicia cuando el actor selecciona la opción “*Scale*” en el menú de opciones de transformación. Finaliza cuando es escalada la entidad seleccionada según los parámetros introducidos. |
| Flujo Normal de Eventos: | |
| Acción del Actor: | Respuesta del Sistema: |
| 1. Escoge la opción “*Scale*” dentro del área de | 1.1 El sistema muestra una ventana con las opciones para realizar el escalado. |

| | |
|---|---|
| transformaciones.<br>2. Escoge la opción "*Select*" y elige la(s) entidad(es) que desea rotar.<br>3. Escoge la opción de seleccionar el punto de inicio del escalado.<br>4. Selecciona el punto que desea como inicio del escalado.<br>5. Introduce manualmente el factor de escalado deseado.<br>6. Presiona el botón "Done" luego de alcanzado el factor de escalado deseado. | 2.1 Captura la(s) entidad(es) a rotar.<br>3.1 Queda a la espera de que se introduzca el punto de inicio del escalado.<br>4.1 Captura el punto de inicio del escalado.<br>5.1 Escala la figura de acuerdo a dicho valor y el punto de inicio previamente definido.<br>6.1 Se muestra la entidad con su nuevo tamaño. |
| Postcondiciones: | Se escaló la entidad y fue visualizada en su nueva posición. |
| Prioridad: | Crítica. |

**Tabla 5 Descripción del Caso de Uso "Aplicar_Escalado"**

Luego de terminado el modelo de dominio, la captura de requerimientos, y el modelamiento de los casos de uso del sistema, se procede a la etapa de diseño, donde son creadas las clases del diseño, se concibe una idea más clara de la solución a desarrollar y a continuación se procede al proceso de implementación. Estas fases serán descritas en el capítulo siguiente.

## 2.7 Conclusiones del capítulo:

En el presente capítulo quedaron explicadas las herramientas, tecnologías y la metodología de desarrollo que fueron seleccionadas para desarrollar el módulo. Se pudieron comprender las características del modelo de dominio que envuelve el problema a resolver, y se dio inicio al proceso de desarrollo de software, pasando por las etapas de captura de requisitos y modelamiento de los casos de uso que garantizaran el cumplimiento de dichos requisitos funcionales.

# CAPÍTULO 3: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se explica la solución desarrollada al problema planteado; brindando una descripción de las diferentes clases de la aplicación. Luego del análisis realizado y la problemática existente, se propone desarrollar un módulo que permita incorporar las funcionalidades de aplicar transformaciones geométricas lineales afines a las entidades 2D que se encuentran dentro del modelador geométrico de la herramienta GALBA-CAD, facilitando interacción con el cliente y el mejor cumplimiento de las funcionalidades que el mismo espera ver cumplidas con el despliegue de la herramienta. La arquitectura del módulo será descrita con detalle a lo largo de este capítulo, fundamentando el papel que desempeña cada clase diseñada.

La aplicación de transformaciones a entidades 2D dentro de un entorno virtual puede ser interpretada como un procedimiento bastante sencillo de realizar, juicio precipitado para quienes lo plantean sin tener en cuenta las condiciones bajo las cuales las mismas serán aplicadas. El grado de dificultad de este módulo va mucho más allá de los algoritmos específicos para aplicar la transformación, sino que se enfoca en la forma de llevarlas a cabo y acoplarlas dentro de la arquitectura general de la herramienta GALBA-CAD, que además en su concepción intenta hacer una emulación bastante fiel de su homóloga privativa Autodesk® Inventor®, la cual maneja las trasformaciones geométricas de una forma un tanto peculiar, que será tenida en cuenta para llevar a cabo las fases de diseño e implementación. El primer paso en la etapa de diseño, es la creación del diagrama de clases del diseño.

## 3.1 Diagrama de clases del diseño

Dentro del flujo de trabajo de Diseño, la fase de elaboración de los diagramas de clases de diseño ocupa un importante papel, pues dichos diagramas muestran las clases finales del diseño con sus atributos y métodos y la forma en que se relacionan entre sí para la realización de los casos de uso.

Es precisamente la elaboración del diagrama de clases del diseño, el punto donde queda evidenciado el grado de dificultad del módulo que se propone, que como se planteó anteriormente, va mucho más allá de la simple aplicación de los algoritmos implementados por OpenCASCADE para la realización de transformaciones geométricas. Se diseñó un diagrama de clases que garantizara en su concepción el cumplimiento de los patrones GRASP, específicamente los patrones “Alta cohesión” y “Bajo

acoplamiento". Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para "*General Responsibility Assignment Software Patterns"* (Patrones Generales de Software para Asignar Responsabilidades). [24]

**Patrón Bajo Acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad [25] . En el presente modulo puede apreciarse en la forma en que se puede eliminar o adicionar un nuevo tipo de transformación al diagrama de clases, insertar nuevos estados, o prescindir de algunas funcionalidades o añadir otras sin que estos cambios repercutan negativamente en el funcionamiento del sistema.

**Patrón Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no afines, o un trabajo excesivo [25]. En el presente módulo se evidencia una alta cohesión funcional cuando las distintas funcionalidades de las clases diseñadas colaboran para producir algún comportamiento bien definido. El proceso seguido desde la selección por parte del usuario del tipo de transformación a aplicar hasta el dibujado de la nueva entidad en el sketcher, funciona de manera armónica mediante la conjunción acoplada de las responsabilidades asignadas a cada clase.

Luego del análisis de los requerimientos, los casos de uso y tomadas las decisiones de los patrones a tener en cuenta, se procedió a la creación del diagrama de clases del diseño:

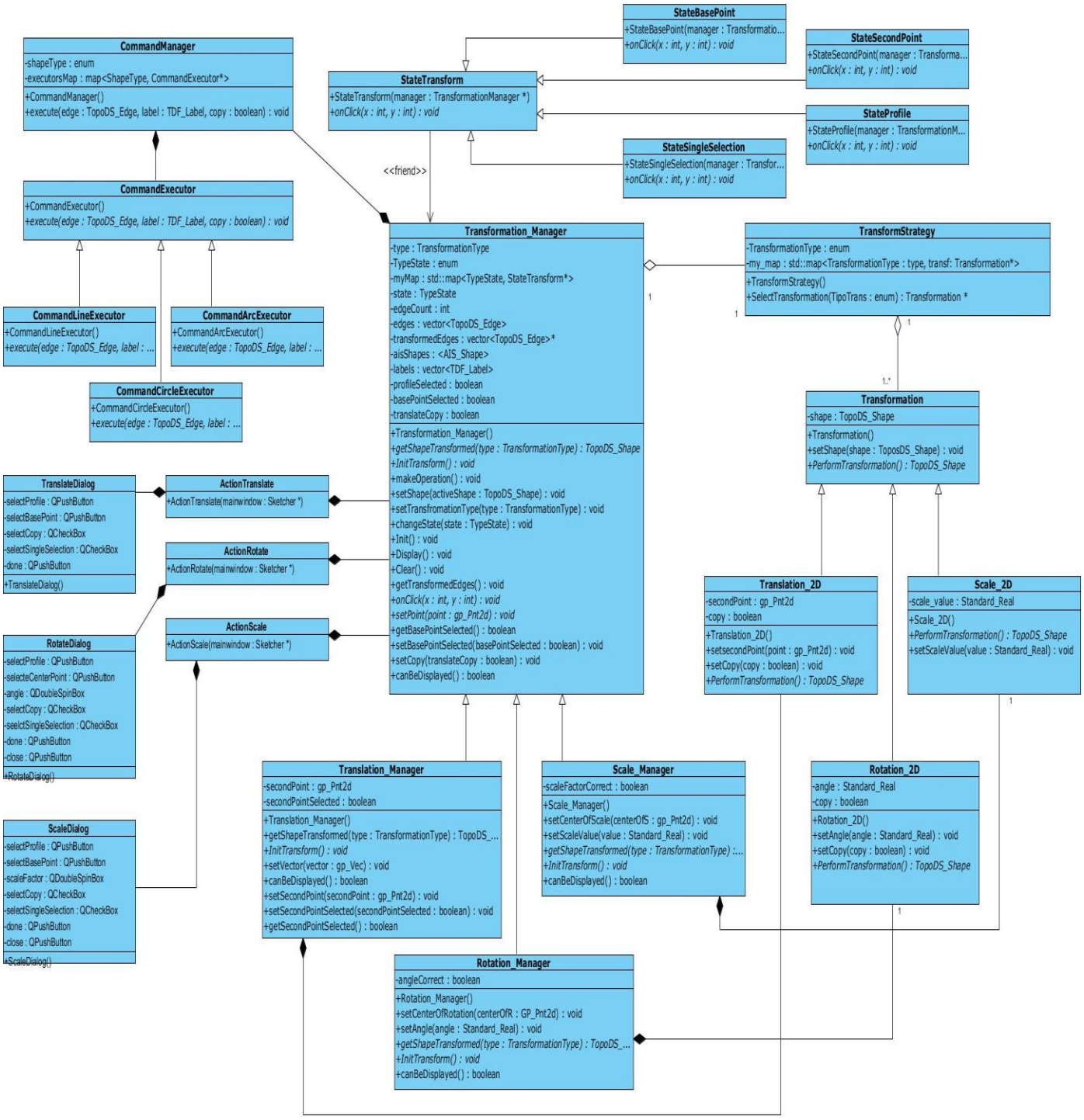


**Figura 3 Diagrama de Clases del diseño**

Se decide además el uso de los patrones de comportamiento "*Strategy*" y "*State*". Los patrones de comportamiento tienen que ver con los algoritmos y la asignación de responsabilidades entre los objetos. No describen solamente patrones de objetos o clases, sino también los patrones de comunicación entre ellos. Permiten abstraerse del control del flujo en tiempo de ejecución; que muchas veces puede tornarse complejo, con lo que brindan la posibilidad de darle una mayor importancia a la forma en que los objetos se interrelacionan entre sí. Estos patrones utilizan la herencia para distribuir el comportamiento entre las clases.

El patrón "*Strategy*" define una familia de algoritmos (por ejemplo en el caso de la clase "Transform_Strategy", donde es posible la instanciación indistintamente de un objeto de la clase "Rotation_2D", "Translation_2D" o "Scale_2D" ), encapsula cada uno, y los hace intercambiables, de modo que se seleccione en cada momento el que más apropiado sea para el usuario. En el módulo propuesto este patrón se observa, además de en la clase descrita anteriormente; en la clase "*CommandManager"*, utilizada para seleccionar qué estrategia de dibujo emplear luego de que fue realizada la transformación, en dependencia del tipo de entidad que se vaya a dibujar. Una mayor explicación de cómo se evidencia la aplicación de este patrón en la solución se puede encontrar en los sub-epígrafes 3.2.2 y 3.2.5.

Por su parte el patrón "*State*" permite a un objeto alterar su comportamiento cuando su estado interno cambia. Es recomendable su utilización cuando el comportamiento del objeto depende de su estado, y debe cambiar el comportamiento en tiempo de ejecución de acuerdo a ese estado. En el caso específico de su utilización en el presente modulo, se utilizó para definir los comportamientos en los diferentes estados que puede tener el evento click del mouse, en dependencia del lugar dentro del *"sketcher"* donde sea pulsado. Se introdujo así una clase abstracta llamada "*StateTransform*" (epígrafe 3.2.4*),* para representar los estados de la selección del usuario con el mouse. Esta clase declara una interfaz común a todas las clases que representan los diferentes estados operacionales, que son sus clases hijas y que en este caso son los estados de selección del primer punto (centro de rotación, centro de escalado o inicio de la traslación según sea el caso), selección del segundo punto de la traslación, selección de la estructura a transformar, y selección de una sola arista en el caso de que se haya seleccionado por parte del usuario la selección simple.

## 3.2 Descripción de las clases del diseño.

### 3.2.1 Clase Transformation_Manager

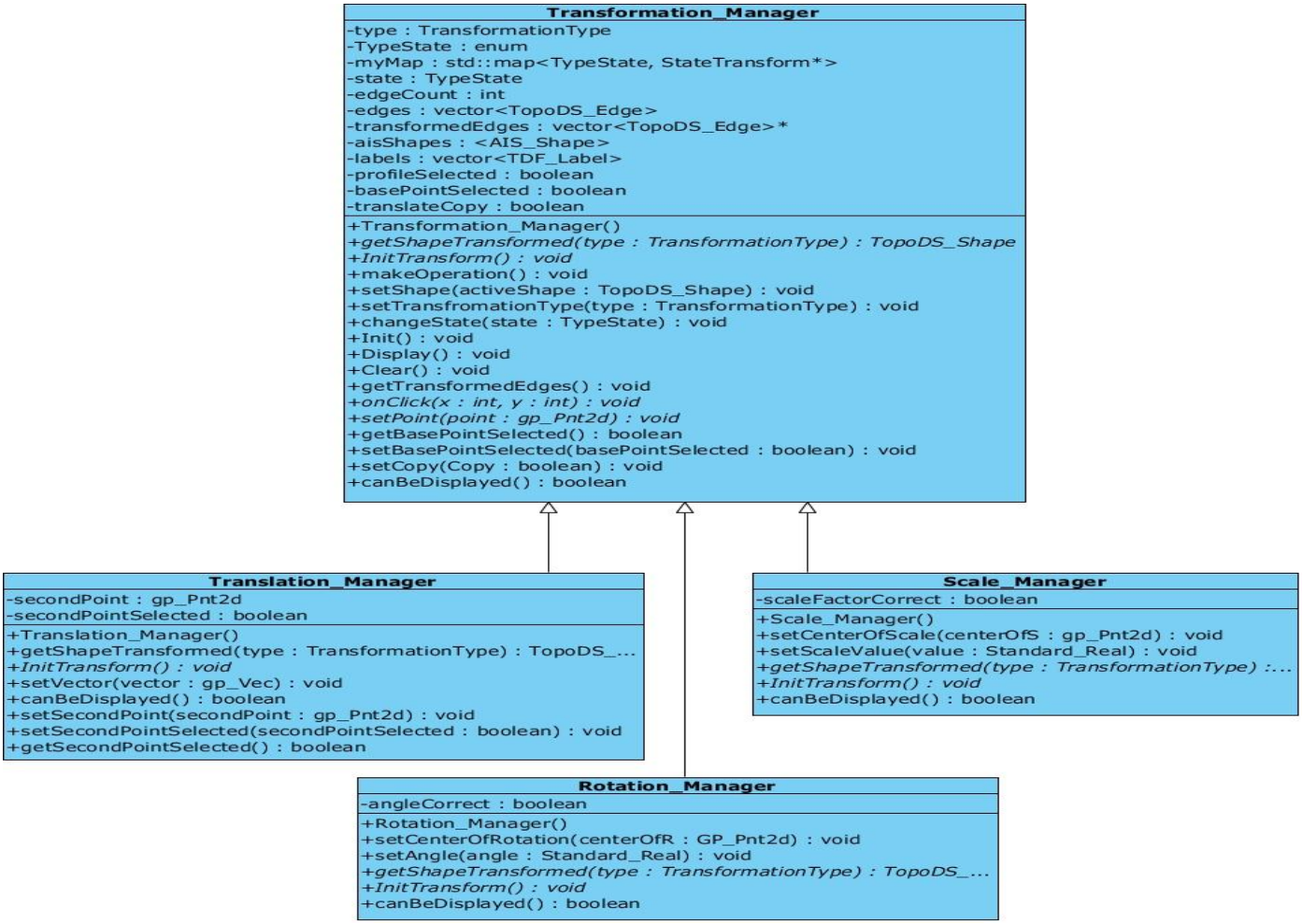


**Figura 4 Clase Transformation Manager**

Esta clase, padre de 3 clases Manager (*Translation_Manager, Rotation_Manager y Scale_Manager*) específicas para cada tipo de transformación, es la encargada de, con los datos que le llegan desde la interfaz a través de clases mediadoras (*actions*), instanciar la entidad a transformar, y luego aplicar los procedimientos necesarios para llevar a cabo la transformación. La misma, luego de la selección por parte

del usuario de la transformación a aplicar, hace uso de la clase *TransformStrategy*, que siguiendo el procedimiento explicado en el epígrafe 3.2.2, inicializa el objeto correspondiente a la transformación a aplicar, para luego de creado el mismo, ejecutar la transformación utilizando los atributos previamente capturados en la interfaz. Luego de ejecutada la transformación hará uso de la clase *CommandExecutor* para dibujar en el *"sketcher"* del modelador geométrico el objeto en sus nuevas coordenadas. Sus principales atributos son:

- **type:** un objeto cuyo tipo de dato viene dado por el enum de la clase *Transform_Strategy,* que tomará su valor de acuerdo al tipo de transformación seleccionada.
- **myMap:** mapa que contiene las llamadas a las diferentes constructores de las clases que implementan los estados específicos (clase *stateTransform*) de acuerdo al valor tomado por el enum (*TypeState*) que sirve como llave del mapa, y que cambia en dependencia del estado en que se encuentre el sistema, y en dependencia de lo que este espere.
- **edgeCount:** contador de la cantidad de aristas que han sido seleccionadas y almacenadas en el vector *edges* para ser transformadas.
- **edges, transformedEdges**, **aisShapes y labels**: vectores que almacenan respectivamente las aristas seleccionadas para transformar, las aristas una vez transformadas, las aristas (luego de convertidas al tipo *AIS_Shape* de OpenCASCADE, que es el tipo que permite que sean visualizadas) que serán dibujadas, y sus *labels* o referencias a su posición en el árbol de OCAF.
- **Atributos booleanos** que toman valor "true" a medida que sean seleccionados los parámetros necesarios para llevar a cabo la transformación que cada uno representa, y que son chequeados (true) antes de proceder a dibujar, en caso contrario (false) el sistema no aplica la transformación

### 3.2.2 Clase TransformStrategy

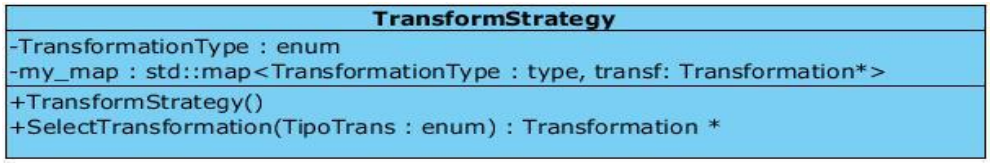


**Figura 5 Clase Transform Strategy**

Esta clase es la base del patrón de diseño "*Strategy*" utilizado. La misma contiene una serie de llamadas a constructores de transformaciones específicas que serán seleccionados indistintamente con la estrategia adecuada, en dependencia de la selección hecha por el cliente. Contiene un "*map*" de la biblioteca std (standard template library) de C++. Dichos pares que conforman cada posición del "*map*" están formados por un enum que identifica el tipo de transformación; y que será recibido desde la clase *Transformation_Manager* luego de realizada la selección por parte del cliente de la transformación a realizar, y una instancia de la clase que modela la estrategia específica correspondiente a ese tipo de transformación, y que será enviada a la clase *Transformation_Manager* para que esta se encargue de ejecutar la transformación.

### 3.2.3 Clase Transformation

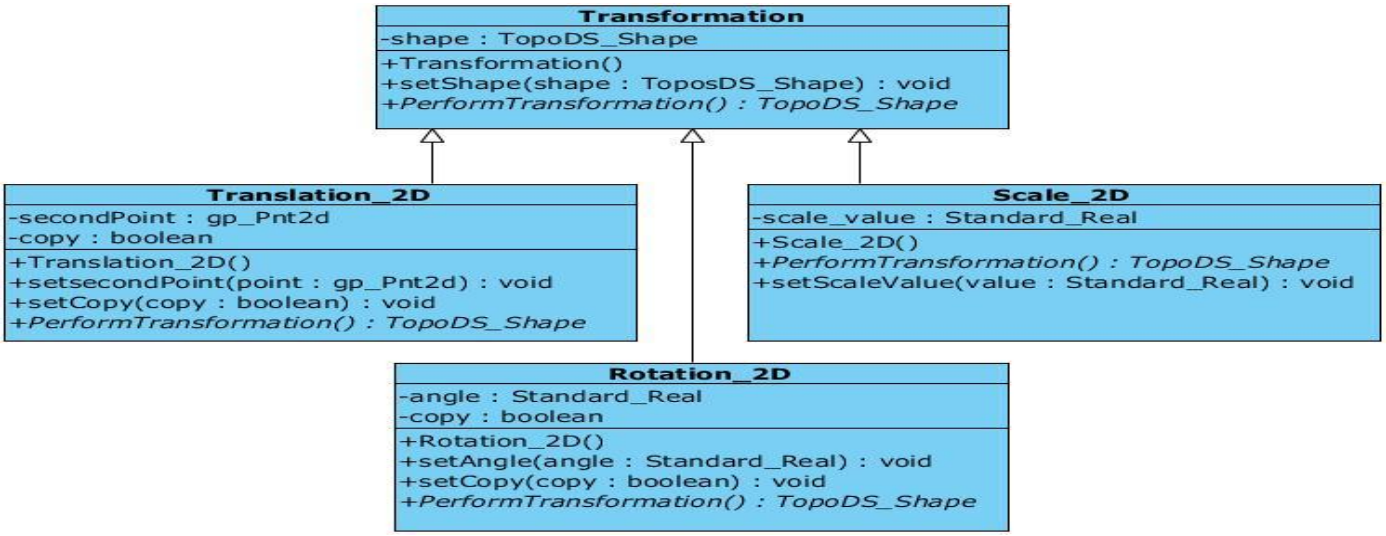


**Figura 6 Clase Transformation**

Es la clase padre (Estrategia Abstracta) de las transformaciones específicas (Estrategias Concretas). Contiene solamente un constructor vacío, y el alma de su existencia que es un método virtual redefinido en las clases hijas llamado "*performTransformation()*", el cual es realmente el encargado de aplicar la transformación, haciendo uso de las funcionalidades de OpenCASCADE. Este método será invocado

desde la clase *Transformation_Manager* (donde finalmente se llevará a cabo la transformación), por la instancia devuelta por la clase *TransformStrategy*.

### 3.2.4 Clase StateTransform

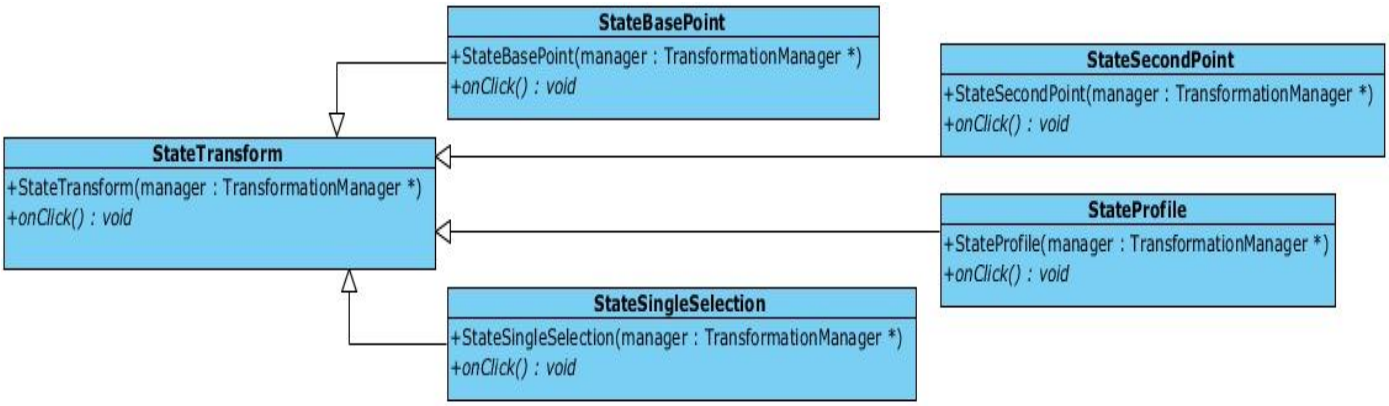


**Figura 7 Clase State Transform**

Esta es la clase que pone en práctica el patrón "*State*" descrito anteriormente. En su método virtual puro "*onClick()"* se definen las operaciones específicas que se han de hacer con los eventos del mouse de acuerdo al estado en que se encuentre la aplicación en tiempo de ejecución. Este estado irá cambiando en la clase *Transformation_Manager* en dependencia de lo que se espere en cada momento, y tras la llamada a su método "*onClick()"* para cada estado específico, se ejecutarán las acciones necesarias para capturar la forma (clase *StateProfile*), el punto inicial de la traslación, rotación (eje de rotación) o escalado (centro de escalado) (clase *StateBasePoint*), el punto final de la traslación (clase *StateSecondPoint*), o la selección de una única arista si así lo decide el usuario (clase *StateSingleSelection*). Dichos métodos "*onClick()"* hacen uso de comandos de OCAF ya implementados en la herramienta GALBA-CAD para capturar los puntos o entidades necesarios.

### 3.2.5 Clase CommandManager.


**Figura 8 Clase Command Manager**

El funcionamiento de esta clase es muy similar al de la clase *"TransformStrategy".* Contiene una serie de llamadas a constructores de clases "CommandExecutor" que se encargan de trabajar con el árbol de OCAF y ejecutar los comandos necesarios para dibujar en el *"sketcher"* las entidades resultantes de aplicar las transformaciones, constructores que serán seleccionados indistintamente según la estrategia adecuada, en dependencia del tipo de entidad que se desea dibujar (línea, círculo o arco de circunferencia), conociendo este tipo de entidad de acuerdo al valor que se especifique del enum "*shapetype*". Contiene un "map" cuyos pares están conformados como llave por el valor del enum previamente descrito que identifica el tipo de entidad a dibujar, y que será recibido desde la clase *Transformation_Manager* luego de ejecutada la transformación, y una instancia de la clase que ejecuta el comando de dibujo específico correspondiente a ese tipo de entidad.

### 3.2.6 Clase CommandExecutor.

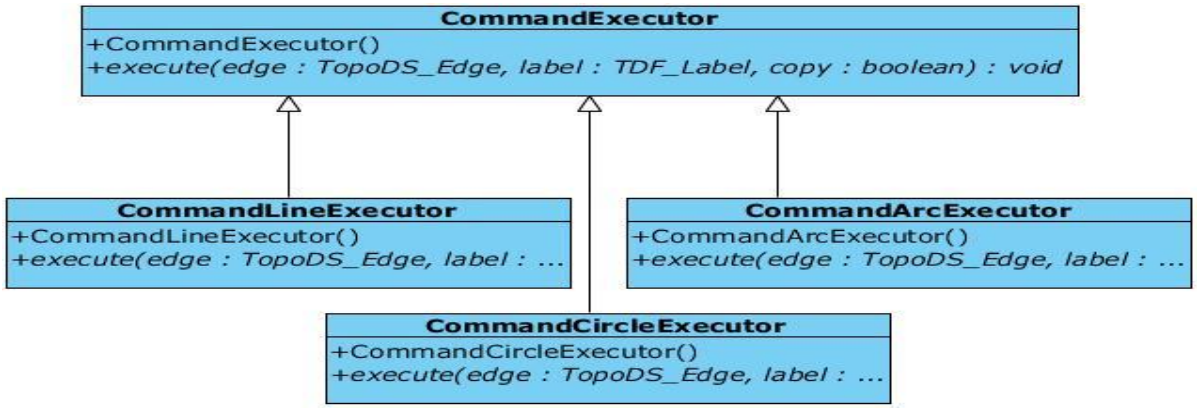


**Figura 9 Clase Command Executor**

Es la clase padre (Estrategia Abstracta) de las clases que contiene los comandos específicos de dibujo (Estrategias Concretas). Contiene solamente un constructor vacío, y el método virtual puro "***execute()",*** que es el encargado de dibujar las entidades específicas según sea la clase, haciendo uso del árbol de OCAF. Utiliza internamente, para llevar a cabo la tarea de eliminar la entidad antigua (en caso de que no se haya seleccionado la opción "Copia"), y dibujar la nueva entidad luego de aplicada la transformación; comandos y llamadas a drivers que ya se encuentran implementados en otros módulos de la herramienta GALBA-CAD. El método "*execute()",* será invocado desde la clase *Transformation_Manager* por la instancia devuelta por la clase *CommandManager* luego de seleccionada la estrategia de dibujo a seguir. Como se expresó anteriormente, otras clases pertenecientes a OpenCASCADE son utilizadas para llevar

a cabo las transformaciones dentro del método "*performTransformation()*", mencionado en el sub-epígrafe anterior Estas clases y sus principales métodos que fueron utilizados son descritas a continuación

## 3.3 Descripción de las clases pertenecientes a OpenCASCADE utilizadas en la solución.

Para llevar a cabo la solución que se propone, tal y como se ha dicho, se hizo uso de la biblioteca OpenCASCADE, y de ella fueron utilizadas varias clases cuya funcionalidad y métodos utilizados se describen a continuación:

### 3.3.1 Clase gp_Trsf 2d

Esta clase define una transformación no persistente en el espacio 2D. Las siguientes transformaciones se encuentran implementadas:

. Traslación, rotación, escalado
. Simetría con respecto a un punto, una línea y un plano.

**Métodos utilizados pertenecientes a la clase gp_Trsf2d:**

- *void* ***SetRotation*** *(const gp_Pnt2d &P, const Standard_Real Ang)*

Cambia la transformación en una rotación. P es el punto que será el centro de rotación y Ang es el valor angular de rotación dado en radianes.

- *void* ***SetScale*** *(const gp_Pnt2d &P, const Standard_Real S)*

Cambia la transformación en un escalado. P es el centro del mismo y S es el valor de escalado.

- *void* ***SetTranslation*** *(const gp_Pnt2d &P1, const gp_Pnt2d &P2)*

Convierte la transformación en una traslación, a partir de un punto P1 a otro P2.

### 3.3.2 Clase BRepBuilderAPI_Transform

La clase *BRepBuilderAPI_Transform* se utiliza para aplicar una transformación a una forma u objeto. Los métodos tienen un argumento booleano para copiar o reubicar el objeto original, siempre que la transformación lo permita (sólo es posible para las transformaciones directas e isométricas, que son aquellas cuyo determinante de su parte vectorial es igual a 1). Por defecto, la forma original es reubicada en las nuevas coordenadas luego de aplicada la transformación.

La transformación a aplicar se define como una transformación de tipo *gp_Trsf2d* y se puede aplicar a todas las curvas que soportan los bordes de un objeto y a todas las superficies que soportan las caras del mismo.

Un objeto de esta clase proporciona un marco para definir la transformación geométrica que debe aplicarse, la aplicación del algoritmo de transformación, y consultar los resultados.

**Métodos utilizados pertenecientes a la clase *BRepBuilderAPI_Transform*:**

- ***BRepBuilderAPI_Transform*** *(const gp_Trsf2d &T)*

Constructor de la clase que crea un marco para aplicar la transformación geométrica T a un objeto. Luego se utiliza la función "*Perform*" para definir el objeto a transformar.

- ***BRepBuilderAPI_Transform*** *(const TopoDS_Shape &S, const gp_Trsf2d &T, const Standard_Boolean Copy=Standard_False)*

Otro constructor de la clase que recibe una transformación del tipo gp_Trsf2d<T>, y la aplica a la entidad "S". Si la transformación es directa e isométrica y "*Copy"* = Standard_False, la entidad resultante la misma "S" recibida como parámetro de entrada, pero creada en una nueva ubicación. De lo contrario, la transformación se aplica en un duplicado de "S".

- *void* ***Perform*** *(const TopoDS_Shape &S, const Standard_Boolean Copy = Standard_False)*

Se aplica la transformación geométrica definida en el momento de la construcción a la entidad S, si se utilizó el primero de los constructores antes mencionados. Si la transformación T es directa e isométrica, y si "*Copy*" = false (el valor por defecto), el objeto resultante de la transformación es el mismo que el original, pero con una nueva ubicación asignada. En todos los demás casos, la transformación se aplica a un duplicado de S. Se debe utilizar el método "*Shape()"* para acceder al resultado. Una misma instancia de esta clase se puede reutilizar para aplicar la misma transformación geométrica a otros objetos. Sólo es necesario especificar este nuevo objeto llamando nuevamente a la función "*Perform()".*

## 3.4 Diagramas de Secuencia del Diseño

Los diagramas de secuencia son diagramas que se usan para modelar la interacción entre objetos en un sistema a través del tiempo y se modela para cada caso de uso. [19]

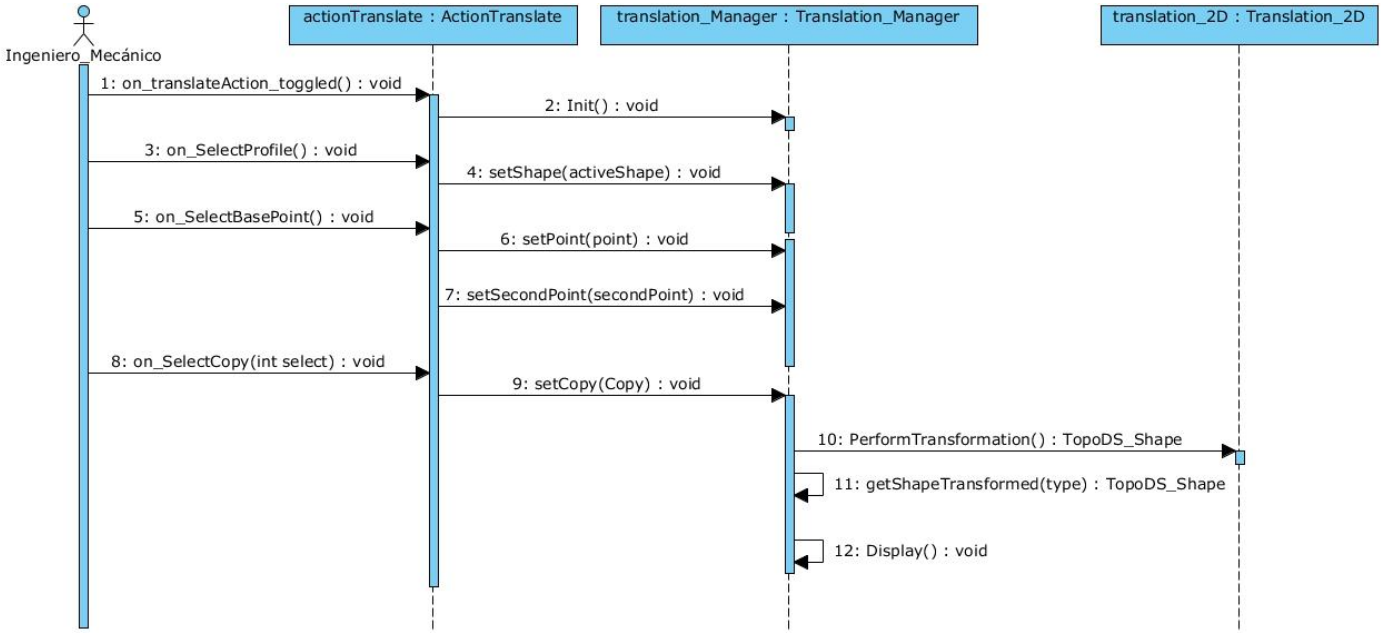


**Figura 10 Diagrama de Secuencia del Caso de Uso "Aplicar_Traslacion".**

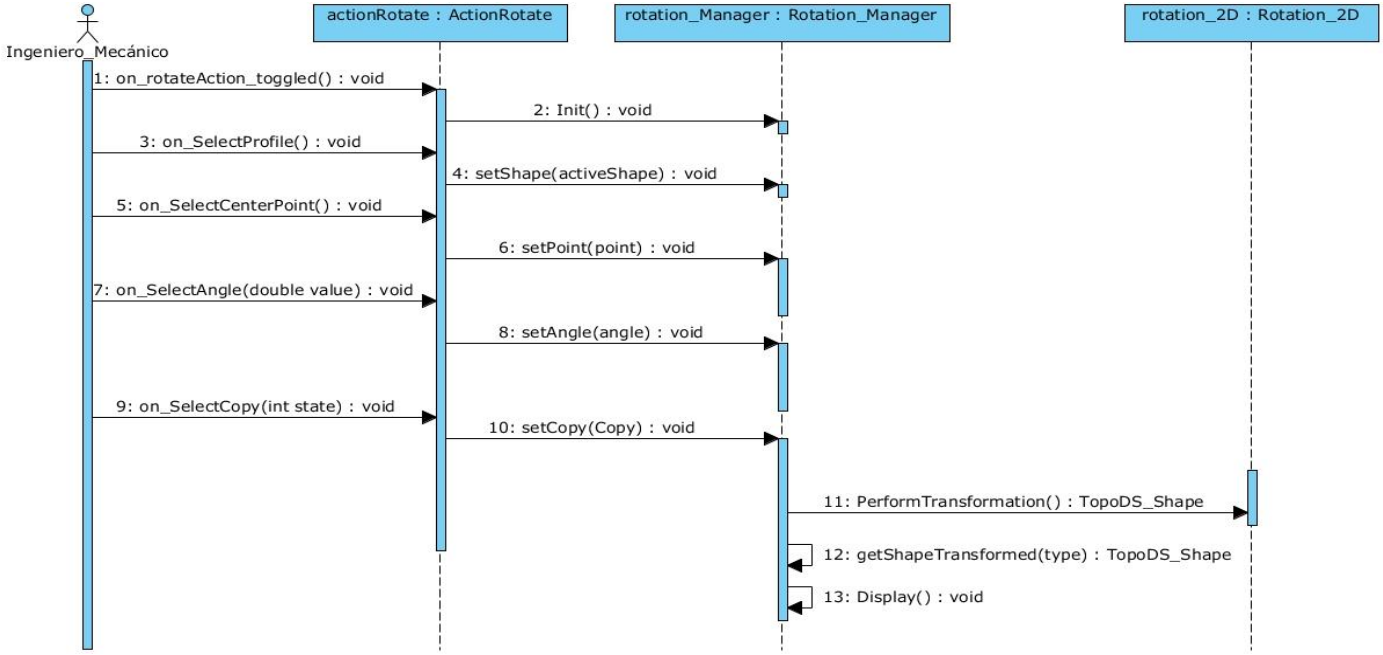


**Figura 11 Diagrama de Secuencia del Caso de Uso "Aplicar_Rotacion".**

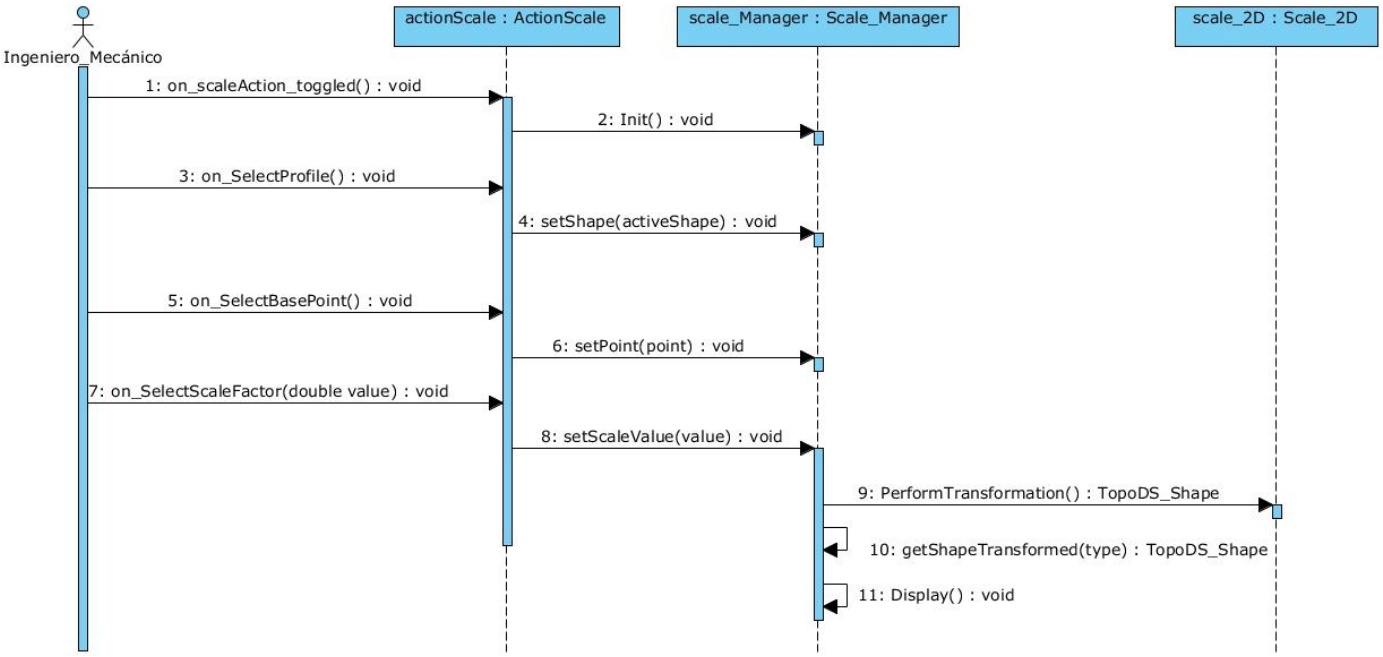


**Figura 12 Diagrama de Secuencia del Caso de Uso "Aplicar_Escalado".**

## 3.5 Diagrama de Componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software, sean éstos componentes de código fuente, binarios o ejecutables. [26]

En el diagrama brindado a continuación se evidencia la interacción entre el componente "qoce-library" y el componente *"sketcher"*, así como las interacciones entre las diferentes clases en su interior, a la vez que ambos interactúan con la biblioteca OpenCASCADE.

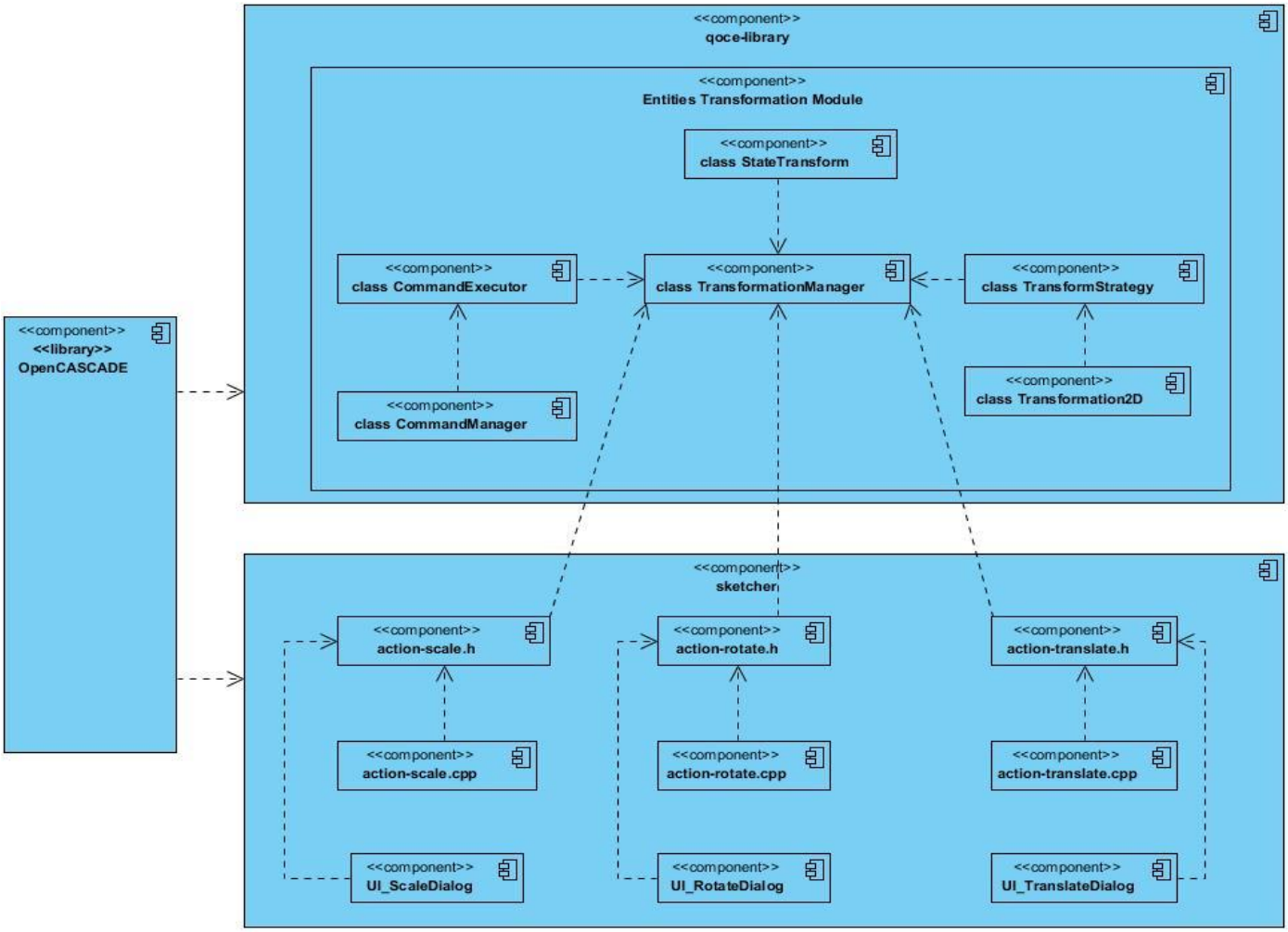


**Figura 13 Diagrama de componentes del módulo de transformaciones a entidades 2D**

## 3.6 Algoritmos de transformación utilizando las clases pertenecientes a OpenCASCADE y sus funcionalidades.

### 3.6.1 Ejemplo de rotación.

```
TopoDS_Shape myShape1 = ...;
// El objeto 1 original
TopoDS_Shape myShape2 = ...;
// El objeto 2 original
gp_Trsf2d T;
gp_Pnt2d CofR = gp_Pnt2d(0,0);
T.SetRotation(CofR, 30*PI/180); // rotación de 30 grados en el punto “CofR”
BRepBuilderAPI_Transformation theTrsf(T);
theTrsf.Perform(myShape1);
TopoDS_Shape myNewShape1 = theTrsf.Shape()
theTrsf.Perform(myShape2,Standard_True); // Aquí se fuerza la duplicación al poner el atributo en “true”
TopoDS_Shape myNewShape2 = theTrsf.Shape();
```

### 3.6.2 Ejemplo de escalado.

```
TopoDS_Shape S = BRepPrimAPI_MakeWedge(60.,100.,80.,20.); //objeto
gp_Trsf2d theTransformation;
gp_Pnt2d theCenterOfScale(200,60,60);
theTransformation.SetScale(theCenterOfScale,0.5); // Escalado con valor 0.5
BRepBuilderAPI_Transform myBRepTransformation(S,theTransformation);
//otra variante sin usar la función “Perform”, en este caso siempre se duplica el objeto.
TopoDS_Shape TransformedShape = myBRepTransformation.Shape();
```

### 3.6.3 Ejemplo de traslación

```
TopoDS_Shape S = BRepPrimAPI_MakeWedge(6.,10.,8.,2.); //objeto
gp_Trsf2d theTransformation;
gp_Vec2d theVectorOfTranslation(6,6,6);
theTransformation.SetTranslation(theVectorOfTranslation);
BRepBuilderAPI_Transform myBRepTransformation(S,theTransformation);
TopoDS_Shape TransformedShape = myBRepTransformation.Shape();
```

## 3.7 Conclusiones del capítulo:

En este capítulo se da continuidad al proceso de desarrollo de software definido por RUP, y se describen las etapas de Diseño e Implementación. Se presentó la arquitectura propuesta para el módulo, de modo que se ajustara a la arquitectura base de la herramienta GALBA-CAD, y cumpliera con los estándares de diseño que definen los patrones GRASP “Bajo Acoplamiento” y “Alta Cohesión”. Se brinda la propuesta de solución al problema planteado, que reside en la reutilización e integración de los algoritmos implementados por OpenCASCADE para la transformación de entidades geométricas 2D, ajustándolos al diseño de la herramienta privativa Autodesk® Inventor®, y a las características del *“sketcher”* del modelador geométrico de la herramienta GALBA-CAD. Se describen los procedimientos seguidos para la utilización de dichas clases y algoritmos dentro del diagrama de clases del diseño propuesto, así como los patrones de comportamiento empleados para garantizar una propuesta de solución elegante y profesional.

# CAPÍTULO 4: PRUEBAS AL SISTEMA

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas a los software es un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto. Probar es un proceso de ejecución de un programa con la intención de descubrir un error. Una prueba tiene éxito si se descubre un error no detectado hasta entonces. [27]

RUP define en su Flujo de Trabajo de Pruebas 4 niveles de prueba por los que debe pasar un software: Nivel de Unidad, Nivel de Integración, Nivel de Sistema y Nivel de Aceptación. De ellos el que concierne al trabajo realizado y se vuelve inmediatamente necesario es el nivel de unidad, utilizado para verificar todos los flujos de control. El tipo de prueba más adecuado para este nivel es la prueba de caja negra. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software; o sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

En estas pruebas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Las pruebas de caja negra se realizan mediante casos de prueba. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los mismos, y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa, en dependencia de si se desea hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos de cómo son llevados a cabo los diferentes procedimientos por parte del sistema

## 4.1 Casos de Prueba

Los casos de prueba son un conjunto de condiciones o variables mediante los cuales se determina si los requisitos de una aplicación son parcial o completamente satisfechos. Un caso de prueba consta de una entrada conocida y una salida esperada, estos son formulados antes de que se ejecute la prueba, donde la entrada conocida debe probar una precondición y la salida esperada debe probar una postcondición. Los valores asignados a las variables significan V (aparece y es válida), y NA (No aparece la variable)

| Escenario | Descripción | Entidad(es) a trasladar | Punto inicial | Punto final | Respuesta esperada (según resultado en Autodesk® Inventor®) | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|---|---|
| *EC 1.1 Introducir las variables correctamente* | *El usuario introduce de forma correcta todas las variables necesarias* | *V* | *V* | *V* | *El sistema traslada la entidad desde el punto inicial hasta el final* | *Esperada* | *Ícono "Move"/Selección de Entidad/Selección de punto inicial/Selección de punto final/Botón "Done"* |
| *EC 1.2 Ausencia de alguna de las variables* | *Falta alguna de las variables necesarias* | *V* | *V* | *NA* | *El sistema no ejecuta ninguna acción al presionarse el botón "Done"* | *Esperada* | *Ícono "Move"/Selección de solo algunas de las variables necesarias/Botón "Done"* |
| | | *NA* | *V* | *V* | | | |
| | | *V* | *NA* | *NA* | | | |
| | | *NA* | *V* | *NA* | | | |
| | | *NA* | *NA* | *NA* | | | |

**Tabla 6 Caso de prueba para el caso de uso "Aplicar Traslación"**

| Escenario | Descripción | Entidad(es) a rotar | Punto eje de rotación | Ángulo | Respuesta esperada (según resultado en Autodesk® Inventor®) | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|---|---|
| EC 1.1 Introducir las variables correctamente | El usuario introduce de forma correcta todas las variables necesarias | V | V | V | El sistema rota la entidad de acuerdo al eje de rotación y el ángulo seleccionados. | Esperada | Ícono "*Rotate*"/Selección de Entidad/Selección de punto eje/Selección ángulo deseado/Botón "Done" |
| EC 1.2 Ausencia de alguna de las variables | Falta alguna de las variables necesarias | V | V | NA | El sistema no ejecuta ninguna acción al presionarse el botón "Done" | Esperada | Ícono "*Rotate*"/Selección de solo algunas de las variables necesarias/Botón "Done" |
| | | NA | V | V | | | |
| | | V | NA | NA | | | |
| | | V | NA | V | | | |
| | | NA | NA | NA | | | |
| | | NA | V | NA | | | |
| | | NA | NA | V | | | |

**Tabla 7 Caso de prueba para el caso de uso "Aplicar Rotación"**

| Escenario | Descripción | Entidad(es) a escalar | Punto base | Factor de escalado | Respuesta esperada (según resultado en Autodesk® Inventor®) | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|---|---|
| EC 1.1 Introducir las variables correctamente | El usuario introduce de forma correcta todas las variables necesarias | V | V | V | El sistema escala la entidad desde el punto establecido como base y según el factor de escalado introducido | Esperada | Ícono "*Scale*"/Selección de Entidad/Selección de punto eje/Selección ángulo deseado/Botón "Done" |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EC 1.2 Ausencia de alguna de las variables | Falta alguna de las variables necesarias | V | V | NA | El sistema no ejecuta ninguna acción al presionarse el botón “Done” | Esperada | Ícono “*Scale*”/Selección de solo algunas de las variables necesarias/Botón “Done” |
| | | NA | V | V | | | |
| | | V | NA | NA | | | |
| | | V | NA | V | | | |
| | | NA | NA | NA | | | |
| | | NA | V | NA | | | |
| | | NA | NA | V | | | |

**Tabla 8 Caso de prueba para el caso de uso “Aplicar Escalado”**

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Entidad | Se selecciona con eventos del mouse dentro del *“sketcher”* de la herramienta, utilizando la tecla *shift* para selecciones múltiples | No | Al seleccionar con el click la(s) entidad(es) que se desea trasladar, el sistema la(s) captura en formato “*TopoDS_Edge”* de OpenCASCADE. Al ser una variable capturada por el sistema mediante un click, nunca aparece con la opción I (Inválido) en los casos de prueba |
| 2 | Punto de inicio de la traslación | Se selecciona con eventos del mouse, mediante click en el *“sketcher”* de la herramienta | No | Al ser seleccionado mediante click el lugar deseado para dar inicio a la translación, el sistema captura las coordenadas de dicho punto y las convierte y almacena con el formato gp_Pnt2d de OpenCASCADE. Al ser una variable capturada por el sistema mediante un click, nunca aparece con la opción I (Inválido) en los casos de prueba |
| 3 | Punto final de la traslación | Se selecciona con eventos del mouse mediante click en el *“sketcher”* de la herramienta | No | Al ser seleccionado mediante click el lugar deseado como final de la translación, el sistema captura las coordenadas de dicho punto y las convierte y almacena con el formato gp_Pnt2d de OpenCASCADE Solo puede ser seleccionado una vez seleccionado el punto inicial de la traslación, de lo contrario el sistema se mantiene a la espera del primero. Al ser una variable |

| | | | | |
|---|---|---|---|---|
| | | | | capturada por el sistema mediante un click, nunca aparece con la opción I en los casos de prueba |
| 4 | Punto eje de rotación | Se selecciona con eventos del mouse mediante click en el *"sketcher"* de la herramienta | No | Al ser seleccionado mediante click el lugar deseado como eje de rotación el sistema captura las coordenadas de dicho punto y las convierte y almacena con el formato gp_Pnt2d de OpenCASCADE. Al ser una variable capturada por el sistema mediante un click, nunca aparece con la opción I en los casos de prueba |
| 5 | Angulo | QSpinBox | Si | El valor debe ser un número real entre 0 y 360. El sistema garantiza que el valor no se encuentre fuera de ese rango ajustando las propiedades del SpinBox, por lo que nunca aparece con la opción I (Inválido) en los casos de prueba. Puede ser nulo pues si se encuentra en cero la rotación se podrá ejecutar, lo que no tendrá efecto al tener ángulo cero. |
| 6 | Punto Base | Se selecciona con eventos del mouse mediante click en el *"sketcher"* de la herramienta | No | Al ser seleccionado mediante click el lugar deseado como punto base del escalado el sistema captura las coordenadas de dicho punto y las convierte y almacena con el formato gp_Pnt2d de OpenCASCADE. Al ser una variable capturada por el sistema mediante un click, nunca aparece con la opción I en los casos de prueba |
| 7 | Factor de escalado | QDoubleSpinBox | Si | El valor debe ser un número real. El sistema garantiza que el valor no sea de otro tipo de dato al utilizar un QDoubleSpinBox que solo admite esos valores, por lo que nunca aparece con la opción I (Inválido) en los casos de prueba. Puede ser nulo pues si se encuentra en cero el escalado se podrá ejecutar, lo que no tendrá efecto al tener factor cero. |

**Tabla 9 Descripción de las variables**

## 4.2 Conclusiones del capítulo:

Como resultados de las pruebas de unidad, específicamente de caja negra no se obtuvieron no conformidades, por los que se pueden clasificar como positivas.

De esta forma concluye la etapa de pruebas definida por RUP, quedando solamente como fase a realizar el despliegue del sistema, el cual se llevará a cabo con el despliegue final de la herramienta GALBA-CAD.

# CONCLUSIONES:

Con la realización de este trabajo de diploma, se desarrolló un módulo que permite la aplicación de transformaciones geométricas a los elementos 2D presentes dentro del *"sketcher"* del modelador geométrico de la herramienta GALBA-CAD, que fue implementado directamente sobre el código fuente de la misma y dio solución a la problemática planteada.

Con la creación de este módulo, el usuario final cuenta con la posibilidad de poder modificar la orientación y posición de los diferentes objetos 2D, lo que le permite poder combinarlos para dar lugar a diseños más complejos de piezas mecánicas, con lo que queda evidenciada la contribución hecha por el presente trabajo a la herramienta. De esta forma se vio cumplido el objetivo propuesto al comienzo de la investigación.

Además, se creó una arquitectura flexible, profundizando en las ventajas que ofrecen los patrones GRASP y de comportamiento para la obtención de un producto, elegante y de calidad, previendo la introducción futura de nuevas transformaciones sin que estas afecten el funcionamiento del sistema en general.

# RECOMENDACIONES:

Se recomienda luego de desarrollado el módulo:

- Adicionarle para el despliegue final de GALBA-CAD el trabajo con restricciones geométricas, del mismo modo que lo hace la herramienta Autodesk® Inventor®, a la hora de seleccionar para ser transformadas entidades que contengan restricciones geométricas con otras.

- Incluirle en un futuro nuevos tipos de transformaciones geométricas que puedan ser de utilidad como la transformación "*mirror",* el escalado diferencial, y nuevas transformaciones resultado de combinar alguna de las ya conocidas.

# REFERENCIAS BIBLIOGRÁFICAS:


1. Scribd. [En línea] [Citado el: 10 de Enero de 2012.] http://es.scribd.com/doc/17754860/Sistema-CAD.
2. TeleWork Spain. [En línea] [Citado el: 10 de Enero de 2012.] http://www.teleworkspain.com/Art019.htm..
3. **Costa, José Antonio Velásquez.** *Computer Integrated Manufacturing CIM.* Lima, Perú : Universidad Ricardo Palma, 2009.
4. Open Cascade. [En línea] [Citado el: 12 de Enero de 2012.] http://www.opencascade.org/..
5. **Alvaro Martínez, Andrés Pastorin.** Taller de Estudio e Implementación de Videojuegos 2D. *Taller de Estudio e Implementación de Videojuegos 2D.* Montevideo : s.n., 2011.
6. **Sierra, C. Álvarez de Zayas y V.** La investigación científica en la sociedad del conocmimiento. La Habana, Cuba : s.n., 2002. Vol. Material de Apoyo a la docencia.
7. Rincón del Vago. [En línea] [Citado el: 17 de Enero de 2012.] http://html.rincondelvago.com/sistemas-cadcamcae.html.
8. Kalipedia. [En línea] [Citado el: 20 de Enero de 2012.] http://www.kalipedia.com/tecnologia/tema/dibujo/sistemas-cae-i-computer.html?x=20070822klpingtcn_169.Kes&ap=4.
9. AutoCAD cumple 30 años en el mercado. *Plataforma Arquitectura.* [En línea] 4 de Febrero de 2012. [Citado el: 12 de Marzo de 2012.] http://www.plataformaarquitectura.cl/2012/02/04/autocad-cumple-30-anos-en-el-mercado/.
10. **Ashampoo.** Ashampoo.com. [En línea] [Citado el: 18 de Enero de 2012.] http://www.ashampoo.com/es/eur/pin/0460/CAD_Construccion/Ashampoo-3D-CAD-Architecture-3.
11. Inventor - Software CAD 3D y de Diseño Mecánico 3D. *Autodesk.* [En línea] autodesk. [Citado el: 12 de Abril de 2012.] http://latinoamerica.autodesk.com/adsk/servlet/pc/index?id=14601337&siteID=7411870.
12. IdeasWeb. [En línea] [Citado el: 19 de Enero de 2012.] http://ideasweb.info/blog/?p=106.
13. Proyecto pingüino. [En línea] [Citado el: 19 de Enero de 2012.] http://proyectopinguino.blogspot.com/2008/08/alternativas-autocad-para-linux.html.
14. GenBeta. [En línea] 13 de Enero de 2012. [Citado el: 19 de enero de 2012.] http://www.genbeta.com/herramientas/librecad-un-programa-sencillo-para-iniciarse-en-el-mundo-del-cad.
15. VTK. [En línea] Kitware. [Citado el: 25 de Mayo de 2012.] http://www.vtk.org/.
16. Mixprogramas. [En línea] [Citado el: 21 de Enero de 2012.] www.mixprogramas.com/open-cascade-6-5-1/.

17. Transformaciones geométricas bidimensionales. [aut. libro] Carlos Balderrama Vásquez. *Computación gráfica.*
18. Scribd. [En línea] 9 de Junio de 2011. [Citado el: 16 de Febrero de 2012.] http://es.scribd.com/doc/64077925/Metodologias-de-Desarrollo-de-Software.
19. **IBM Corporation.** *Rational Unified Process.* 2006.
20. **Fowler, Martin y Scott, Kendall.** *UML gota a gota.* 1999.
21. Monografías.com. [En línea] [Citado el: 10 de Abril de 2012.] http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml.
22. Eclipse Platform Technical Overview. s.l. : Object Technology International, Inc. , 2003.
23. Ecured. [En línea] [Citado el: 10 de Abril de 2012.] http://www.ecured.cu/index.php/Qt.
24. Ecured, conocimiento con todos y para todos. [En línea] [Citado el: 14 de Marzo de 2012.] http://www.ecured.cu/index.php/Patrones_de_Asignaci%C3%B3n_de_Responsabilidades.
25. **Larman, Craig.** *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN: 970-17-0261-1 .
26. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico.* s.l. : Mc Graw Hill, 2001.
27. **Myers, Glen.** *The Art of Software Testing.* 1979. ISBN-10: 0471078786.

# ANEXOS

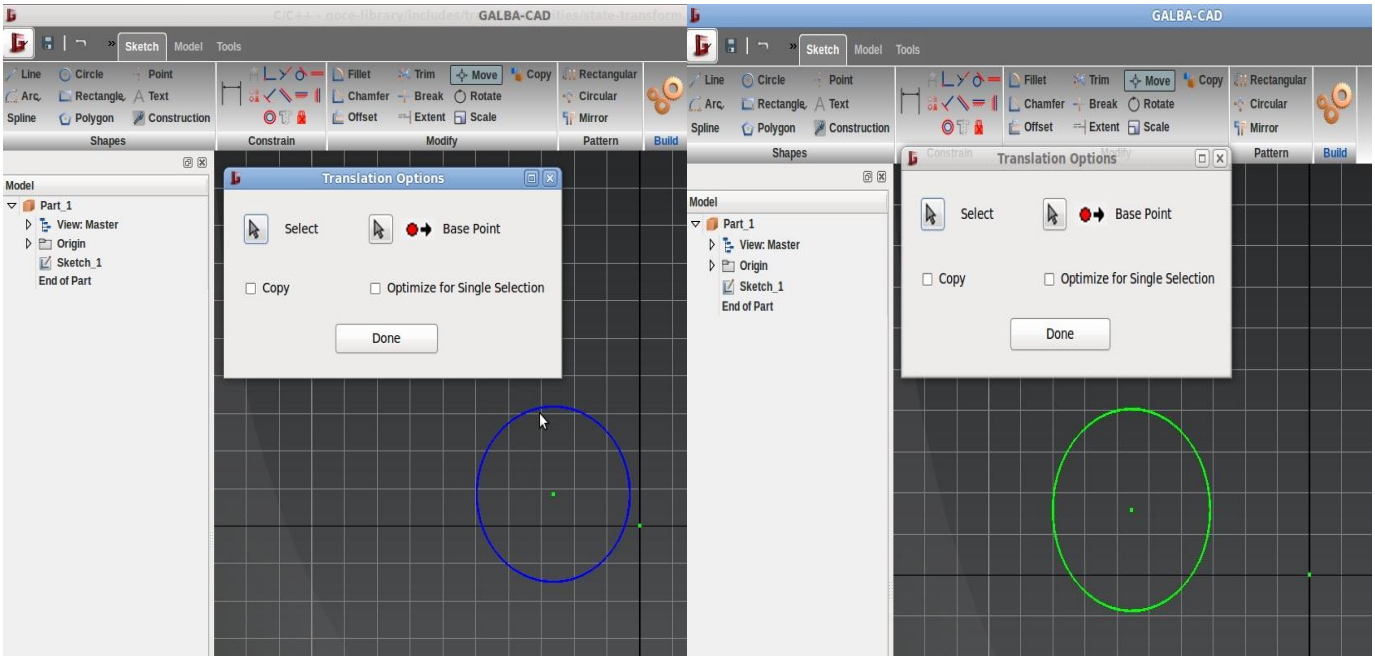


**Figura 14 Ejemplo de traslación de un círculo en la aplicación**

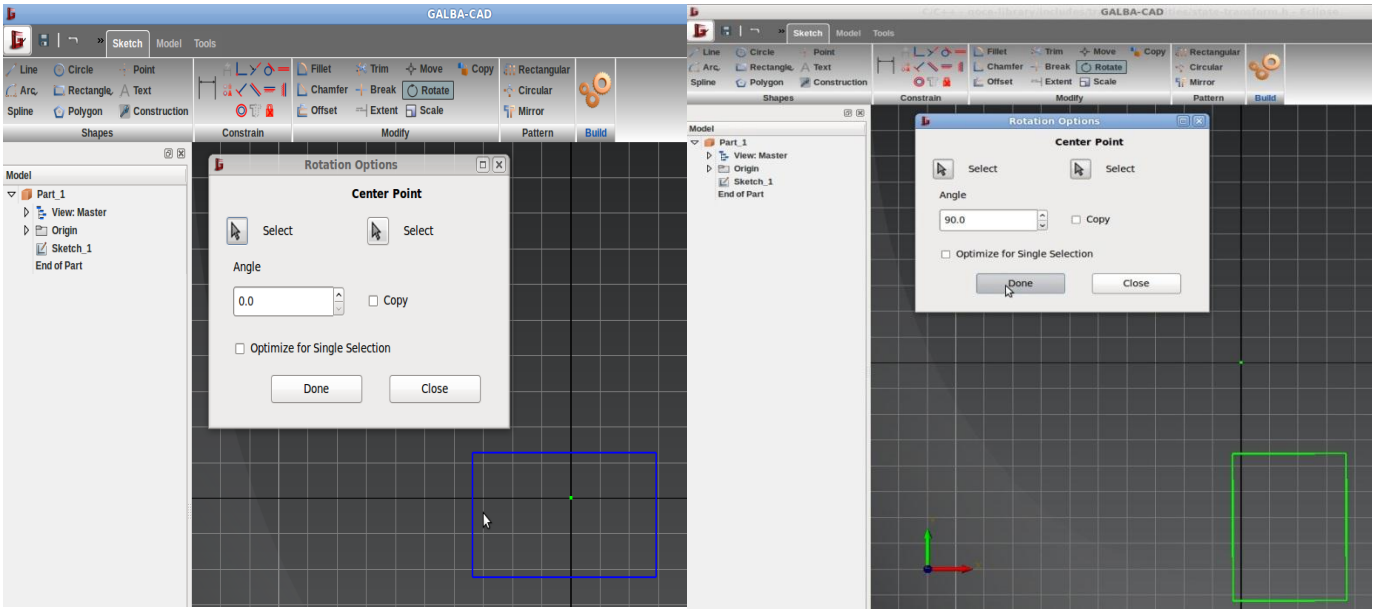

**Figura 15 Ejemplo de rotación de un rectángulo en la aplicación**

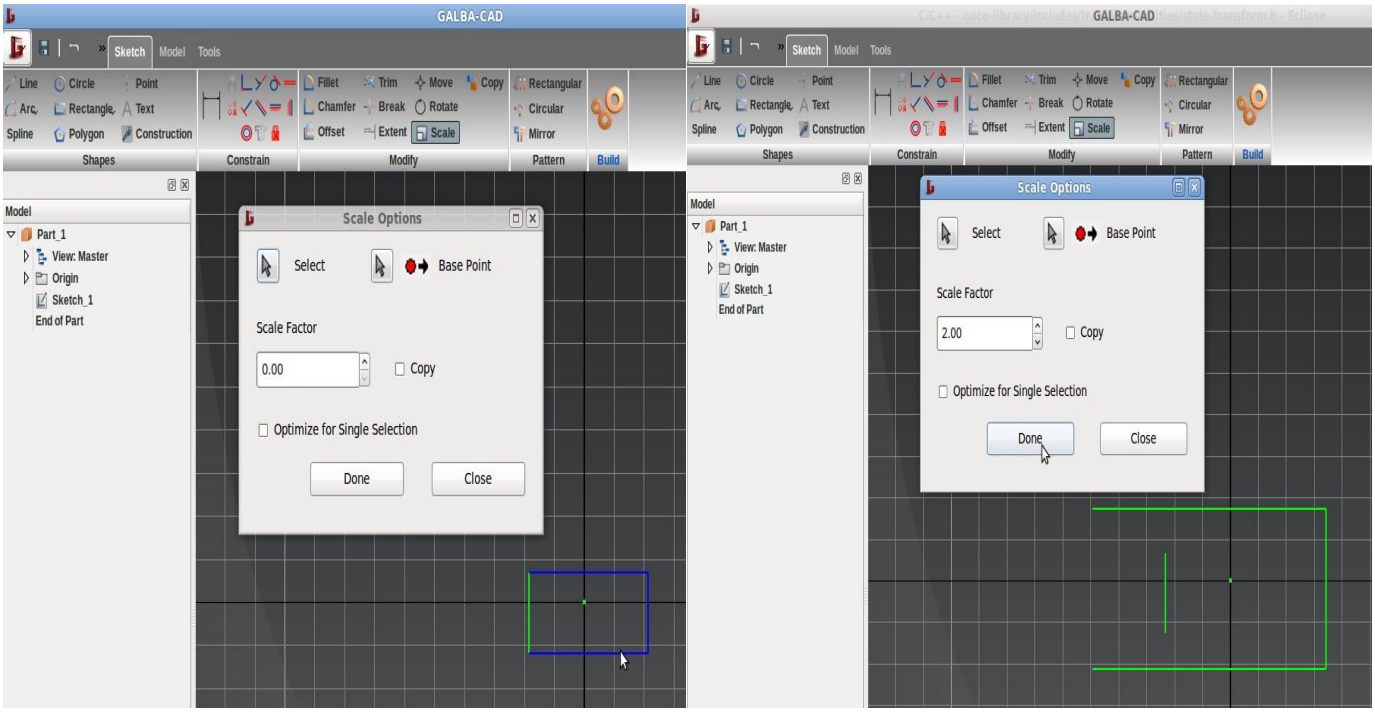


**Figura 16 Ejemplo de escalado de algunas aristas de un rectángulo en la aplicación**

# GLOSARIO DE TÉRMINOS:

**Boceto:** Se refiere al esquema o el proyecto que sirve de bosquejo para cualquier obra. Se trata de una guía que permite volcar y exhibir sobre un papel una idea general antes de arribar al trabajo que arrojará un resultado final.

**Chamfer:** creación de un borde biselado que conecta 2 aristas. Si estas están formando un ángulo recto, el chamfer se observa como un corte de la esquina donde cada arista forma con el mismo un ángulo de 45 grados.

**Fillet:** Consiste en darle a las uniones de aristas de un polígono forma redondeada, convirtiéndola en un arco de circunferencia.

**Framework:** marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes software para el desarrollo rápido de aplicaciones.

**Fresado:** Operación mecánica que permite labrar superficies planas o con distintos Perfiles, así como perforar y canalizar piezas mecánicas.

**Layout:** Suele utilizarse para nombrar al esquema de distribución de los elementos dentro un diseño.

**Map:** Contenedor dinámico perteneciente a la biblioteca estándar de C++, que almacena en cada posición 2 valores, una llave, que será la utilizada para hacer referencia a esa posición, y no la posición como sucede en los arreglos tradicionales; y el valor como tal.

**Matriz:** Conjunto de números colocados en líneas horizontales y verticales y dispuestos en forma de rectángulo; la posición de cada número en la matriz determina las operaciones matemáticas que hay que hacer para hallar un resultado

**Plugin:** Es un accesorio de software o paquete de hardware que es utilizado en conjunto con una aplicación o dispositivo ya existente, para extender sus capacidades o brindar funciones adicionales.

**Primitiva:** Forma básica 2D que al ser conjugada con otras, permite construir diseños y bocetos en 2 dimensiones. Son ejemplos el punto, la línea, y el arco de circunferencia.

**Prototipo:** Primer ejemplar que se fabrica de una figura, un invento u otra cosa, y que sirve de modelo para fabricar otros iguales, o molde original con el que se fabrica

**Punto pivote:** Punto utilizado como centro para ejecutar alguna operación específica.

**SDK:** Un kit de desarrollo de software o SDK (siglas en inglés de software development kit) es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas y sistemas operativos.

**Sistema de referencia global:** Coordenadas locales de los espacios X, Y y Z que define un entorno virtual en específico.

**Spline:** es una curva diferenciable definida en porciones mediante polinomios. Hace referencia a una amplia clase de funciones que son utilizadas en aplicaciones que requieren la interpolación de datos, o un suavizado de curvas. Son utilizados para trabajar tanto en una como en varias dimensiones.

**Subversion:** Subversion es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido.

**Trim:** Es un corte que se realiza a las aristas 2D por sus extremos, donde pueden estar conectadas con aristas vecinas.