Universidad de las Ciencias Informáticas
SIPII, Soluciones Informáticas para la Ingeniería y la Industria, Facultad 4

# Funcionalidades para el módulo *Drawing* en la aplicación de diseño asistido por computadora **Ingeniero**.

**Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas**

Autor: Mario Sánchez Regueira

Tutores: Dr.C. Augusto Cesar Rodríguez Medina

Ing. Kevin Pérez Mandina

**La Habana, 2019**

*Ser culto es el único modo de ser libre*
*José Martí*

# Dedicatoria

*A mi mamá quien defino como mi creadora, madre, amiga y hermana, a mi papá quien defino como mi creador, padre, amigo, hermano y ejemplo a seguir, y a mi hermano persona que no es mi creador y ha tomando la tarea de educarme, enseñarme y quererme.*

# Agradecimientos

En esta ocación quiero agraecerle a todas las personas que han jugado un papel muy importante en mi desarrollo como ingeniero y ser humano.
Quiero agradecer a mí madre Margarita Regueira Delgado por ser la escolta de cada uno de mis sueños aunque sea una locura, por se mi creadora, hermana y amiga incondicional; a mi padre Mario Alberto Sánchez Walcott por ser el guerrero que no me deja solo en ninguna battalla, por dejarlo todo en el terreno, ser mi creador, hermano, ekobio, amigo y ejemplo a seguir; a mi hermano Alberto Sánchez Regueira que a pesar de solo ser 6 años mayor que yo, siempre has estado ahí para mi como un padre de mi edad, un ejemplo ha seguir, una escuela de comportamiento ante la sociedad, por ser mi bro; a Lillevi Rossi Sánchez por enseñarme que con conceptos como la disciplina y constancia todo se puede lograr, a mi tutor Augusto Cesar Rodriguez Medina por a confiar en mi para desarrollar un tema tan impresionante como este, pos ser un profesional ejemplar y digno de admirar; a mi cotutor Kevin Pérez Mandina, hermano, amigo, apoyo, compañero de travesia en estos añs de la vida universitaria; a mi primo Carlito por apoyarme durante todo el proceso y por ser mas que un primo un hermano, a mi amigo Hernes Noriega por tambien ser un hermano y acompañarme durante este período; a Eduardo Acosta Pimentel por ser más que un amigo mi familia, a la persona tan especial que me ha acompañado los ultimos dos años de esta hermosa travesia, sin importar la situacion siempre estaba ahi, fajados , contentos, peleados, unidos a mi novia Jennifer Rutt Dominguez Cuff; a amigos como Leandro el mulato, Eduardo, Bernardo, Israel Machine la maquina francesa, Javier, entre muchos más; a mis compañeros de aula por aguantarme; a todos los profesores que han ayudado y apoyado a que este momento se haga realidad, formandome como futuro profesional; muchas

*gracias a todos los presentes que no mencione por compartir este momento tan importante para mi gracias a todos y gracias a la vida.*

# Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

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

Mario Sánchez Regueira
Autor

Dr.C. Augusto Cesar Rodríguez Medina
Tutor

Ing. Kevin Pérez Mandina
Tutor

# Resumen

El Diseño Asistido por Computadoras (CAD, por sus siglas en inglés) se refiere al uso de herramientas informáticas para acelerar de diseño y la fabricación de cualquier tipo de producto. Estos sistemas desarrollan el sector industrial mejorando la calidad y producción. El módulo ***Drawing*** está entre los principales componentes de los sistemas de diseño asistidos por computadora CAD. Este módulo tiene como finalidad la obtención de planos a partir de objetos tridimensionales modelados en la aplicación, para ser posteriormente construidos. Estos planos se conforman utilizando elementos como vistas o proyecciones, escalas, dimensiones, tablas, notas, etc. En este documento se presenta una investigación que tiene como fin el desarrollo de un funcionalidades para facilitar la obtención de la documentación técnica como las vistas principales, el salvado en formato svg, actualizar las vistas, eliminar las mismas, etc. Además, se realizaron pruebas de calidad evidenciando que el componente implementado funciona correctamente y cumple con los requisitos especificados por el cliente.

**Palabras clave:** ***Drawing***, svg, planos,documentación técnica.

# Índice general

# Índice de figuras

# Índice de tablas

# Introducción

El presente trabajo contiene los resultados de una investigación realizada, con el propósito de agregar funcionalidades al módulo ***drawing***, de la aplicación para el diseño asistido por computadora **Ingeniero**; estas le permiten al diseñador conformar parte de la documentación técnica de los modelos diseñados, específicamente los planos con las vistas principales, vista base, vistas proyectadas(lateral, frontal, superior e isométrica), la vista de detalle y la vista de sección.

El documento está estructurado en tres capítulos, el primero contiene la fundamentación teórica del trabajo, en la cual se reflejan las características de algunos sistemas de diseño asistidos por computadora comerciales, específicamente de los módulos que contienen esos sistemas para obtener los planos y también de algunas alternativas de código abierto con funcionalidades destinadas a ese fin; en el segundo se expone la propuesta de solución, el modelado del dominio y otros aspectos fundamentales requeridos para el desarrollo del módulo; el capítulo 3 contiene aspectos relacionados con la fase de implementación y pruebas realizadas a la propuesta de solución.

Los módulos ***Drawing*** en los sistemas de diseño asistido por computadora, pueden verse como una solución a la automatización del proceso de producción de planos en un proyecto, lo cual constituye una de las fases más importantes para concebir cualquier producto en las industrias. La documentación técnica es importante debido a que es la manera en que el diseñador hace llegar la información del diseño al encargado de la manufactura del producto de manera concreta y sin ambigüedades; en los sistemas CAD la producción de esta se genera mediante el módulo Drawing, grupo de funcionalidades para la obtención de planos de manera similar a la tradicional, imprimiendo la información necesaria sobre una plantilla. Algunos de los sistemas más utilizados son "***Autodesk Inventor***"de "***Autodesk Incorporated*** ", "***SolidWork***"de "***SolidWorks corporation***", "***Solid Edge***"de "***Siemens PLM Softwares***"y "***Catia V5***"de "***Dassault System***".

A nuestro país se le prohíbe el acceso a sistemas informáticas considerados de alta tecnología, debido a las cláusulas del bloqueo económico y financiero impuesto por el gobierno de los Estados Unidos de América(EcuRed,2014), por lo que obtener soluciones informáticas con recursos propios, empleando tecnologías y herramientas de código abierto puede constituir una variante para eludir las restricciones del bloqueo. Este aspecto es uno de los propósitos de este trabajo, que está insertado en un proyecto del grupo de investigación Soluciones Informáticas para la Ingeniería y la Industria (SIPII) perteneciente a la Facultad 4.

Partiendo de investigaciones anteriores se han tenido en cuenta las necesidades y carencias para realizar un correcta documentación técnica, caracterizando el contexto problemático en el que se inserta este tema.

Los aspectos tecnológicos de la situación problemática identificados son:

- No se cuenta con un módulo informático en sistemas CAD de código abierto, que permita obtener una documentacion técnica de manera correcta y eficiente.
- Existen varias aplicaciones de código abierto que poseen algunas funcionalidades, el más completo se encuentra en la aplicación **FreeCAD**, pero posee insuficiencias como:
    - La vista principal no se proyecta en el medio de la hoja.
    - No permite modificar la escala.
    - No permite mover las vistas.
    - La vistas ortogonales varían de escala y tamaño dependiendo de la cantidad de proyecciones.
    - No posee varias funcionalidades como: vista de detalle, vista de sección, actualizar las vistas, eliminar las vistas, ni salvar el documento.

Sobre la base de lo anteriormente expuesto se formula el siguiente **problema de investigación:** ¿Como conformar las vistas principales en los planos de piezas del módulo, en la aplicación de diseño asistido por computadora **Ingeniero**?
Para dar solución al problema anteriormente planteado queda definido como **objeto de estudio:** Algoritmos para extraer las principales proyecciones de los objetos tridimensionales y el **campo de acción:** queda enmarcado en las funcionalidades de un módulo de planos en la aplicación **Ingeniero**.

Para dar solución al problema descrito anteriormente, se define como **objetivo general:** Desarrollar funcionalidades necesarias para obtener las vistas principales de modelos 3D en los planos de piezas de la aplicación para el diseño asistido por computadora **Ingeniero**.

Para dar cumplimiento al objetivo general, se deberá cumplir con los siguientes **objetivos específicos:**

- Identificar los aspectos principales de las herramientas propietarias que pueden apoyar al proceso de desarrollo a la propuesta de solución.
- Implementar las funcionalidad para la vista frontal.
- Implementar las funcionalidad para la vista lateral.
- Implementar las funcionalidad para las vistas isométrica.
- Implementar las funcionalidad para la escala.
- Implementar las funcionalidad para actualizar la vista.
- Implementar las funcionalidad para la vista de detalle.
- Implementar las funcionalidad para eliminar una vista.
- Implementar las funcionalidad para la vista de sección.
- Implementar las funcionalidad para salvar en formato .SVG.
- Validar las funcionalidades del módulo para gestionar la documentación de la información en la aplicación para el diseño asistido por computadora **Ingeniero**.

Para el cumplimiento de los objetivos planteados se emplearon los siguientes métodos de investigación, estos son:

**Métodos teórico**

- Analítico-Sintético: se empleó durante el estudio de la fundamentación teórica mediante el análisis documental y análisis de código fuente existente lo que permitió identificar aspectos sobre como manejar la información del modelo tridimensional y la síntesis se empleo en la propuesta de solución.
- Modelado: Se empleó a través de los diagramas UML utilizados para representar las características y la relaciones entre objetos de la solución.

**Métodos-Empíricos:**

- Observación: Con la aplicación de este método fue posible observar el funcionamiento de las aplicaciones CAD más utilizadas a nivel mundial.
- Experimentación: Se empleó en las pruebas funcionales para observar las respuestas de cada funcionalidad en el sistema.

En los capítulos siguientes se recorre el proceso de documentación referente a la implementación de las funcionalidades.

# CAPÍTULO 1

## Fundamentación Teórica del módulo Drawing

En este capítulo contiene los aspectos fundamentales relacionados con el estado del arte en relación a los módulos destinados a elaborar planos en las tecnologías de diseño asistido por computadora existentes y algunos elementos históricos de los mismos.

## 1.1. Proceso evolutivo de los sistemas CAD

Antes de llegar a su estado actual, se ha transitado por una línea evolutiva, que se describe en los siguientes párrafos con algunos elementos históricos relacionados con el desarrollo de los sistemas CAD.
Tradicionalmente el proceso de producción de plano antes se realizaba de forma manual, esto requería mayor cantidad de personal especializado para realizar la tarea. Se obtuvieron grandes avances como por ejemplo la ***Universal Drafting Machine***, mesa de dibujo, la cual contaba con varias herramientas integradas como escuadras y reglas, con esto mejora el proceso pero seguía siendo de manera manual. En trabajos de gran envergadura existía la posibilidad a la ocurrencia de errores, era desastroso, ya que de existir se debería realizar el proceso nuevamente. Esto cambió en la medida que fueron desarrollándose los sistemas de computo y los sistemas informáticos, los cuales permitieron automatizar este proceso y por consiguiente reducir la cantidad de especialistas, tiempo y costo. Debido al desarrollo de los softwares y ordenadores a partir de la segunda mitad del siglo XX se pensó en automatizar este proceso surgiendo los software para el **Diseño Asistido por Computadora**, en sus inicios solo realizaban trabajos similares al manual en dos dimensiones, pero se convirtieron en una poderosa herramienta de visualización al incluir el modelado 3D y la implementación de varias técnicas para el modelado paramétrico, que permiten diseñar objetos en tres dimensiones, visualizarlos en cualquier ángulo sin volverlo a dibujar, además de modificarlos de forma paramétrica, aumentando la productividad del diseñador, ya que al modificar un parámetro se evita volver a dibujar la pieza; permitiendo trabajar con los modelos existentes, para obtener nuevos resultados. Con esto se aceleró el proceso de diseño reduciendo tiempo, costo y recursos humanos relacionados con este proceso, aumentando la productividad.(Weisberg, 2008)

## 1.2. Sistemas comerciales para el diseño asistido por computadora

### 1.2.1. Solid Edge

Fue lanzado al mercado por primera vez en 1996 inicialmente desarrollado por ***Intergraph***, hoy en día pertenece y es desarrollado por ***Siemens PLM Software***. ***Solid Edge*** es un portafolio de herramientas de software asequible y fácil de usar que abordan todos los aspectos del proceso de desarrollo de productos, diseño en 3D, simulación, fabricación, gestión del diseño y mucho más gracias a un creciente ecosistema de aplicaciones. ***Solid Edge*** combina la velocidad y simplicidad del modelado directo con la flexibilidad y el control de diseño paramétrico, hecho posible con *synchronous technology* (tecnología sincrónica). (Wikipedia, 2010) Este programa comercial solo se puede usar en **Windows** y **MacOS**. ***Solid Edge*** es un software de modelado 3D, con características paramétricas, proporcionando funciones de modelado de sólido, modelado de ensamblajes y vista ortográfica 2D para diseñadores mecánicos. Utilizaba el kernel de modelado geométrico **ACIS**, luego cambió a usar el kernel **Parasolid**. En 1998, fue comprado y desarrollado por ***UGS Corp*** (la fecha de compra corresponde al swap de kernel). Desde septiembre de 2006, ***Siemens*** también ha ofrecido una versión 2D gratuita llamada **Solid Edge 2D Drafting**. Este posee todas la funcionalidades necesarias para una correcta documentación.

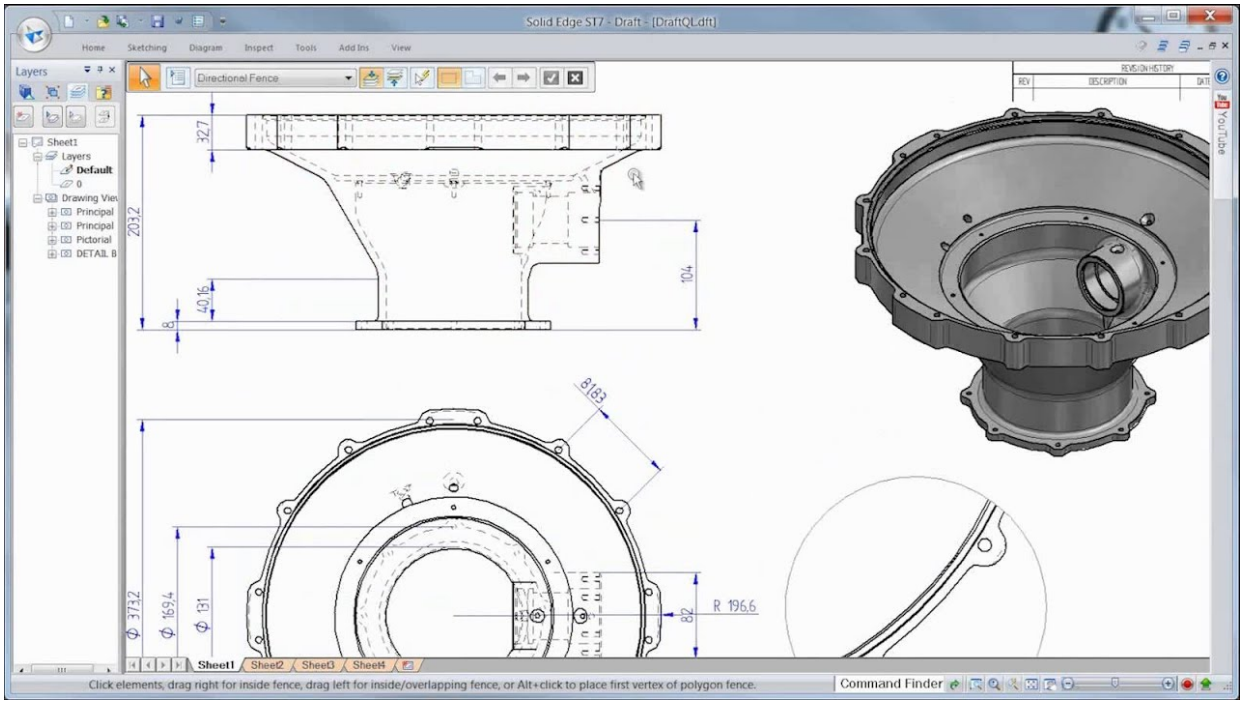



Figura 1.1. Imagen de la creación de planos con **Solid Edge**.

### 1.2.2. Catia V5

Familia ***CATIA*** de programas CAD 2D y 3D de IBM, este fue uno de los primeros programas CAD en proporcionar modelado sólido 3D. El programa fue desarrollado por ***Dassault Systems***, una compañía aeroespacial francesa. ***CATIA*** se ejecuta en IBM AIX, estaciones de trabajo HP-UX, Windows 2000 / XP, SGI IRIX y plataformas Sun Solaris. (Farlex, 2019) Esta herramienta es una de los mas usadas en el modelado 3D y ya que después de generado el modelo se necesita documentarlo en un lenguaje general como los planos técnicos. Posee un módulo para ello con opciones almacenadas en un menú, lo que dificulta un poco la navegación dentro del módulo debido a que existen funciones dentro de otras, la ayuda esta en línea fuera de la aplicación y posee algunas funcionalidades como:

- A través de las propiedades cambiar la escala a una escala global.
- Crear una vista frontal(Esta te permite orientar la vista antes de crearla dígase girar o desplazar).
- Crear una vista proyectada(Esto te permitiría a partir de la vista frontal proyectar la figura en otros ángulos).
- Crear vistas auxiliares.
- Generar dimensiones sobre las vistas obtenidas.

(Kovacevic, 2017) Esta es una de las aplicaciones CAD mas utilizadas, ofrece muchas funcionalidades el módulo drawing de esta aplicación, esta muy completo y tiene como desventaja que la interfaz es un poco engorrosa ya que varias funcionalidades quedan dentro de otras y la navegación se hace un poco difícil.

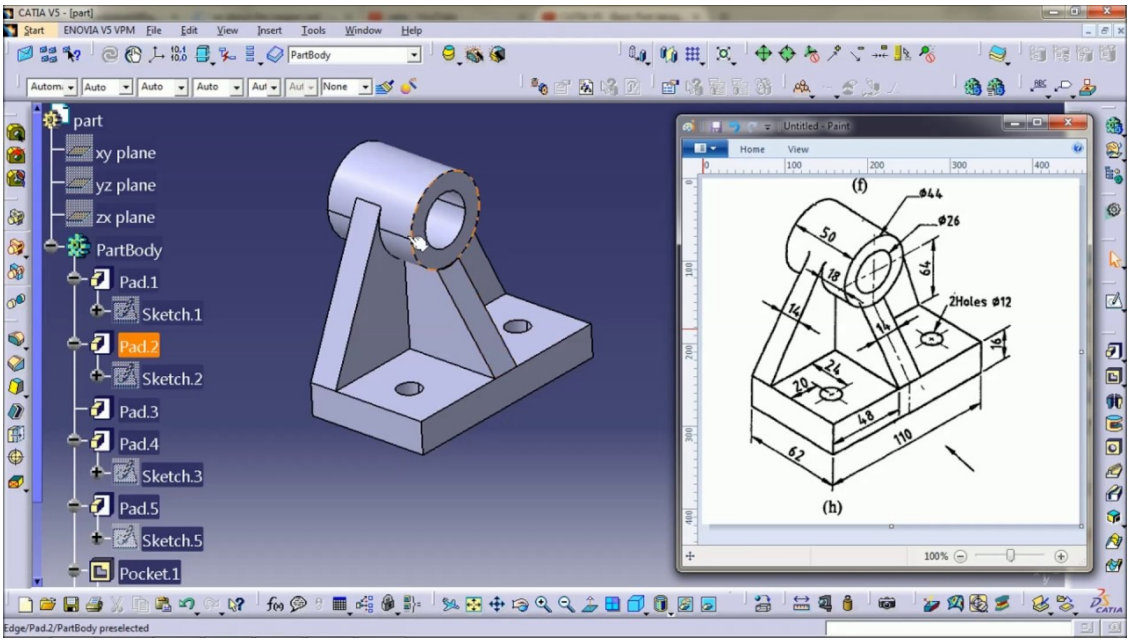



Figura 1.2. Imagen de la creación de planos con **Catia V5**.

### 1.2.3. SolidWorks

La empresa ***SolidWorks Corpotration*** fue fundada en 1993 por **Jon Hirschtick** con su sede en Concord, Massachusetts y lanzó su primer producto, SolidWorks 95, en 1995. En 1997 ***Dassault Systemes***, mejor conocida por su software ***CATIA***, adquirió la compañía y actualmente posee el cien por ciento de sus acciones. ***SolidWorks*** es un paquete de software de automatización de diseño mecánico utilizado para construir piezas, ensamblajes y dibujos que aprovechan la familiar interfaz gráfica de usuario de Microsoft Windows; este software es una herramienta de diseño y análisis fácil de aprender, que hace posible que los diseñadores puedan dibujar rápidamente conceptos en 2D y 3D, crear piezas y ensamblajes en 3D y detallar dibujos en 2D.(Wikipedia, 2012) En ***SolidWorks***, los documentos de pieza, montaje y dibujo están relacionados. Esta herramienta posee un módulo utilizado para insertar, agregar y modificar vistas en un dibujo de ingeniería. Los detalles se refieren a las dimensiones, notas, símbolos y lista de materiales utilizados para documentar el dibujo. Los estándares y tamaños de dibujo especificados para ***Solid Work*** son ANSI(Instituto Nacional Estadounidense de Estándares) o ISO(Organización Internacional de Normalización). Utiliza dimensiones, tolerancia y notas en piezas y ensamblajes para incorporar la intención del diseño en el Dibujo. (Planchard, 2012) Siendo este software uno de los punteros en esta categoría, siendo utilizados en mucho campos como por ejemplo la industria automovilística, la naval, etc.

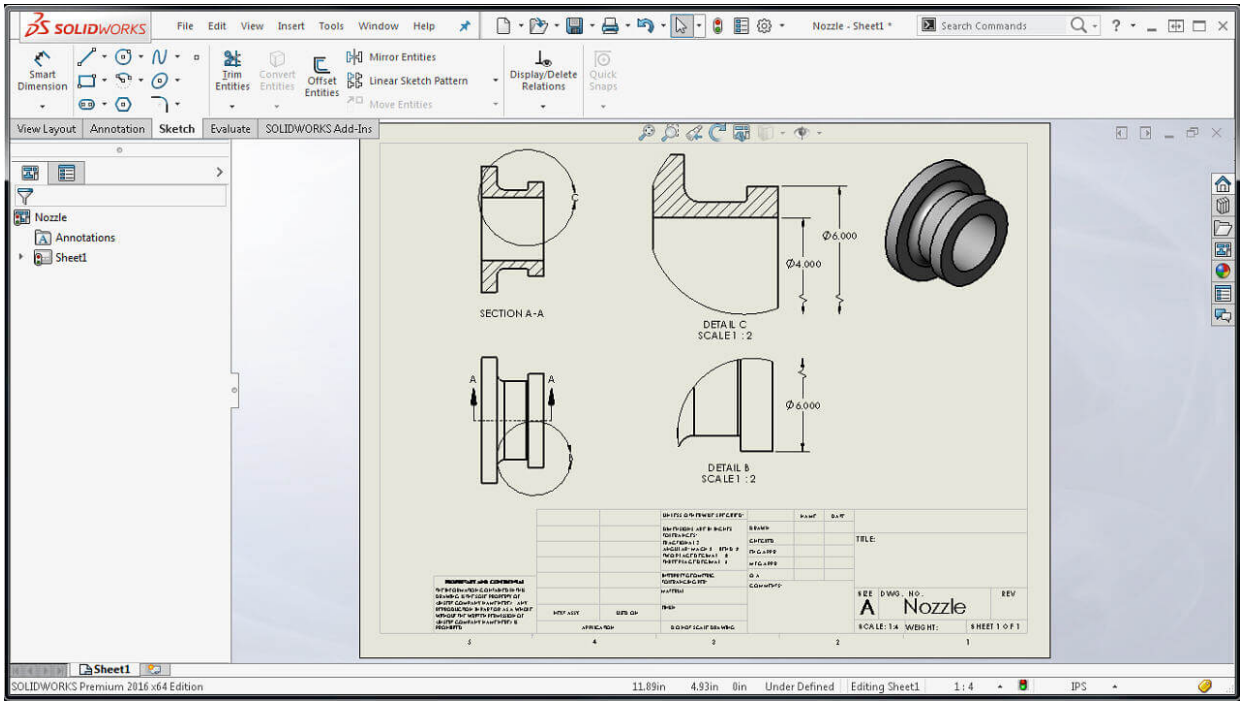
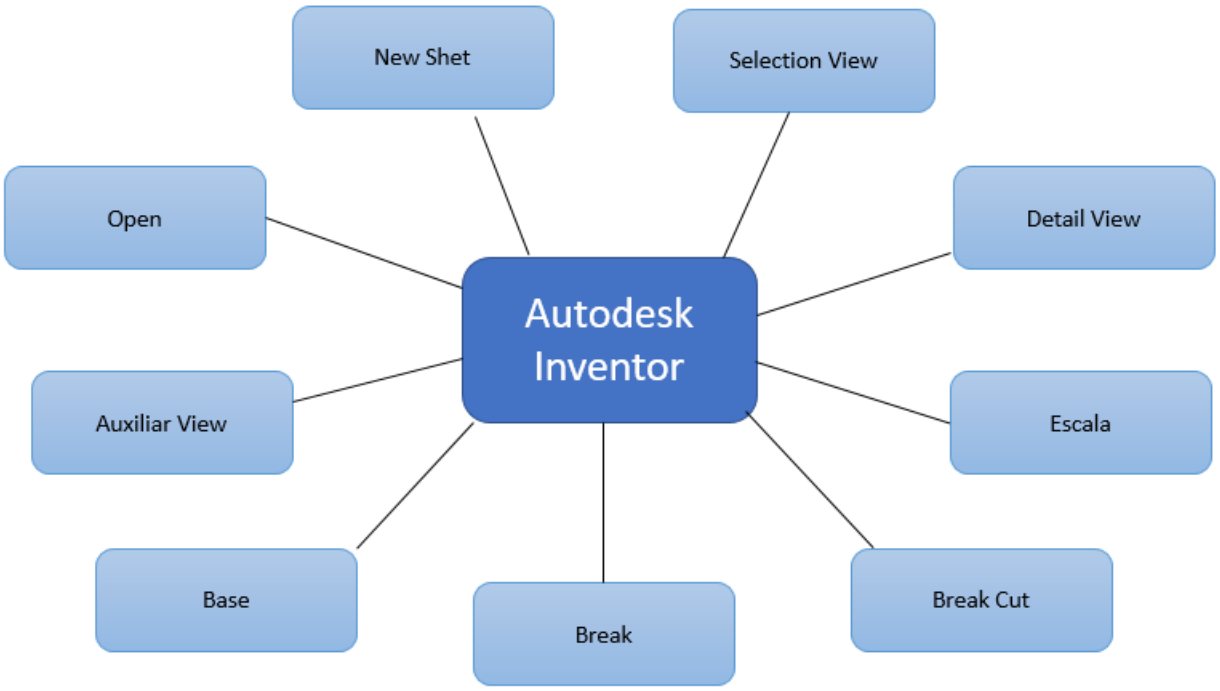



Figura 1.3. Imagen de la creación de planos con **SolidWorks.**

### 1.2.4. Autodesk Inventor

***Autodesk Inventor*** permite la integración de datos 2D y 3D en un solo entorno, creando una representación virtual del producto final que permite a los usuarios validar la forma, el ajuste y la función del producto antes de que se construya. Es una herramienta de diseño asistido por computadora producida por la empresa ***Autodesk*** y salió al mercado por primera vez en 1999; es un modelador paramétrico de sólidos en 3D que permite a los ingenieros o diseñadores crear prototipos digitales, gracias a un conjunto de herramientas para el diseño mecánico 3D, simulación, análisis, visualización y documentación; con esta herramienta se pueden integrar planos en 2D de ***AutoCAD*** y crear representaciones digitales del producto final que les permite validar la forma, dimensiones, precisión del ensamble; incluye herramientas potentes de parametrización, edición directa y modelado de forma libre, así como capacidades de traducción multi-CAD y en sus dibujos .DWG estándar. ***Inventor*** utiliza ShapeManager, el núcleo de modelado geométrico patentado de ***Autodesk***. (Wikipedia, 2014). El programa ahora incluye una gran cantidad de funciones que aumentan la velocidad y la versatilidad de la creación y anotación de dibujos en 2D. Aun así, no tiene el conjunto completo de funciones de dibujo que se encuentran en ***Mechanical Desktop***. ***Inventor*** se ha convertido en uno de los mejores modeladores paramétricos sólidos, pero las versiones anteriores se quedaron cortas en el departamento de documentación 2D. (cadalyst, 2004)
Después de analizado y probado sus funcionalidades este software posee un módulo drawing muy sofisticado y bien implementado llegando a ser la línea a superar por la solución. Al mismo se le realizó un análisis de todas sus funcionalidades expuestas a continuación.




- La opción New Sheet permite abrir una nueva plantilla seleccionando la norma y escala a utilizar

sobre la misma.

- Open se utiliza para abrir un trabajo que se desea modificar o no se ha terminado.
- Base se utiliza para plasmar una vista base del modelo definida por el usuario a través de la visualización del mismo sobre la plantilla.
- Auxiliar View permite proyectar vistas no principales para información adicional que sea necesaria.
- Section View se emplea para poder ver la estructura interna de la pieza u objeto en aras de garantizar una correcta fabricación.
- Detail View es una vista que permite ver un detalle de la pieza u objeto en una escala mayor permitiendo tener una exacta visión del mismo.
- Escala es una propiedad del dibujo que queda definida desde que se selecciona la plantilla de dibujo y solo varia en caso de ser necesario como por ejemplo en las vistas de detalle.
- Break Cut permite realizar una vista de rotura. para ver el interior de la figura es un tipo de vista de detalle.
- Break es utilizado para piezas largas y monotonas permitiendo conservar una parte de la misma.

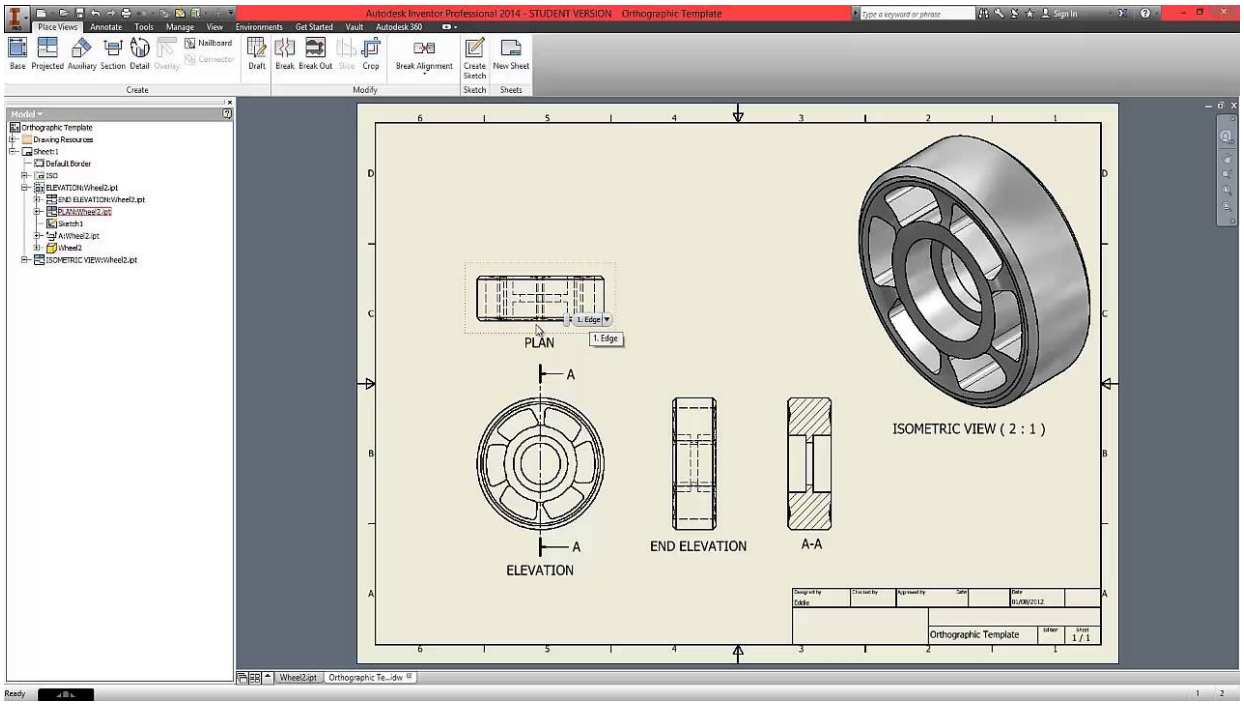



Figura 1.4. Imagen de la creación de planos con **Autodesk Inventor**.

El análisis de estos módulos permitió generar ideas con respecto a la implementación de las funcionalidades, e integración de las mismas en la aplicación **Ingeniero**. A continuación presentamos algunos conceptos de suma importancia para una mejor comprensión del tema, relacionado con la rama de la informática conocida como **gráficos por computadoras**, empleados en la investigación.

## 1.3. Gráficos por computadora

Sería difícil exagerar la importancia de los gráficos por computadoras en nuestras vidas. La combinación de computación, redes y el complejo sistema visual humano, a través de gráficos de computadora, ha sido fundamental en estos avances y ha creado nuevas formas de mostrar información, como ver mundos virtuales y comunicarse con otras personas y máquinas. Los gráficos por computadora se ocupan de todos los aspectos relacionados con la producción de imágenes utilizando una computadora. La actividad de diseño es un área que se ha visto vinculada con el desarrollo de esta rama de la computación. El campo comenzó hace 50 años, con la exhibición de algunos líneas en los Tubos de Rayos Catódicos (CRT, por sus siglas en inglés) Ahora podemos generar imágenes por computadora que son indistinguibles de las fotografías y objetos reales. (Mcguire; Hudges y Dam, 2015)

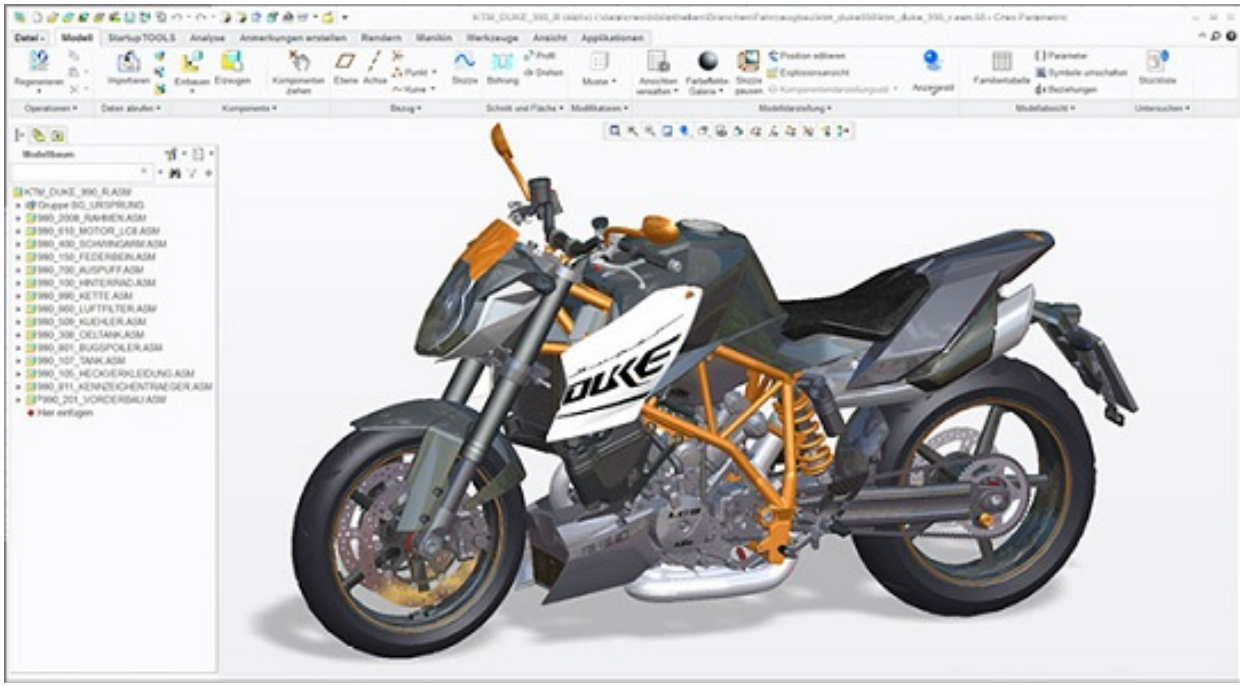


Figura 1.5. Gráficos por computadoras

**Aplicaciones de los gráficos por computadora**

Los gráficos por computadoras se pueden dividir en 4 grandes grupos los cuales son: (Martins; Oliveira; S. Silva; A. Silva y Teixeira, 2011)

- Visualización de la información.
- Simulación y animación.
- Interfaces de usuario
- Diseño.

Aunque muchas aplicaciones abarcan dos o más de estas áreas, el desarrollo de este campo se basó en el

trabajo separado en cada uno.

### 1.3.1. Diseño Asistido por computadoras

El uso de las herramientas gráficas interactivas en diseño asistido por computadora CAD se extienden en campos como la arquitectura y el diseño de piezas mecánicas. Los programas CAD modernos son muy costosos; tienen una amplia variedad de tipos, propuestos para diferentes aplicaciones. Pueden usar curvas Non Uniform Relational B-Spline (NURBS, por sus siglas en inglés), alambres o sólidos para crear arquitecturas de dibujos verdaderamente en 3D. Este diseño tiene verdadera profundidad en su disposición, que permite a los diseñadores rotar sus diseños 360 grados alrededor de tres ejes diferentes; estos sistemas la representación más precisa de cómo se verá el objeto cuando se produzca realmente. Luego de eso se procede a documentar la información del modelo tridimensional en planos técnicos mediante varias funciones con el objetivo de comunicar esta información a la industria de manera física agilizando el proceso aumentando su calidad y disminuyendo su costo, tiempo y mano de obra.(Martins; Oliveira; S. Silva; A. Silva y Teixeira, 2011)

**Ventajas del diseño asistido por computadora**

Las ventajas del sistema CAD son numerosas, como se menciono pero: "Se estima que un buen sistema CAD ahorre"

- 90 por ciento en tiempo de diseño conceptual.
- 25 por ciento en costo de diseño.
- 30 por ciento en tiempo de licitación.
- 15 por ciento en costo de construcción.
- 40 por ciento en gastos de préstamo de construcción.

(ResearchGate, 2015)

Para poder representar imágenes en la pantalla de los ordenadores, en los sistemas de diseño fue necesario definir varias estructuras o componentes geométricos para manejar la información.

### 1.3.2. Primitivas geométricas en gráficos por computadoras

**Punto**

Nuestro objeto geométrico fundamental es un punto, la única propiedad que posee un punto es su ubicación; un punto matemático no tiene ni tamaño ni forma. En las aplicaciones de diseño son utilizados para definir los vértices de las geometrías de los cuerpos, los puntos son útiles para especificar objetos geométricos pero no son suficientes para ello. Se necesitan números para especificar cantidades como la distancia entre dos puntos.

**Línea**

La línea se puede definir como la suma de un punto y un vector (o la resta de dos puntos), lleva a la noción de una línea en un espacio afín; también se puede denominar como una secuencia de puntos, las líneas pueden ser rectas, curvas o quebradas.

Estas dos primitivas fundamentales se utilizan para representar todas las geometrías; pero no es solo su definición sino que necesitamos combinarlas y desplazarlas por las coordenadas de la pantalla, esto se puede lograr mediante las transformaciones.

### 1.3.3. Transformaciones

En esta sección, primero se muestra cómo podemos describir las transformaciones afines más importantes de forma independiente de cualquier representación. Consideramos las transformaciones como formas de mover los puntos que describen uno o más objetos geométricos a nuevas ubicaciones. Aunque hay muchas transformaciones que moverán un punto en particular a una nueva ubicación, casi siempre habrá solo una forma única de transformar una colección de puntos a nuevas ubicaciones preservando las relaciones espaciales entre ellos, aplicado la misma transformación a todos los vértices del objeto, resultará en un objeto desplazado del mismo tamaño y orientación. (Snaider y Eberly, 2016)

**La Traslación**

Es una operación que desplaza puntos por una distancia fija en una dirección. Para especificar una traslación, solo necesitamos especificar una vector de desplazamiento d, porque los puntos transformados están dados por P = P + d para todos los puntos P del objeto.(Angel y Shreiner, 2012)
esto desplazaría el objeto una distancia igual a la del vector d en la dirección del mismo.

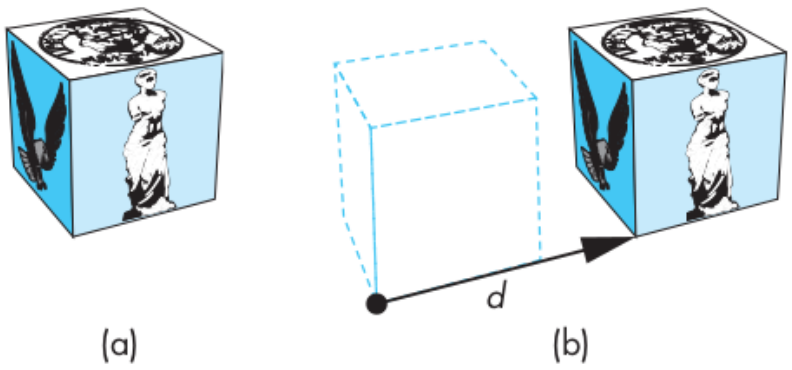



Figura 1.6. Traslación

**La Rotación**

La rotación es más difícil de especificar que la traslación porque debemos especificar más parámetros, un

punto fijo y un ángulo de giro por ejemplo un objeto con respecto al punto central del mismo como se muestra en la figura a continuación. Definiendo como el punto(X,Y) y (X,Y)prima la variación Gamma como ángulo de giro. Con esta transformación se obtendrá algo así. (Angel y Shreiner, 2012)

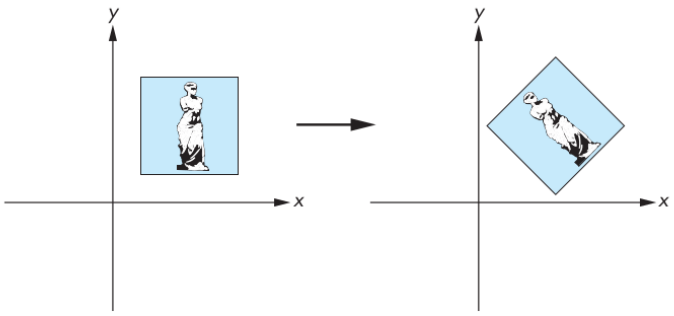



Figura 1.7. Rotación de un objeto

**Escalado**

El escalado es una transformación afín no rígida del cuerpo mediante la cual podemos hacer un objeto más grande o más pequeño. Necesitamos una escala no uniforme para construir el conjunto completo de transformaciones afines que usamos en el modelado y la visualización mediante la combinación de una secuencia elegida de escalas, traslaciones y rotaciones. Por lo tanto, para especificar una escalada, debemos definir el punto fijo, una dirección en la que se desea variar el objeto y un factor de escala. Para un número >1, el objeto se alarga en el valor especificado en esa dirección; para un número entre 0 y 1, el objeto se hace más pequeño en esa dirección.(ibíd.)

a continuación se presenta una imagen con una escala uniforme y otra no uniforme

Todo estos procesos son realizados mediante matrices que permiten aplicar todo tipo de operaciones y transformaciones básicas de manera matemática y de fácil manejo para la máquina.(Martins; Oliveira; S. Silva; A. Silva y Teixeira, 2011) Con solo esto no se puede realizar todas los cambios deseados y trabajar con cada uno de estos elementos como una operación es trabajoso. Por eso surgen grupos de funcionalidades con fácil acceso denominados API(Aplication programing interface o Interfaz de programación de aplicaciones)

## 1.4. Interfaz de Programación de Aplicaciones

Una Aplication Programing Interface (API, por sus siglas en inglés) es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. La interfaz entre un programa de aplicación y un sistema de gráficos puede ser especificado a través de un conjunto de funciones que reside en estas bibliotecas de gráficos. Las aplicaciones CAD solo ven la API y, por lo tanto, está protegido de los detalles del hardware y la implementación del software de la librería gráfica. Los controladores de software son solo para interpretar la salida de la API y convertir estos datos a una forma

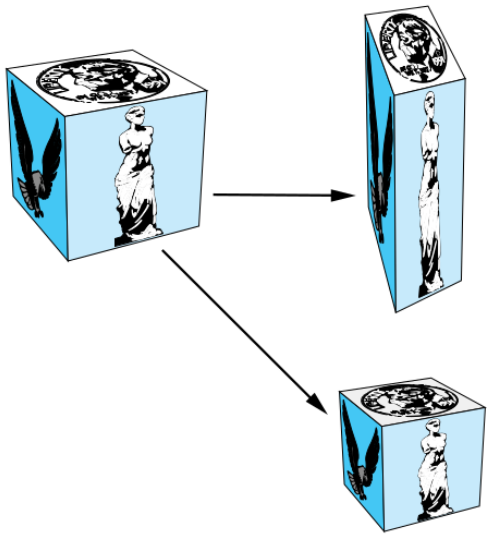


Figura 1.8. Escalado de un objeto

que se entienden por el hardware particular.(ReseachGate, 2014)

Existen varias APIs especializadas en el proceso de diseño como por ejemplo OpenGL y OpenCASCADE. Las mismas ofrecen muchas funcionalidades para el trabajo con modelos tridimensionales tomando los


Figura 1.9. OpenGL


Figura 1.10. Open Cascade

modelos bidimensionales como un caso particular de los tridimensionales donde la profundidad es 0. En la aplicación ingeniero se utiliza Open Cascade como tecnología de modelado brindando muchas funcionali-

dades para cualquier banco de trabajo de la misma. Para la propuesta de solución fue necesario un estudio de las funcionalidades que brinda para la obtención de las principales vistas; esto a través de una relación del objeto modelado y el visor espectador que es manejado por un vector tridimensional que lo coloca en el sistema para obtener la proyección deseada. Para la conformación de planos de piezas mecánicas mayormente se utilizan dos tipos de vista(Ortogonales e isométrica)

## 1.5. Proyecciones en gráficos por computadora

Cuando un arquitecto dibuja una imagen de un edificio, sabe qué lado desea mostrar y, por tanto, dónde debe colocar el espectador en relación con el edificio. Cada vista clásica está determinada por una relación específica entre los objetos y el espectador.(Martins; Oliveira; S. Silva; A. Silva y Teixeira, 2011) Existen varios tipos de vistas clásicas a continuación se muestran las más utilizadas en el diseño asistido por computadora.

### 1.5.1. Proyección ortográfica

Nuestra primera vista clásica es la proyección ortográfica. En las vistas ortográficas (u ortogonales), los proyectores son perpendiculares a una de las caras del objeto. En una proyección ortográfica multivista, realizamos múltiples proyecciones; por lo general, usamos tres vistas, como la frontal, la superior y la derecha, para mostrar el objeto. (Snaider y Eberly, 2016) La visualización de estas imágenes puede requerir habilidad por parte del espectador. La importancia de este tipo de vista es que conserva distancias, ángulos y no hay distorsión de la imagen.

### 1.5.2. Proyecciones axonométricas

Si queremos ver más caras principales de nuestro objeto en una sola vista, debemos eliminar una de nuestras restricciones. En las vistas axonométricas, los proyectores aún son ortogonales al plano de proyección, pero el plano de proyección puede tener cualquier orientación con respecto al objeto. Si se coloca el plano de proyección simétricamente con respecto a las tres caras principales que se encuentran en una esquina de nuestra objeto rectangular, entonces tenemos una vista isométrica; si se coloca el plano de proyección simétricamente con respecto a dos de las caras principales, entonces la vista es dimétrica. Por ejemplo en estas vistas un círculo se proyecta en una elipse. Esta distorsión es el precio que pagamos por la capacidad de ver más de una cara principal en una vista que puede producirse fácilmente ya sea a mano o por ordenador. Las vistas axonométricas se utilizan ampliamente en arquitectura y diseño mecánico.(Angel y Shreiner, 2012) La más empleada de ellas es la isométrica por arquitecto, diseñadores e ingenieros.

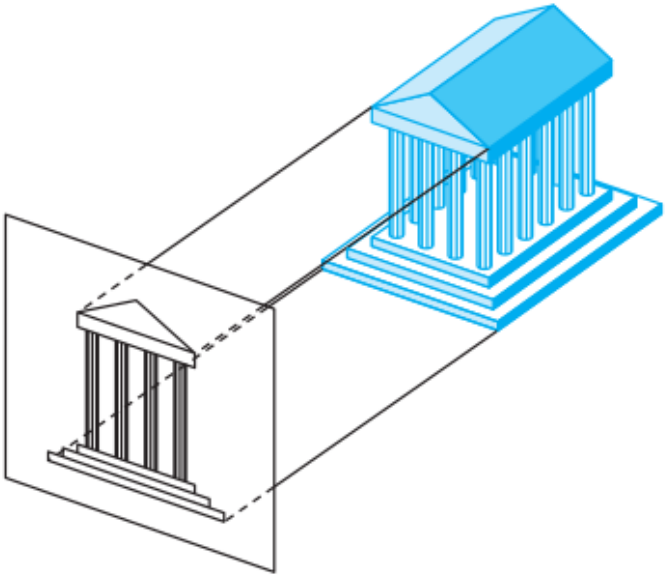


Figura 1.11. Proyección ortográfica

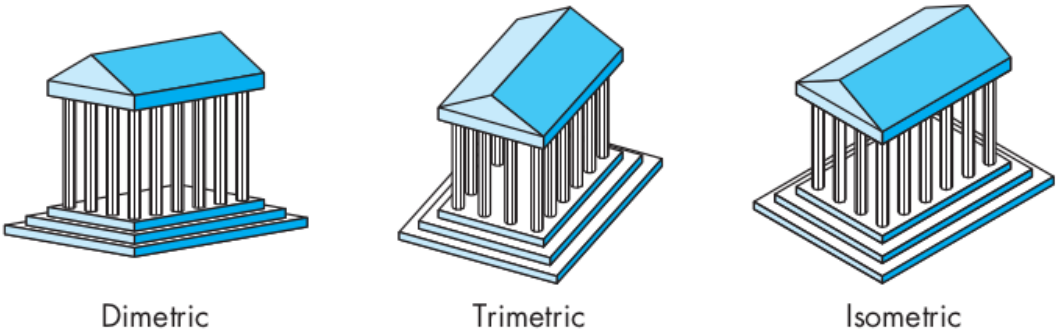



Figura 1.12. Proyección en axonométrica

### 1.5.3. Proyecciones en perspectivas

Todas las vistas en perspectiva se caracterizan por la disminución de tamaño. Cuando los objetos se alejan del espectador, sus imágenes se hacen más pequeñas. Este cambio de tamaño da la perspectiva en su aspecto natural; sin embargo, debido a la cantidad en que la línea está acortada depende de lo lejos que esté la línea del espectador, no podemos hacer mediciones desde una vista en perspectiva. Por lo tanto, el mayor uso de las vistas en perspectiva es en aplicaciones como la arquitectura y la animación, donde es importante lograr imágenes de aspecto natural. (Mcguire; Hudges y Dam, 2015) Las vistas en perspectiva clásicas se conocen generalmente como de uno, dos y tres puntos de fuga. Cualquier esquina del edificio incluye las tres direcciones principales. En el caso más general la perspectiva de tres puntos: líneas paralelas en cada

una de las tres direcciones principales converge a un punto de fuga finito, debe ser evidente desde el punto de vista del desarrollador de programas que las tres situaciones son simplemente casos especiales de visión la general en perspectiva.
aquí se puede observar la figura 1.11 (a en tres puntos de fuga la (b en dos puntos y (c en un punto.

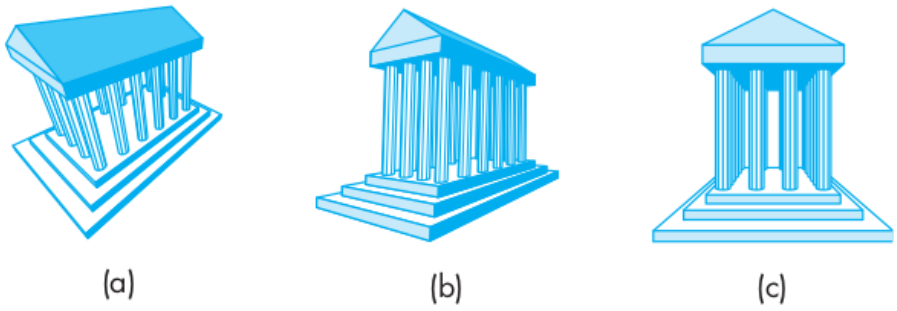



Figura 1.13. Proyección en perspectiva

Todo esto es posible de manejar mediante funciones de las librerías mencionadas anteriormente, estas poseen una inmensa cantidad de funciones que fue necesario utilizar y documentar debido a la importancia que poseen en la presente investigación. Todo lo anteriormente descrito se pudo observar en el código de la aplicación FreeCAD, único modelador tridimensional de código abierto, cual posee algunas funcionalidades para el tratamiento de las vistas y gestión de la documentacion técnica.

## 1.6. FreeCAD como alternativa de codigo abierto

"FreeCAD es un modelador CAD 3D. El desarrollo es completamente de código abierto (licencia LGPL 5 ). FreeCAD está dirigido directamente a la ingeniería mecánica y diseño de productos, pero también se emplea en una amplia gama de usos en la ingeniería, como la arquitectura. Es un modelador paramétrico con una arquitectura de software modular que hace fácil proporcionar funcionalidades adicionales sin modificar el sistema central. Al igual que muchos modeladores modernos CAD 3D tiene componentes 2D con el fin de esbozar formas en dos dimensiones o extraer los detalles del diseño del modelo de 3D para crear dibujos de producción en 2D."(Habrent y comunity, 2015) FreeCAD hace un uso intensivo de todas las grandes bibliotecas de código abierto que existen en el campo de la Computación Científica; entre ellas se encuentran Open Cascade, un núcleo CAD de gran alcance, Coin3D, una encarnación de Inventor abierto, Qt y Python. FreeCAD también es totalmente multiplataforma, y actualmente se ejecuta sin problemas en los sistemas Windows, Linux / Unix y Mac OSX, con el mismo aspecto y las funcionalidades exactas en todas las plataformas. Contiene un módulo drawing que se encarga de la creación y manipulación de hojas de dibujo en 2D, que se utilizan para mostrar vistas de su trabajo 3D en 2D.

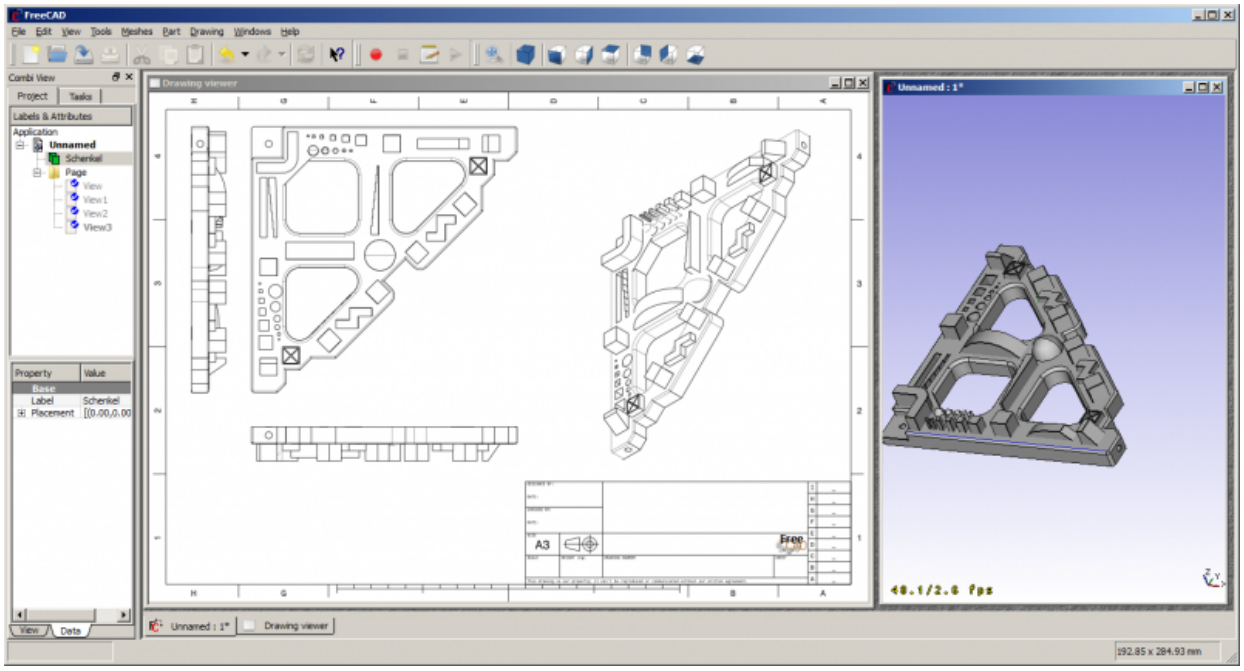


Figura 1.14. Imagen de la creación de planos con **FreeCad**.

Brindando funcionalidades como las que se muestra en la imagen continuación.

| Tool | Description | Tool | Description |
|---|---|---|---|
| New sheet | Creates a new drawing sheet | Insert view | Inserts a view of the selected object in the active drawing sheet |
| Annotation | Adds an annotation to the current drawing sheet | Clip | Adds a clip group to the current drawing sheet |
| Browser preview | Opens a preview of the current sheet in the browser | Ortho Views | Automatically creates orthographic views of an object on the current drawing sheet |
| Symbol | Adds the contents of a SVG file as a symbol on the current drawing sheet | Draft View | Inserts a special Draft view of the selected object in the current drawing sheet |
| Export | Saves the current sheet as a SVG file | | |

Este módulo drawing es el mas completo en herramientas de softwares libre, pero comparado con los de las herramientas comerciales se denota de ineficiente, debido a que le faltan por implementar varias funciona-

lidades y las existentes poseen varios defectos antes mencionados, el mismo fue estudiando con el objetivo de reutilizar el código existente.
A continuación se muestra una tabla de comparación de los softwares antes mencionados respecto a las funcionalidades que implementan.

Tabla 1.1. Tabla de comparación de los módulos existentes

| Funcionalidades | Autodesk Inventor | Catia | SolidWork | Solid Edge | FreeCAD |
|---|---|---|---|---|---|
| New Sheet | si | si | si | si | si |
| Open | si | si | si | si | si |
| BaseView | si | si | si | si | si |
| ProyectedView | si | si | si | si | si |
| SectionVeiw | si | si | si | si | no |
| DetailView | si | si | si | si | no |
| Break | si | no | no | no | no |
| BreakOut | si | no | no | no | no |
| UpdateView | no | no | no | no | no |

Para el comienzo de la realización de la propuesta de solución fue necesario la revisión del código de la aplicación FreeCAD; de donde se tomaron ideas para dar solución a el problema inicial.

## 1.7. Conclusiones parciales

Luego del estudio llevado a cabo se arriba a las siguientes conclusiones parciales:

- Se identificó todas las carencias que poseen todos los módulos **Drawing** en las herramientas de código abierto.
- Considerar que una de las cuestiones principales en los sistemas CAD y MCAD es el módulo **Drawing**, este es uno de los requerimientos principales para que la industria pueda asimilarlos.
- También permitió analizar 9 funcionalidades para realizar la propuesta de solución.

# CAPÍTULO 2

## Descripción de la propuesta de solución

Tomando como base las inferencias de información del primer capítulo y la evaluación de la disponibilidad de los sistemas que existen, se identifican los requerimientos de software y los artefactos generados para darle solución al problema planteado en la investigación.

## 2.1. Herramientas para el desarrollo

Para el desarrollo de software según las necesidades y condiciones se eligen las herramientas que mejor respondan a las mismas. En este apartado se describen las herramientas empleadas en el proceso productivo, estas fueron seleccionadas siguiendo el paradigma de la aplicación ingeniero garantizando una correcta integración.
Para el desarrollo de la propuesta de solución se seleccionaron herramientas teniendo en cuentas las empleadas en la aplicación a la cual posteriormente integraremos la solución, además de ser las utilizadas por el proyecto. Se elige el uso del lenguaje de programación C++ versión 11, framework de desarrollo QTCreator versión 5.9.5 y como tecnología para el modelado Open Cascade porque son herramientas de código abierto y están desarrolladas en el mismo lenguaje, facilitando la relación entre variables. Como herramienta de modelado se utilizó Visual Paradigm versión 8.0, facilitando la creación de artefactos ingenieriles guiado por la metodología seleccionada para garantizar una correcta implementación del módulo; y también Git como herramienta de control de versiones. También fue necesario la selección de una metodología de desarrollo, la cual se presenta continuación.

## 2.2. Metodología de desarrollo

La metodología para el desarrollo de software es un conjunto de reglas y pasos sistemáticos que persiguen realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito.

### 2.2.1. AUP-UCI

Al no existir una metodología general que se adapte a los factores externos e internos de cualquier proyecto (equipo de desarrollo, recursos, cliente, etc) por parte de los directivos de la UCI se definió una variación de la metodología **AUP** *(Agile Unified Process)* que se ajuste al ciclo productivo definido para la actividad productiva de la institución. De las 4 fases que propone AUP, las cuales son **Inicio**, **Elaboración**, **Construcción** y **Transición**, se decide mantener la fase de **Inicio** modificando su objetivo, el resto son unificadas en la fase llamada **Ejecución**, y se añade la fase de **Cierre**.
(Rodriguez, 2015) La variante **AUP-UCI** define cuatro escenarios en los que se puede ubicar el desarrollo de una aplicación de acuerdo a un conjunto de características y a las exigencias de nuestro proyecto el que más se ajusta es el escenario 4 por las características que se describen a continuación .
**Escenario 4:** El negocio está bien definido. El cliente siempre estará acompañando al equipo e desarrollo para convenir en detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda para proyectos no muy extensos.

## 2.3. Mapa Conceptual

Los mapas conceptuales son uno de los tipos de recursos esquemáticos que mayor aplicación tiene en los procesos de creación, representación y adquisición de la información y el conocimiento; útil para el desarrollo del aprendizaje de los estudiantes. Desarrollado por Novak en la Universidad de Cornell en Estados Unidos. (Ecured, 2018) Esto se utilizó para relacionar conceptos vinculados a la propuesta de solución. Permitiendo un mejor entendimiento de los conceptos y sus relaciones.

En la figura 2.1 se presenta el mapa conceptual definido en la presente investigación: El módulo drawing crea un dibujo técnico, el cual posee bases geométricas y normas internacionales para su realización. También tiene una plantilla con el objetivo de contener vistas y proyecciones, las mismas poseen propiedades como escala, cotas, tipos, y también posibilitan añadir tablas, símbolos, imágenes, anotaciones.

## 2.4. Descripción del componente

Las funcionalidades serán diseñadas para la gestionar la documentación de un proyecto en una aplicación para el diseño asistido por computadora. Estas estarán integrada a la aplicación CAD que esta siendo desarrollada por el grupo de investigaciones SIPII, la cual tendrá una sesión de trabajo destinada al módulo ***Drawing***. Al activarse el mismo se mostraran una serie de botones que tendrán asociados los algoritmos necesarios para ejecutar las funcionalidades correspondientes. Se anuncian a continuación los botones asociados a las funcionalidades.

- El botón New que permite abrir una plantilla de formato .SVG sobre la cual se realizaran las proyecciones y demás operaciones que se deseen realizar.

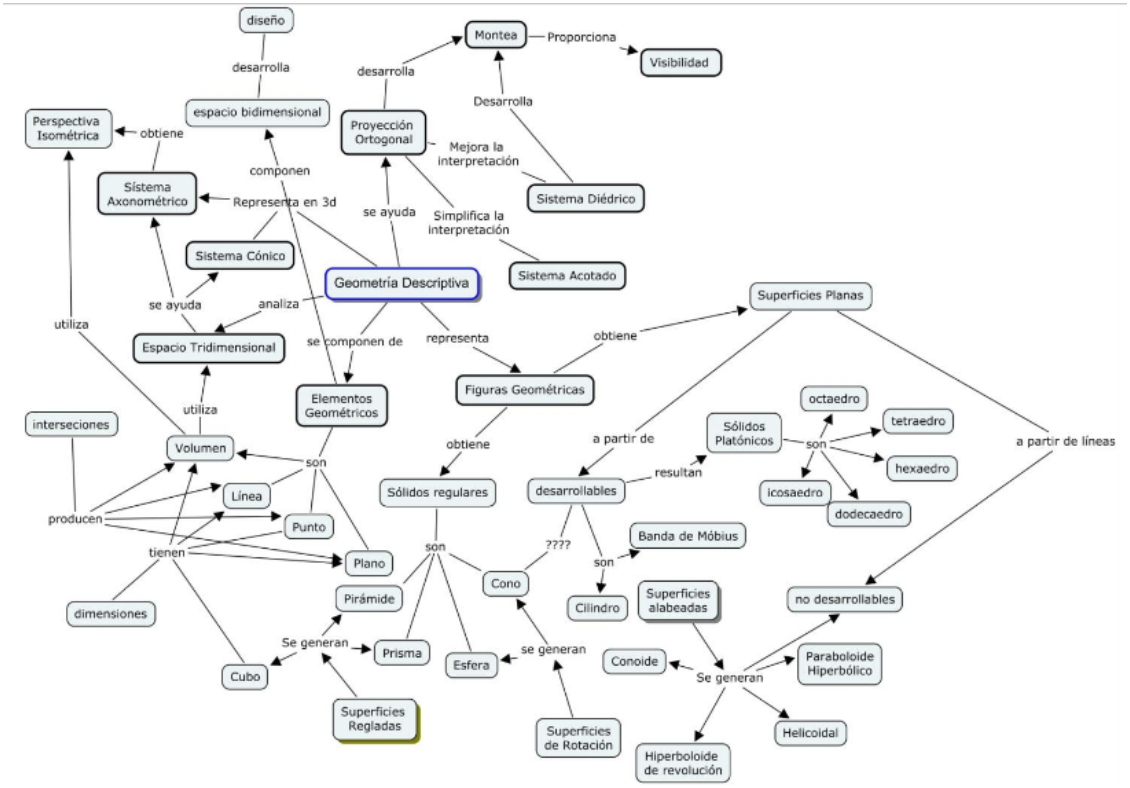



Figura 2.1. Mapa Conceptual

- El botón Open que permite cambiar la norma y estilo de la plantilla sobre la que se desea trabajar, existen algunas por defecto pero de quere incluirla solo debería buscar la dirección en la máquina.
- El botón Anotación que te permite insertar una anotación en la plantilla para dejar algún dato deseado.
- El botón symbol que permite insertar un símbolo en la plantilla dependiendo de la rama de ingeniería para la que lo este utilizando.
- El botón sheet que permite insertar una tabla en la plantilla insertando el número de columnas y filas deseada por el usuario con el objetivo de documentar algún parámetro.
- El botón image que permite insertar una imagen en la plantilla con el objetivo de documentar algún parámetro.
- El botón View que proyecta la vista base de la figura sobre la plantilla ejecutada, especificando la escala de proyección y las coordenadas.
- El botón ProyectView que permite hacer distintas proyecciones de la figura sin ir al modelo solo arrastrando la vista base hacia la dirección deseada.
- El botón SectionView que permite hacer la proyección de una vista con un corte de 1/4 de la figura conocido como half section view o de 1/2 de la figura también conocido como full section view y también se ofrece la oportunidad de hacer proyecciones de las mismas.
- El botón help donde se muestra un dialogo con toda la información necesaria para poder operar con

el módulo en caso de ser un principiante usando estos sistemas.

- En el menú del click derecho se encuentra la funcionalidad salvar que permite salvar todos los datos agregados a la plantilla.
- En el menú del click derecho se encuentra la funcionalidad eliminar que permite eliminar una vista o proyección.
- En el menú del click derecho se encuentra la funcionalidad actualizar que permite hacer una modificación al modelo y actualizar las vistas.
- En el menú del click derecho se encuentra la funcionalidad detalle que permite Obtener un detalle de las vistas en una escala mayor.
- En el menú del click derecho se encuentra la funcionalidad pintar utilizada para fijar la vista cuando no se desee modificar más.

## 2.5. Requisitos

"Los requisitos son clasificados como funcionales y no funcionales; los funcionales responden a las responsabilidades o tareas que la aplicación debe ejecutar a partir de su interacción con el usuario, son las funcionalidades que brinda el sistema. Los no funcionales son derivados de restricciones en el diseño y características que debe cumplir la aplicación, están relacionados con condiciones específicas del entorno de desarrollo, estas características responden directamente a atributos de calidad de software". (Sommerville, 2006)

### 2.5.1. Requisitos funcionales

A continuación, se enumeran los requisitos funcionales definidos para el módulo:

Tabla 2.1. Requisitos Funcionales

| Nro | Requisito Funcional |
|---|---|
| RF 1 | Importar modelo de una plantilla a la aplicación. |
| RF 2 | Proyectar vista base de un modelo seleccionando la escala y ubicación. |
| RF 3 | Realizar las vistas proyectadas. |
| RF 4 | Realizar las vistas de secciones. |
| RF 5 | Realizar la vista de detalle. |
| RF 6 | Actualizar las vistas. |
| RF 7 | Eliminar una vista una vez proyectada. |
| RF 8 | Mover la vista una vez proyectada. |
| RF 9 | Salvar en formato SVG. |

Más adelante se describen los requisitos mediante las historias de usuario; estas contienen datos relacionados con el proceso de desarrollo e implementación de cada requisito funcional bajo el enfoque metodoló-

gico seleccionado.

### 2.5.2. Requisitos no funcionales

La forma de expresar los requisitos no funcionales es la empleada en la metodología Agile Unified Process (AUP, por sus siglas en inglés)-UCI:

**Software:**

SO.: GNU-Linux, GCC: 4.2.4

**Hardware:**

RAM: 1 Gb, Espacio libre en disco: 1 Gb.

**Usabilidad:**

Comprensibilidad: el módulo debe ser de fácil uso con una interfaz amigable.

### 2.5.3. Historias de Usuario

Las historias de usuario son tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento se pueden modificar, reemplazar por otras más específicas o generales o añadirse nuevas. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla.(JEFFRIES, 2001)

Tabla 2.2. Historia de Usuario Proyectar vista base de un modelo

| Número: 2 | Nombre del requisito: Proyectar vista base de un modelo |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1ra |
| Prioridad: Alta | Tiempo Estimado: 3 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 2, 5 semanas |
| **Descripción:** Esta funcionalidad se utiliza para proyectar vista base de un modelo o vista principal.<br>**1- Objetivo:**<br>A partir del objeto tridimensional modelado proyectar sobre la plantilla una vista base o principal.<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para proyectar una vista base del modelo en la aplicación se debe tener en cuenta los siguientes aspectos:<br>- Estar ubicado en el módulo drawing<br>- Seleccionar el botón View<br>**3- Flujo de la acción a realizar:**<br>- En el diálogo correspondiente seleccionar la escala que desea usar.<br>- La posición sobre la plantilla donde se ubicará.<br>- Se oprime el botón aceptar.<br>- Si se selecciona el botón Cancel se cierra la ventana. | |
| Observaciones: | |

**Prototipo de interfaz:**

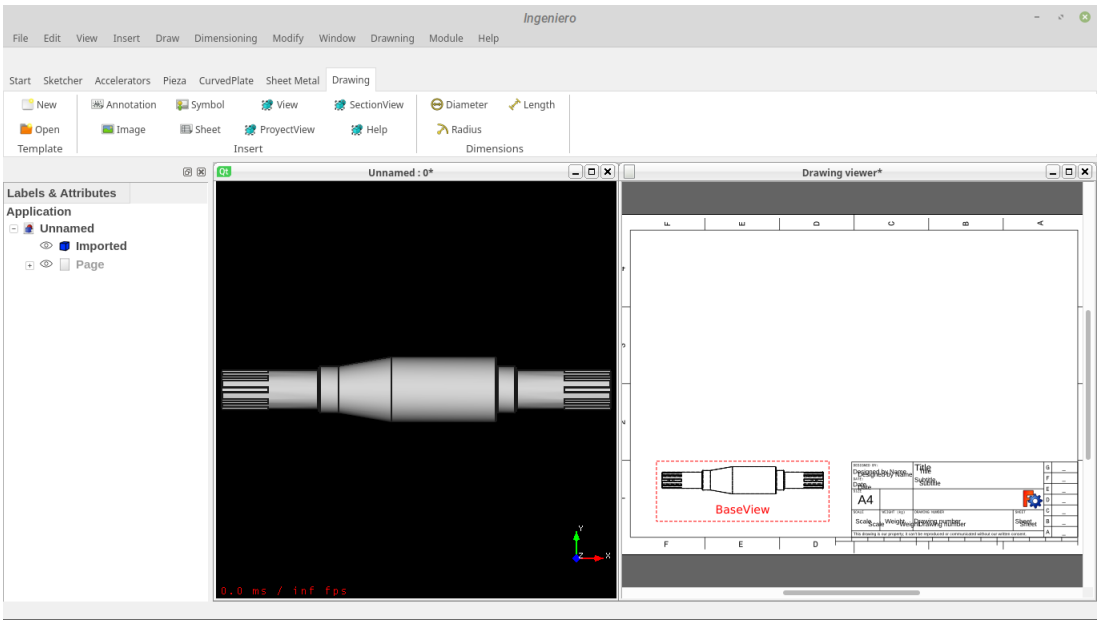



Tabla 2.3. Historia de Usuario Proyectar vista base de un modelo

| Número: 2 | Nombre del requisito: Proyectar vista base de un modelo |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1era |
| Prioridad: Alta | Tiempo Estimado: 2 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 1, 5 semanas |
| **Descripción:**<br>**1- Objetivo:**<br>Permitir proyectar vista base de un modelo<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para proyectar vista base de un modelo se debe tener en cuenta los siguientes datos:<br>- Tener abierta una plantilla<br>- Tener el modelo en el visor tridimensional<br>**3- Flujo de la acción a realizar:**<br>- Se marca el objeto que se desea representar<br>- Se presiona view y se selecciona la escala. | |
| Observaciones: | |
| **Prototipo de interfaz:** | |


### 2.5.4. Diseño de software

"La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa esta organización lógica como un modelo en un lenguaje definido como UML y otras veces simplemente se utiliza notaciones informales y esbozos para representar el diseño ". (monografias.com, 2018) El proceso de diseño tiene asociado la decisión del tipo arquitectura y los patrones de diseño que empleará el sistema, así como la confección de distintos diagramas que favorezcan el trabajo en la fase de implementación. Según (Pressman, 2003) el diseño debe comenzar "por lo grande"centrándose en la arquitectura general y terminar en "lo pequeño"ateniéndose a cada componente.

### 2.5.5. Estilo y patrón arquitectónico

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En caso de que una arquitectura existente se vaya a someter a reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una reasignación de la funcionalidad de los componentes.En

cambio los patrones arquitectónicos aplicados al software, definen un enfoque específico para el manejo de alguna característica de comportamiento del sistema. (Pressman, 2003)

### 2.5.6. Estilo arquitectónico del software

La arquitectura de llamada y retorno es un estilo arquitectónico que permite a un diseñador de software obtener una estructura de programa que resulte relativamente fácil modificar y cambiar de tamaño. En esta categoría hay dos subestilos. (ibíd.)

- Arquitectura de programa principal/subprograma. Esta estructura de programa clásica separa la función en una jerarquía de control donde un programa principal invoca a varios componentes de programa, que a su vez pueden invocar a otros componentes. En la figura 2.2 se ilustra una arquitectura de este tipo.
- Arquitectura de llamada de procedimiento remoto. Los componentes de una arquitectura de programa principal/subprograma se distribuyen entre varias computadoras de una red. El módulo pertenece a un sistema que contiene un programa principal donde se invocan componentes para realizar las funcionalidades que necesita el usuario.

Por tanto el estilo arquitectónico apropiado para el módulo es llamada y retorno, en la categoría de arquitectura de programa principal/subprograma como se muestra en la siguiente imagen.

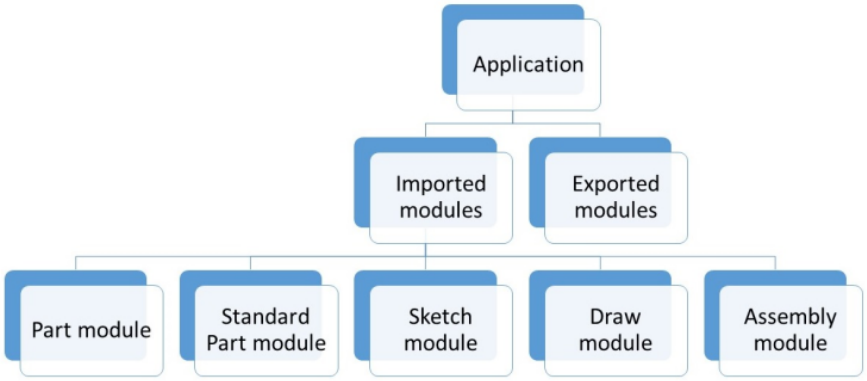



Figura 2.2. Estilo arquitectónico

### 2.5.7. Arquitectura basada en capas

La arquitectura basada en capas ayuda a las aplicaciones a distribuir sus componentes a partir de niveles de abstracción acorde a sus subtareas. Cada capa ofrece servicios a la superior y a su vez se sirve de los que ofrece la inferior. Este modelo es altamente flexible debido a que los niveles de abstracción pueden ser definidos acorde a las necesidades estructurales del sistema, generalmente son definidas cuando más hasta 4 capas, en la mayoría de los casos se emplean solo 3. Una desventaja de este modelo es que el número de

capas puede influir en el rendimiento (cuando son muchas) o provocar múltiples dependencias entre componentes innecesariamente (cuando son pocas). (Sommerville, 2007).
En la figura 2.3 se muestran las “capas”del software: El “núcleo”alberga las clases bases y bibliotecas esenciales, esta capa presta servicios a los componentes contenidos en la capa de “Aplicación”(App); quea su vez manejan la lógica de cada módulo y brindan a la capa de visualización e interacción con el usuario (“GUI”: *Graphic User Interface*) los datos que sean necesarios. La capa de “interfaz de usuario”es, por regla general, la que cambia con mayor frecuencia.
Cada una de estas capas poseen un nivel de impacto a los cambios; un cambio en la estructura del núcleo (añadir,eliminar o modificar una clase perteneciente a la capa) provoca numerosos cambios en la para la capa superior; un cambio en la capa aplicación generalmente provoca cambios en un solo módulo a excepción de las funcionalidades que vinculen más de uno y un cambio en la capa Interfaz solo provoca cambios en la misma.
Cuando se añade alguna funcionalidad se debe analizar cuan profundos son los cambios para satisfacerlo, según el nivel donde se desee hacer la modificación. Esto permite pensar en la distribución de los componentes y módulos teniendo en cuenta la sensibilidad para futuras variaciones funcionales o procesos de reingeniería.

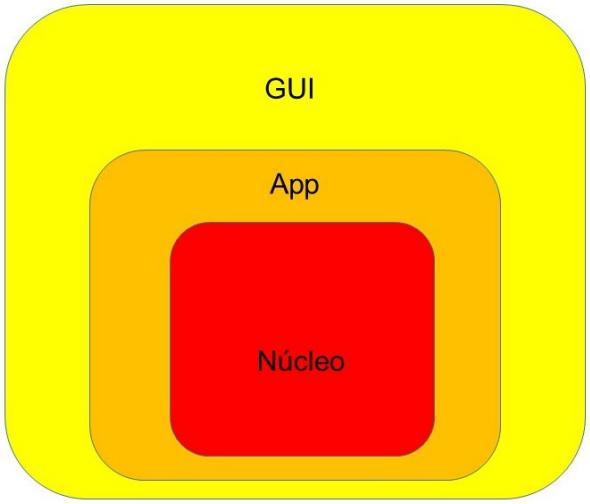



Figura 2.3. Arquitectura por capa

### 2.5.8. Patrones de diseño

Los equipos de desarrollo se han encontrado con la necesidad de diseñar sistemas que puedan acomodarse fácilmente ante cambios estructurales y funcionales; cumplir con buenas prácticas de programación como lo son la herencia y el polimorfismo no son suficientes para obtener aplicaciones fáciles de mantener, escalable, robustas y flexibles. Por esta razón existen los patrones de diseño, estrategias que lidian con

determinados problemas inherentes a subtareas específicas, este conocimiento ha sido obtenido gracias a la experiencia acumulada por múltiples equipos de desarrollo que han aportado al mundo de la ingeniería de software posibles soluciones, cada una con ventajas y debilidades, ante problemas comunes o recurrentes. Otro aspecto útil en el uso de patrones es que brinda una manera eficiente de documentar y comunicar la intención del código al resto del equipo o a futuros ingenieros y técnicos encargados del mantenimiento. (Larman, 2003)

**Los patrones General Responsibility Asignment Software Patterns (GRASP, por sus siglas en inglés)** , su utilidad viene a la hora de caracterizar objetos y clases. Saber dónde ubicar una responsabilidad (un servicio), quién es responsable de crear objetos, manejar eventos, etc. Definidos en las funcionalidades:

- **Experto:** mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidenciará en la clase "DrwaingViewPart", pues esta poseerá un objeto de la clase "DrawingView"para poder acceder a la información y funcionalidades de los modelos activos en el visor tridimensional.
- **Creador:** permite crear objetos de una clase determinada. Se utilizará en algunas de las clases de interfaz para crear instancias de formularios y entidades.
- **Polimorfismo:** se emplea para asignar comportamientos distintos según el tipo de clase. Se evidenciará en las clases "viewProviderImage", "viewProviderSheet"y "viewProviderSymbol", que heredarán de la clase abstracta "viewProvider"y definirán el comportamiento de la funcionalidad "makeProvider".
- **Alta cohesión:** este patrón caracteriza a las clases que posean responsabilidades estrechamente relacionadas, es decir, que no realicen un trabajo enorme. Con el objetivo de que las clases "escalaDlg "y "vistaProyectadalDlg "no realizaran un trabajo enorme y poder reutilizar código.

Se especifican 23 patrones que son categorizados según el tipo de problemas a los que responden, estas categorías son patrones de creación, estructurales y de comportamiento. GOF(Gang of Four) es el nombre con que son denominados este conjunto de patrones.

**Ejemplo de patrones GOF definido en las funcionalidades:**

**Algunas características de los patrones GOF**

- Son soluciones concretas. Proponen soluciones a problemas concretos, no son teorías genéricas.
- Son soluciones técnicas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- Se utilizan en situaciones frecuentes. Debido a que se basan en la experiencia acumulada al resolver problemas reiterativos.
- Ayudan a construir software basado en la reutilización de código y clases. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

(Ecured, 2019)

- **Observador :**es un patrón que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Este patrón se evidencia en la función mustExecute.
- **Visitor (Visitante):**Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera este se evidencia en la clase ViewProvider.
- **Adapter (Adaptador):** Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla esto se implementa en los diálogos y ventanas del módulo.
- **Singleton (Instancia única):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia este se utiliza para poder acceder al modelo tridimensional y obtener los datos desde el punto de vista del observador y lograr las proyecciones.

(Microsoft, 2017)

### 2.5.9. Diagrama de clases del diseño

El modelo del diseño se utiliza como método de abstracción de la implementación y el código. Se utiliza como entrada esencial para actividades como la implementación y prueba se utiliza para concebir y documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.(ResearchGate, 2018)

### 2.5.10. Conclusiones parciales

Una vez analizados los elementos de diseño y técnicos de la propuesta de solución se arriba a las siguientes conclusiones:

- Nuestra propuesta es viable por que se usa una arquitectura semejante a la aplicada en software FreeCAD, de donde se reutilizara la arquitectura base, lo que permite una migración de clases rápida ademas de un funcionamiento similar al de FreeCAD.
- La propuesta de solución permite proseguir con el desarrollo del módulo Drawing para posteriores investigaciones con un modelado escalable de la solución.

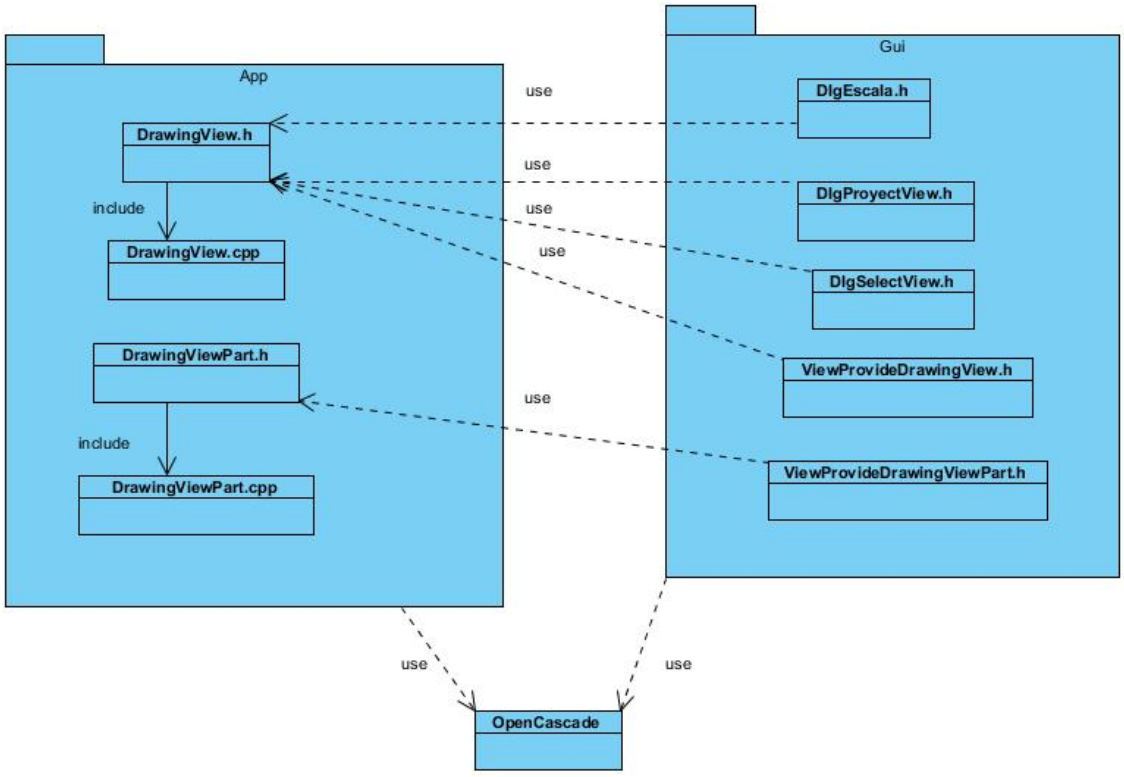



Figura 2.4. Mapa Conceptual

# CAPÍTULO 3

## Implementación y prueba

En este capítulo se realiza una evaluación de la solución obtenida durante el proceso de implementación y pruebas. Se abordan aspectos importantes asociados a los estándares de codificación, utilización de patrones de diseño. Además se evidencian las pruebas realizadas a la propuesta de solución para garantizar la calidad del componente.

## 3.1. Implementación

La fase de implementación del software posee como entradas los artefactos de la fase anterior (diseño), como: diagramas de clases, especificación de arquitectura, patrones de diseño, entre otros. En la implementación se define el estándar de codificación a emplear, se realizan las implementaciones a las historias de usuarios, se define el diagrama de componentes del sistema, entre otras actividades.

### 3.1.1. Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código, este se debe seleccionar de forma prudente. Un código debe reflejar un estilo armonioso en toda su integridad, como si un único programador hubiera escrito todo el código de una sola vez, aunque la forma de escribir código es propia de cada programador. La selección del estándar debe depender de la facilidad para entender el código teniendo en cuenta que este sera empleado por todas las personas del mismo equipo de desarrollo. Para esta solución se selecciono el estándar CamelCase. (ASPL, 2018)

| Descripción | Ejemplo |
|---|---|
| **Definición de Objetos, Clases, funciones y atributos** | |
| Todos los nombres de las clases implementadas comenzarán con letra mayúscula. En caso de poseer un nombre compuesto se escribirán de acuerdo a la normativa CamelCase-UpperCamelCase. | class Foo{<br>cuerpo de la clase<br>}<br>class FooFirst{<br>cuerpo de la clase<br>} |
| Siempre se declara para todas las clases implementadas su respectivo destructor de clase. | virtual ~Foo() |
| La declaración de funciones o métodos siempre comenzarán en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase. | <Tipo dato retorno> funcion()<br><Tipo dato retorno> funcionCompuesta()<br><Tipo dato retorno>funcionDobleCompuesta() |

Figura 3.1. Estándar de codificación

| | |
|---|---|
| Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase. | <Tipo dato> atributo;<br><Tipo dato> atributoNombreCompuesto; |
| **Definición de parámetros dentro de las funciones y constructores de clases** | |
| Los nombres de los identificadores de los parámetros en las funciones deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCaselowerCamelCase. | <Tipo dato retorno> funcion(<tipo><id1>, <tipo><id2>, <tipo><idN>) |
| Los identificadores de los parámetros dentro de los constructores de las clases deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCase-lowerCamelCase. | Clase(<tipo><id1>, <tipo><id2>, <tipo><idN>) |
| **Definición de expresiones** | |
| Para una mejor comprensión en la lectura y legibilidad del código los operadores binarios exceptuando los punteros, función de llamado a miembros, escritura de un arreglo y paréntesis de una función se escribirán con un espacio entre ellos | x + y;<br>x == y;<br>idFuncion.miembro();<br>idFuncion->miembro();<br>array[]; |

Figura 3.2. Estándar de codificación

| **Definición de estructuras de control y bucles** | |
|---|---|
| Las estructuras de control y los bucles estarán definidos de igual manera en ambos casos siguiendo el estándar determinado por el *framework* de Qt. | Para las estructuras if, else, if else :<br><estructura control>(condición){<br>tarea a ejecutar<br>}<br>Para los bucles while, for, do while y otros:<br><bucle>(condiciones){<br>tarea a ejecutar<br>} |
| **Comentarios en el código según el estándar de C++.** | |
| Comentarios pequeños. | /* comentario sencillo */ |
| Otros comentarios. | /*<br>*Comentario<br>*/ |
| Comentario de versión, descripción de clase y otras características de la clase o paquete. | /*<br>**********<br>*Comentario amplio *<br>**********<br>*/ |

Figura 3.3. Estándar de codificación

### 3.1.2. Ejemplo del estándar de codificación

En la siguiente imagen se puede observar como se utiliza el estándar de codificación **CamelCase** en la implementación de la solución.

```
class MDIViewPage : public Gui::MDIView
{
    Q_OBJECT
    SelectionObserver_HEADER()

public:
    MDIViewPage(ViewProviderPage *page, App::Document* doc, QWidget* parent = 0);
    virtual ~MDIViewPage();

    /// Observer message from the Selection
    void onSelectionChanged(const Gui::SelectionChanges& msg);
    void preSelectionChanged(const QPoint &pos);
    void selectFeature(App::DocumentObject *obj, bool state);
    void clearSelection();
    void blockSelection(bool isBlocked);
```

Figura 3.4. Ejemplo del estándar de codificación

### 3.1.3. Diagrama de componente

Un diagrama de componentes permite observar las relaciones entre los componentes y ayuda al equipo a entender el diseño existente, los mismos no son más que una unidad física de implementación con interfaces bien definidas pensada para ser utilizada como parte reemplazable de un sistema. Puede mostrar un sistema configurado, con la selección de componentes usados para construirlo o un conjunto de componentes disponibles (una biblioteca de componentes) con sus dependencias. (Jacobson; Rumbaugh y Booch, 2000). A continuación se muestra el diagrama de componente de la solución general.

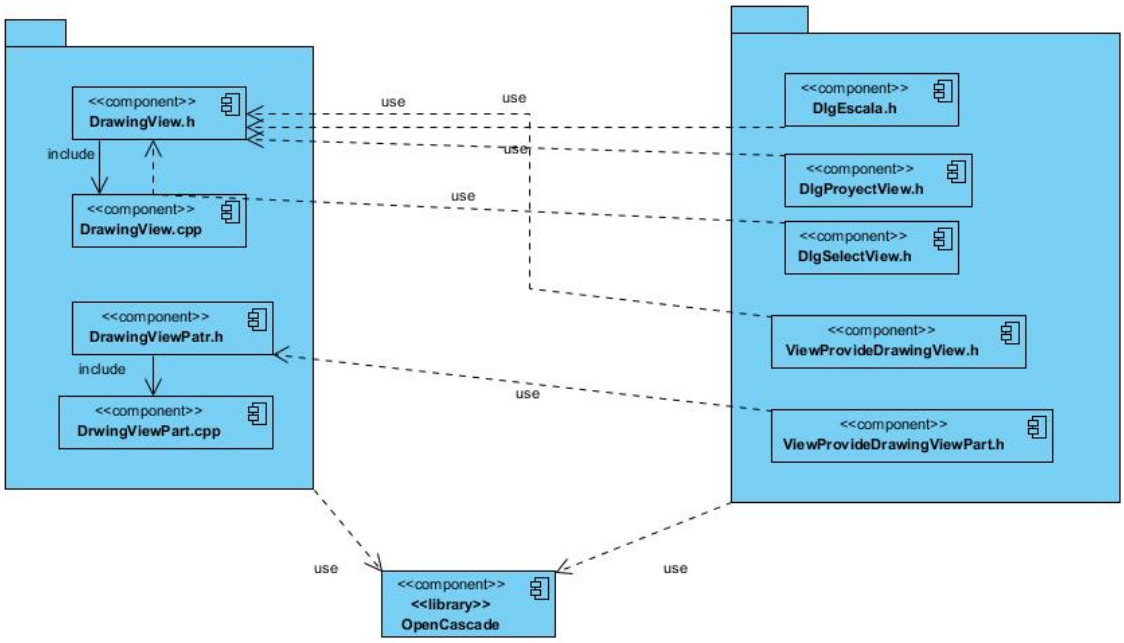



Figura 3.5. Diagrama de componente

### 3.1.4. Detalles técnicos de la implementación

En la implementación se utilizaron varias funcionalidades agrupadas en la biblioteca gráfica OpenCascade como las que se muestran a continuación.

- **gp-Ax2d:** Crea un eje representando en el eje X un objeto de referencia en el sistema de coordenadas.
- **gp-Trsf:** Función que brinda la posibilidad de realizar transformaciones en el espacio tridimensional como traslación o rotación.
- **gp-Pnt2d:** Crea un punto en dos dimensiones.
- **GCE2d-MakeSegment:** Crea un segmento entre dos puntos en dos dimensiones.
- **TopoDS-Wire:** Topología presente en las biblioteca Open CASCADE.
- **TopoDS-Face:** Topología presente en las biblioteca de Open CASCADE que tiene como objetivo crear un face de un wire cerrado.

- **TopoDS-Shape:** Topología presente en las biblioteca de Open CASCADE, la cual tiene como objetivo hacer un sólido.
- **SetDisplacement:** Función que modifica una transformación y permite crear un nuevo sistema de coordenadas.
- **BRepAlgoAPI-Cut:** Función que realiza una operación boolean para realizar un corte sobre sólidos.

### 3.1.5. Resultados de la implementación

En esta sección se muestran varias imágenes de los resultados obtenidos, a través de capturas de pantallas.

Estas son algunas de las funcionalidades implementadas en la propuesta de solución.

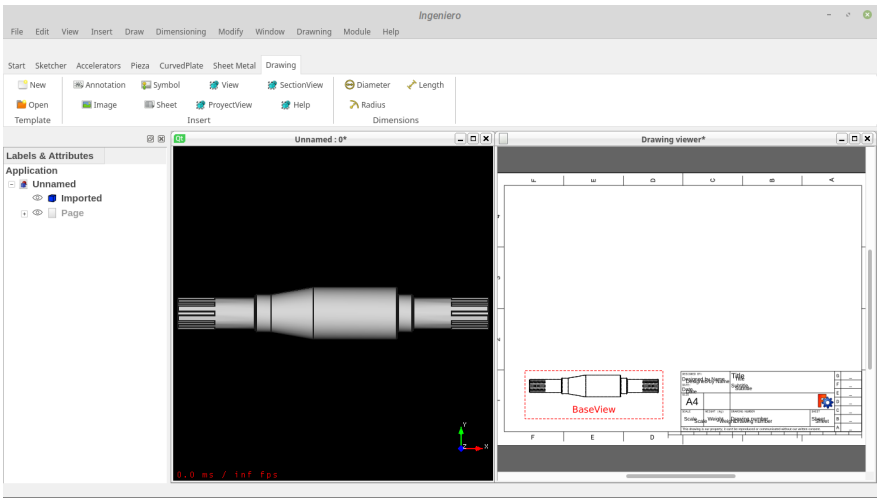



Figura 3.6. Funcionalidad view

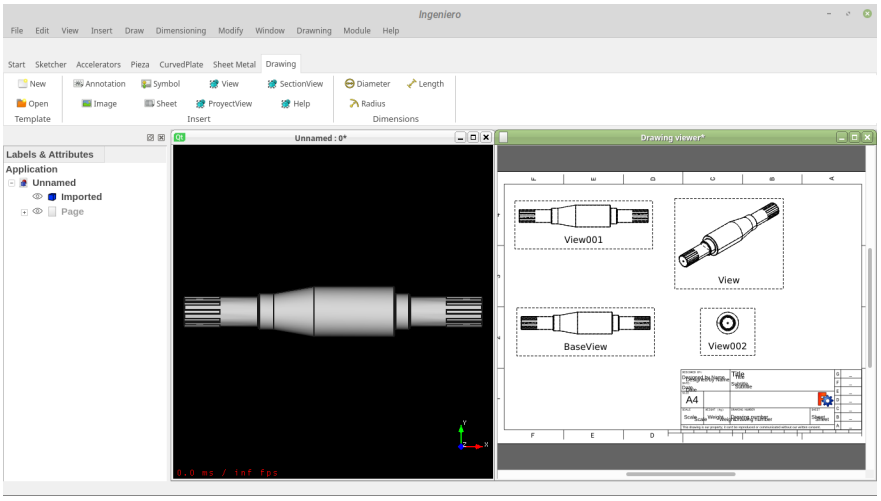



Figura 3.7. Funcionalidad Proyect View

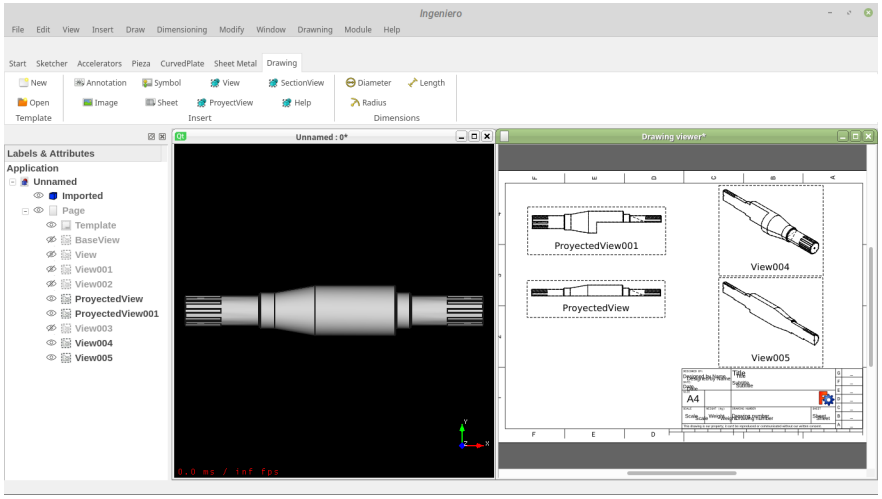



Figura 3.8. Funcionalidad Section View

## 3.2. Validación y pruebas

En la ingeniería de software la validación es el proceso que se realiza para demostrar que el software cumple con los requerimientos establecidos realizando al menos una prueba a cada uno de ellos y que logra su objetivo, se evalúa el sistema o parte de este, durante o al final del desarrollo para determinar si satisface los requisitos iniciales.
La estrategia de prueba a seguir posee un enfoque incremental, inicia con las pruebas de unidades individuales del programa, pasa a pruebas diseñadas para facilitar la integración de las unidades, y culmina con las pruebas que se realizan sobre el sistema construido. Más adelante se muestran los tipos de pruebas aplicadas a la solución. (Pressman, 2003)

### 3.2.1. Pruebas unitarias

Las pruebas unitarias se le realizan a cada funcionalidad de los componentes para asegurar que cada uno de estos funcionan correctamente por separado. Al aplicarlas los errores son más fáciles de detectar debido a que la cantidad de código no es excesivamente grande, esto permite llegar a la fase de integración con seguridad del correcto funcionamiento de cada uno de los componentes. (Sommerville, 2006)

### 3.2.2. Método de caja blanca

Este método se emplea para revisar los detalles procedimentales. Se prueban las rutas lógicas del código y las relaciones entre componentes, al diseñar casos de pruebas que ejecuten condiciones, bucles o ambos. (Pressman, 2003)
Al emplear este método el ingeniero deberá diseñar casos de pruebas que:

- Garantice que todas las rutas independientes del código se ejecuten al menos una vez.
- Comprueben los casos verdaderos y falsos de todas las condiciones lógicas.
- Ejecuten todos los bucles dentro de sus limites operacionales.
- Ejerciten estructuras de datos internos para comprobar su validez.

En (Pressman, 2003) queda definido como debe realizarse este proceso, a continuación se presenta la descripción del mismo, utilizando la técnica de flujo básico, esta permite conocer las complejidades ciclomáticas de un bloques de código.

**Notación del grafo de flujo:** tomando como base el código se representa un grafo donde los nodos representan una o más sentencias procedimentales y las aristas son el flujo de control.

**Complejidad ciclomática:** es una métrica que nos permite conocer una cantidad cuantitativa de un programa. El valor define la cantidad de caminos independientes y propone que se realice al menos un caso de prueba por cada uno de estos, se calcula de tres formas:

- Número de regiones del grafo de flujo.
- V(G)= A-N+2, donde A es la cantidad de aristas del grafo y N es el número de nodos del grafo.
- V(G)= P+1 donde P es la cantidad de nodos predicados del grafo.

**Determinar un conjunto básico de caminos linealmente independientes:** el valor de V (G) coincide con el número de caminos linealmente independientes de la estructura de control del programa.

**Obtención de casos de prueba:** se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

Para la realización de estas pruebas se utilizó la herramienta QTest que permite la ejecución desde el ***framework*** Qtcreator arrojando los siguientes resultados.

```
********* Start testing of TestUnitTest*********
Config: Using QTest library 5.5.1, Qt 5.5.1 (x86_64l
PASS   : TestUnitTest::initTestCase()
PASS   : TestUnitTest::testDrawView()
PASS   : TestUnitTest::testDrawProyectView()
PASS   : TestUnitTest::testEscala()
PASS   : TestUnitTest::testSectionView()
PASS   : TestUnitTest::testDetailView()
PASS   : TestUnitTest::testDeleteView()
PASS   : TestUnitTest::testUpdateView()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 9 passed, 0 failed, 0 skipped, 0 blacklisted
********* Finished testing of TestUnitTest *********
```

Figura 3.9. Resultado de las pruebas con QTest

### 3.2.3. Método de caja negra

Las pruebas de caja negra se realizan para comprobar el correcto funcionamiento las Interfaz de usuario (IU) del sistema, permiten al ingeniero de software aplicar conjuntos de condiciones de entrada que ejercitarán por completo todas las IU de un programa.
Según la literatura citada anteriormente este método trata de encontrar errores en las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en la estructura de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Según (Pressman, 2003) para desarrollar las pruebas de caja negra existen varias técnicas, entre las cuales se encuentran :

- **Técnica de la partición de equivalencia:**divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del análisis de valores límites:**prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de grafos de causa-efecto:**permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de caja negra la técnica de la partición de equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

Para verificar el correcto funcionamiento del componente desarrollado se realizaron pruebas funcionales, por cada requisito funcional.

### 3.2.4. Diseño de caso de prueba para caja negra

Los casos de prueba incluyen todas las funciones que el programa es capaz de realizar. Deben tener en cuenta el uso de todo tipo de datos de entrada/salida , cada comportamiento esperado , todos los elementos de diseño, y cada clase de defecto. Todos los requisitos deberán ser cubiertos por los casos de prueba, de manera tal que cubra el software a fondo. Se puede decir que son la descripción de las actividades que se van a ejecutar con el fin de validar la aplicación. (Duarte y Valle, 2018)

- Se usan las mismas técnicas, pero con otro objetivo.

- No hay programas de prueba, sino sólo el código final de la aplicación.
- Se prueba el programa completo.
- Uno o varios casos de prueba por cada requisito.
- Se prueba también rendimiento, capacidad, etc. (y no sólo resultados correctos).
- Pruebas alfa(desarrolladores) y beta (usuarios).

Tabla 3.1. Diseño de Casos de Prueba RF 1

| Descripción general | | | |
|---|---|---|---|
| Permitir importar un modelo de plantilla a la aplicación | | | |
| SC 1 Importar modelo de plantilla | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Importar un modelo | Selecciona el botón Open el cual mostrará un diálogo para seleccionar la ubicación del modelo que se desea importar. | Permite cargar cualquier en cualquier norma de dibujo. | DrawPage. /Drawing |
| EC 1.2 Opción de Close | Selecciona la opción de Close | Cierra la ventana. | DrawPage/Drawing /Close |

### 3.2.5. No conformidades obtenidas

| **No. NC** | **Requisito Funcional** | **Descripción** | **Complejidad** | **Estado** |
|---|---|---|---|---|
| 1 | RF. 2 | Localización de la vista base sobre la plantilla. | Alta | Resuelta |
| 2 | RF. 2 | No se modifica la escala de la plantilla. | Media | Resuelta |
| 3 | RF. 6 | No se actualiza la vista con la misma escala. | Media | Resuelta |
| 4 | RF. 3 | El sistema no permite proyectar más de una vista en la plantilla. | Media | Resuelta |
| 5 | RF. 5 | La vista de detalle no imprime nada. | Alta | Resuelta. |
| 6 | RF. 7 | No borra la instancia de cada item view del árbol de operaciones | Alta | Resuelta |
| 7 | RF. 8 | No permite mover las vistas sobre la plantilla | Baja | Resuelta. |
| 8 | RF. 9 | Salva solo puntos. | Alta | Resuelta |
| 9 | RF. 9 | salva la geometría pero no en la ubicación requerida. | Baja | No Resuelta |

Tabla 3.2. No conformidades.

### 3.2.6. Resultado de las pruebas

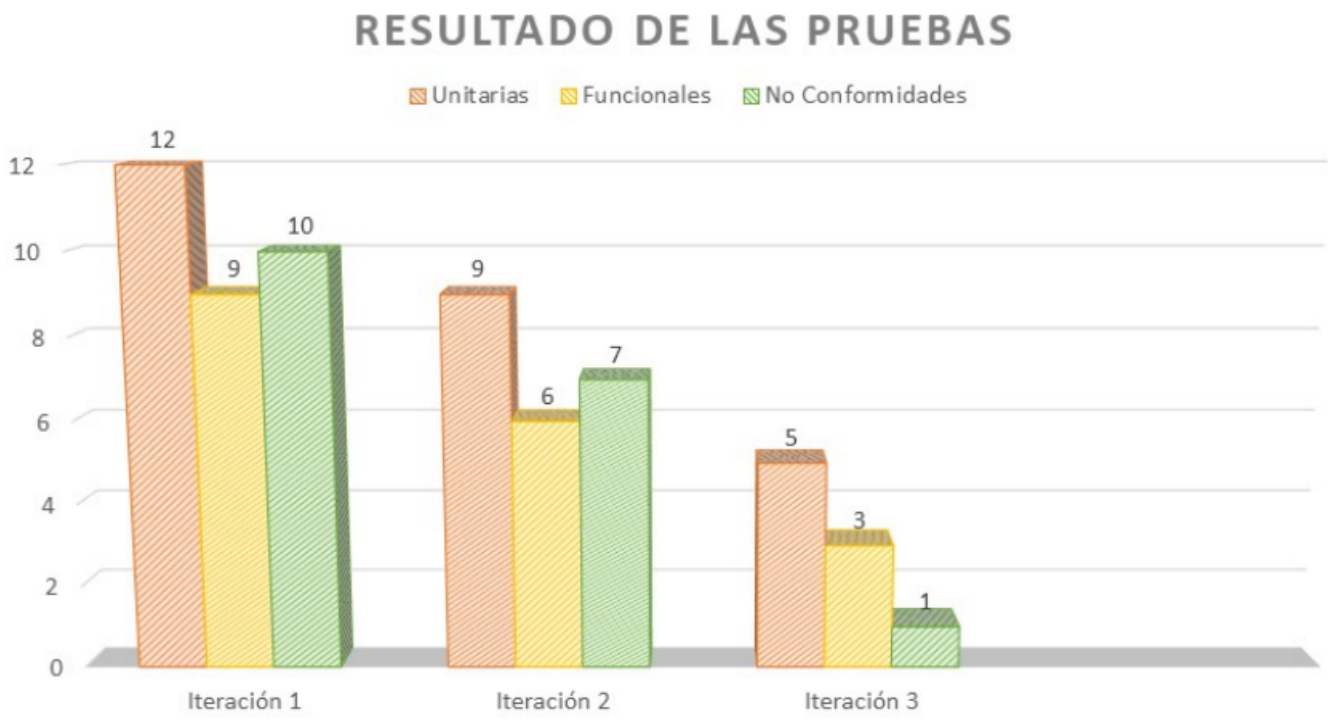



Figura 3.10. Resultados de las pruebas

En la primera iteración se realizaron 12 pruebas unitarias, 9 funcionales arrojando un total de 10 no conformidades. En la segunda iteración se realizaron 9 pruebas unitarias, 6 funcionales arrojando un total de 7 no conformidades. En la tercera iteración se realizaron 5 pruebas unitarias, 3 funcionales obteniendo 1 no conformidad la misma afecta al correcto funcionamiento de 2 requisitos funcionales se denomina como la no obtención de coordenadas sobre la plantilla SVG y afecta en los requisitos 5 y 9.

### 3.2.7. Pruebas de integración

Estas pruebas se realizan para comprobar que los elementos del software interactúan entre sí y funcionan de manera correcta. De esta manera se verifica la correcta interrelación entre los componentes del sistema; en el caso de la solución en cuestión fue integrar las funcionalidades desarrolladas al módulo ***Drawing*** de la aplicación Ingeniero para verificar el correcto funcionamiento de estas en el sistema como un todo.
Un enfoque no incremental como lo es el enfoque “big bang ”, consiste en integrar todos los módulos y una vez juntos probar el sistema esperando la explosión de errores que se puede generar. La integración mediante el ensamblado de todos los componentes del producto a la vez al final de las fases de diseño y construcción se llama “Integración Big Bang ”. Es muy difícil aislar las causas de los errores encontrados en las pruebas si un producto se integra utilizando este enfoque. Además, la “Integración Big Bang ”no puede

comenzar hasta un momento avanzado del proyecto porque requiere que todos los módulos o sub-módulos del producto hayan sido desarrollados y probados de forma unitaria. (Rowsell, 2015)

**Ventajas**

- Útil para la detección de errores cuando se encuentren todos los módulos ya en construcción.
- Aplicable antes de la entrega de proyecto para evaluar el trabajo de todo el sistema con diversos escenarios.
- Se puede evaluar la interacción entre módulos para agilizar los procesos.
- Es apta para aplicar diversos escenarios para poder analizar el trabajo de todos los módulos en diversas situaciones.

**Desventajas**

- Dificultan la detección de posibles errores.
- Solo es posible aplicarla hasta que se esté avanzado el proyecto.
- Se tienen que trabajar con todos los módulos, no se puede hacer un análisis individual.
- Si el proyecto es de corto plazo no se tiene mucho tiempo para la aplicación de la prueba y realizar las correcciones.

Esta operación arrojó un total de 3 errores, a continuación se muestran los resultados obtenidos en cada iteración de prueba, así como la corrección de los mismos.

### 3.2.8. Resultados de la prueba de integración

Tabla 3.3. Datos Técnicos

| Errores | Primera iteración | Segunda iteración | Tercera iteración |
|---|---|---|---|
| Detectados | 3 | 1 | 0 |
| Resueltos | 3 | 1 | 0 |
| Pendientes | 0 | 0 | 0 |

### 3.2.9. Pruebas de aceptación

Una Prueba de Aceptación tiene como propósito demostrar al cliente el cumplimiento de un requisitos del software. Estas se caracterizan por:

- Describir un escenario (secuencias de pasos) de ejecución o uso del sistema desde la perspectiva del cliente.
- Pueden estar asociada a requisitos funcionales o no funcionales.
- Un requisito tiene una o más pruebas de Aceptación asociadas.
- Las Pruebas de Aceptación cubren desde escenarios típicos/frecuentes hasta los más excepcionales.

Las Pruebas de Aceptación se pueden utilizar para

- Obligar a definir requisitos que sean verificables.
- Valorar adecuadamente el esfuerzo asociado a la incorporación de un requisito.
- Negociar con el cliente el alcance del sistema.
- Planificar el desarrollo iterativo e incremental del sistema.
- Guiar a los desarrolladores.
- Identificar oportunidades de reutilización.

todas las características anteriormente descritas extraídas de (Pressman, 2003)

Estas pruebas fueron realizadas al DrC. Augusto Cesar Rodrigues Medina debido a que es un usuario potencial de este tipo de sistemas. El criterio del experto en el tema fue positivo mostrando aceptación de la solución.

### 3.2.10. Conclusiones parciales

Una vez llevado a cabo el proceso de validación de los resultados obtenidos se arriba a las siguientes conclusiones parciales: Como resultado del proceso de implementacion se implementaron las siguientes funcionalidades:

- Se implementó la vista de sección(**Section View**) que permite ver la conformación interior de los modelos, que no está implementada en FreeCAD; esta mayormente se usa en objetos con perforaciones o estructuras interiores.
- Se implementó la vista de detalle(**DetailView**) que se utiliza para ver una parte del objeto de mayor escala, que no está implementada en FreeCAD.
- Se implementó la **vista proyectada** con la reutilización de código de la aplicación FreeCAD con algunas modificaciones.
- Se implemento la funcionalidad de **Salvar** en formato SVG, que no está implementada en la aplicación FreeCAD.
- Se implemento la funcionalidad **Actualizar** que permite redibujar una vista después de haber modificado el modelo, que no está implementada en FreeCAD.
- Se implemento un dialogo para la modificación de la **Escala**, que no lo posee la aplicación FreeCAD.

- Las pruebas realizadas permitieron detectar lograron identificar 9 no conformidades de las cuales 8 fueron resuelta, para garantizar un mejor funcionamiento.

# Conclusiones

Como resultados del presente trabajo se definen como conclusiones generales:

- El resultado del estudio realizado se identificó las carencias que existen en la herramienta **Ingeniero** relacionadas el módulo ***Drawing***, este modulo aunque todavía esta incompleto se le han añadido 9 funcionalidades lo cual constituye un avance en el desarrollo del módulo.
- Como resultado de la implementación de las funcionalidades, SectionView, DetailView, ProyectedView, Scale, Update, Delete, Save el modulo cuenta con 16 funcionalidades y con 6 que no estaban presente en las herramientas de código abierto, para dotar a la aplicación **Ingeniero** las mismas.
- El resultado obtenido propicia un aporte a la soberanía tecnológica nacional.

# Recomendaciones

A partir del estudio realizado y luego de haber analizado los resultados obtenidos se recomiendan los siguientes elementos a tener en cuenta para futuros trabajos:

- Completar la funcionalidades de dimensionar completando el funcionamiento del módulo.
- Lograr definir las áreas de corte con el sombreado correspondiente.
- Lograr mejorar las funcionalidades de detalle e inserción de la vista obteniendo la posición del mouse mediante un método.
- Crear una forma de salvar la plantilla modificada de manera correcta .

# Acrónimos

**API** Aplication Programing Interface. 17

**CAD** Diseño Asistido por Computadoras. 1, 6, 16

**SIPII** Soluciones Informáticas para la Ingeniería y la Industria. 4

**UCI** Universidad de las Ciencias Informáticas. 1

**VLSI** sigla en inglés de very-large-scale integration. 16

# Referencias bibliográficas


ANGEL, Edward y SHREINER, Dave, 2012. *Interctive Computer Graphics* (vid. págs. 12, 13, 15).

ASPL, 2018. *Estandares de Codificacion.* Url: `http%20:%20/%20/%20www%20.%20aspl%20.%20es%20/%20fact%20/%20files%20/%20aspl%20-%20fact%20/%20estandares-node2.html` (vid. pág. 32).

CADALYST, 2004. *Autodesk Inventor 7 and 8.* Url: `http://www.cadalyst.com/cad/product-design/autodesk-inventor-7-and-8-10078` (vid. pág. 8).

DUARTE y VALLE, Maria Elena, 2018. *Casos de prueba y tipos de pruebas.* Url: `https%20:%20/%20/%20prezi%20.%20com%20/%20jcygazqtotrn/casos-de-prueba-y-tipos-de-pruebas/` (vid. pág. 39).

ECURED, 2018. *Mapa Conceptual*. Url: `https://www.ecured.cu/Mapa_conceptual` (vid. pág. 21).

ECURED, 2019. *Patrones Gof.* Url: `https://www.ecured.cu/Patrones_Gof` (vid. pág. 29).

FARLEX, 2019. *CATIA*. Url: `https://encyclopedia2.thefreedictionary.com/CATIA` (vid. pág. 6).

HABRENT, Yorik Van y COMUNITY, FreeCAD, 2015. *FREECAD A MANUAL* (vid. pág. 17).

JACOBSON; RUMBAUGH y BOOCH, G., 2000. *El Lenguaje Unificado de Modelado. Manual de Referencia* (vid. pág. 35).

JEFFRIES R., A. ANDERSON y C. HENDRICKSON, 2001. *Extreme Programming Installed. Ed. por Addison-Wesley* (vid. pág. 24).

KOVACEVIC, Dr Ahmed, 2017. *CATIA V5* (vid. pág. 6).

LARMAN, 2003. *Patrones de diseÃ±o* (vid. pág. 29).

MARTINS, Paula; OLIVEIRA, Catarina; SILVA, Samuel; SILVA, Augusto y TEIXEIRA, AntÃ³*nio*, 2011. *TONGUE SEGMENTATION FROM MRI IMAGES USING ITK-SNAP: PRELIMINARY EVALUATION* (vid. págs. 10, 11, 13, 15).

MCGUIRE, Morgan; HUDGES, Jhon F. y DAM, Adries Van, 2015. *Computer graphics Princples and Practice* (vid. págs. 10, 16).

MICROSOFT, 2017. *Arquitectura y DiseÃ±o* (vid. pág. 30).

MONOGRAFIAS.COM, 2018. *DiseÃ±odesoftware*. Url: https://www.monografias.com/trabajos73/diseno-software/diseno-software.shtml (vid. pág. 26).

PLANCHARD, David, 2012. *Drawing and Detailing with SolidWorks*. Url: https://Drawing%20and%20Detailing%20with%20SolidWorks.com (vid. pág. 7).

PRESSMAN, 2003. *ingenieria de software* (vid. págs. 26, 27, 37-39, 43).

RESEACHGATE, 2014. *API*. Url: https://www.google.com/search?source=hp&ei=7NHcXIWqM67H_QbSro6oCQ&q=API&oq=API&gs_l=psy-ab.12..0l10.30763.33891..35440...0.0..0.395.698.2j1j0j1....2..0....1..gws-wiz.....0..35i39j0i131j35i39i19.5-ni8OjmLV8 (vid. pág. 14).

RESEARCHGATE (ed.), 2015. *GRAPHIC SKILLS ACQUISITION FOR TEACHING AND LEARNING AND CHALLENGES IN TECHNOLOGY EDUCATION* (vid. pág. 11).

RESEARCHGATE, 2018. *Artefacto : Modelo del diseÃ±o*. Url: https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/workproducts/rup_design_model_2830034D.html (vid. pág. 30).

RODRIGUEZ, T. SÃ¡nchez, 2015. *MetodologÃa de desarrollo para la actividad productiva de la UCI* (vid. pág. 21).

ROWSELL, Allan, 2015. *Prueba de softwae Big Bang*. Url: https://modelosdepruebasw.wordpress.com/ (vid. pág. 42).

SNAIDER, Philip J y EBERLY, David, 2016. *Geometric tool for computer graphics* (vid. págs. 12, 15).

SOMMERVILLE, 2006. IngenierÃa de Software. En: *Software Engineering* (vid. págs. 23, 37).

SOMMERVILLE, 2007. *IngenierÃa de software* (vid. pág. 28).

WEISBERG, David, 2008. *Brief History* (vid. pág. 4).

WIKIPEDIA, 2010. *Solid Edge*. Url: https://en.wikipedia.org/wiki/Solid_Edge (vid. pág. 5).

WIKIPEDIA, 2012. *Solid works*. Url: https://en.wikipedia.org/wiki/SolidWorks (vid. pág. 7).

WIKIPEDIA, 2014. *Autodesk Inventor*. Url: https://en.wikipedia.org/wiki/Autodesk_Inventor (vid. pág. 8).

Generado con LaTeX: 29 de junio de 2019: 1:38pm

# Apéndices

# .1. Historias de usuarios

Tabla 4. Historia de Usuario Importar un modelo de plantilla a la aplicación

| Número: 1 | Nombre del requisito: Importar un modelo a la aplicación |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1ra |
| Prioridad: Alta | Tiempo Estimado: 3 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 2, 5 semanas |
| **Descripción:** Esta funcionalidad se utiliza para importar una plantilla nueva o que se desea modificar a la aplicación.<br>**1- Objetivo:**<br>Importar plantillas de trabajo de otro formato y norma o plantillas sobre las cuales se desea seguir trabajando<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para importar un modelo a la aplicación se debe tener en cuenta los siguientes aspectos:<br>- Estar ubicado en el módulo drawing<br>- seleccionar el botón Open<br>**3- Flujo de la acción a realizar:**<br>- En el diálogo correspondiente buscar la ubicación del modelo que usted desea cargar.<br>- Se selecciona el modelo de la plantilla sobre la cual se necesita trabajar.<br>- Se oprime el botón aceptar.<br>- Si se selecciona el botón Cancel se cierra la ventana. | |
| Observaciones: | |
| **Prototipo de interfaz:** | |

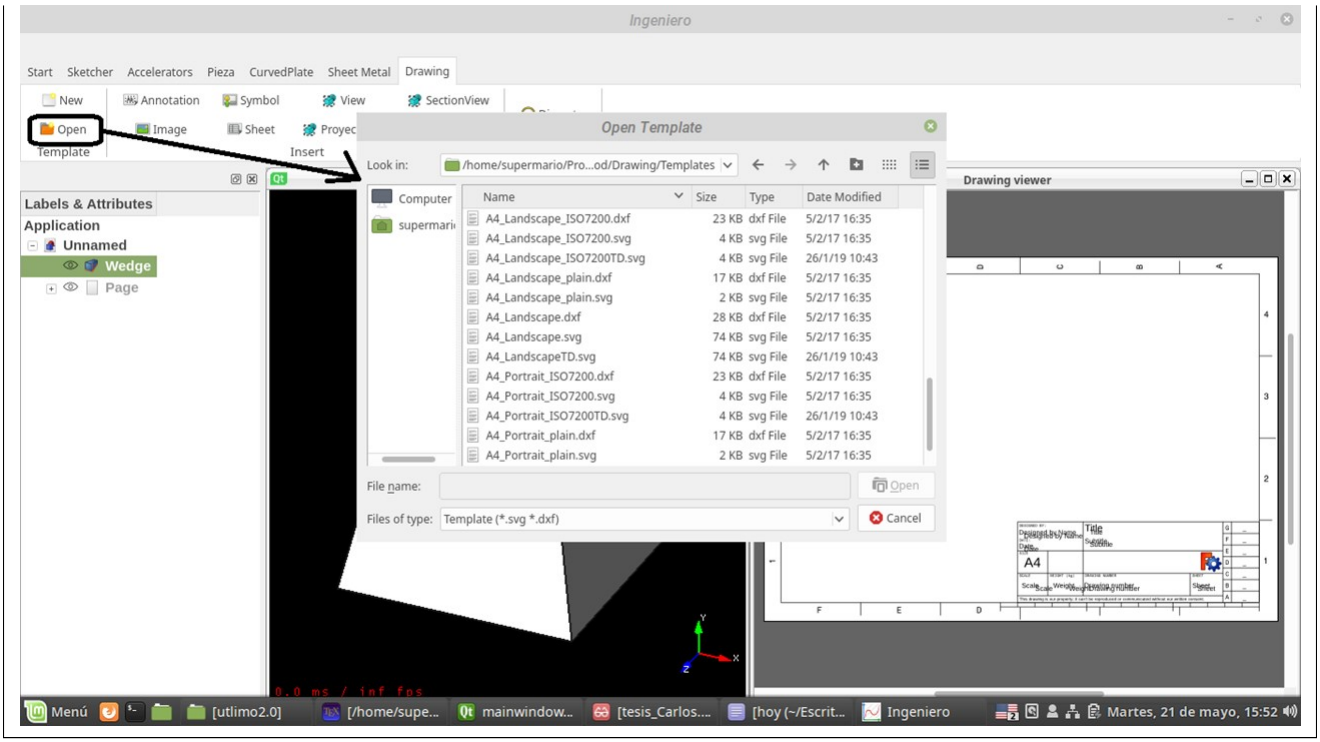



Tabla 5. Historia de Usuario Vistas proyectadas

| Número: 3 | Nombre del requisito: Realizar las vistas proyectadas |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1ra |
| Prioridad: Alta | Tiempo Estimado: 3 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 2, 5 semanas |

**Descripción:** Esta funcionalidad se utiliza para proyectar vistas auxiliares, mejorando la comprensión del modelo.

**1- Objetivo:**

Permitir realizar varias vistas del modelo en diferentes puntos de vistas

**2- Acciones para lograr el objetivo (precondiciones y datos):**

Para proyectar las diferentes vistas se debe tener en cuenta los siguientes aspectos:

- Tener abierta una plantilla de trabajo
- Una vista base del modelo

**3- Flujo de la acción a realizar:**

- Se selecciona el botón ProyectView.
- En el diálogo correspondiente se marca la opción de proyectar.
- Se oprime el botón aceptar.
- Y se arrastra la vista en la dirección enla cual se desee girar la figura para obtener la vista nueva
- Si se selecciona el botón Cancel se cierra la ventana.

Observaciones:

**Prototipo de interfaz:**

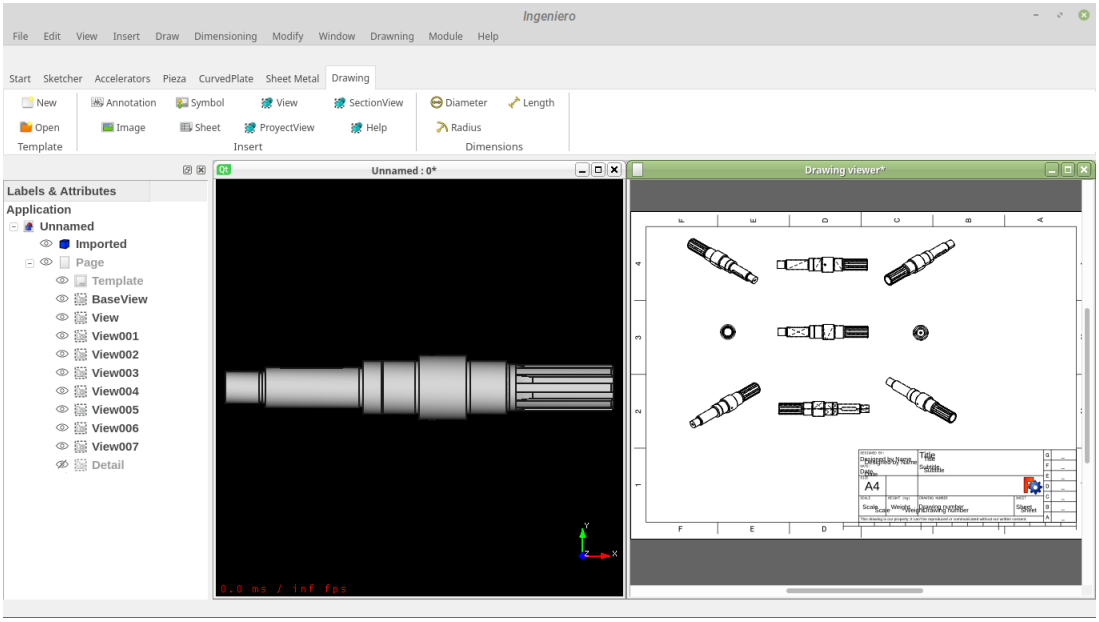



Tabla 6. Historia de Usuario Vistas de secciones

| Número: 4 | Nombre del requisito: Realizar las vistas de secciones |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1ra |
| Prioridad: Alta | Tiempo Estimado: 2 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 1, 5 semanas |
| **Descripción:** Esta funcionalidad permite observar la formación interior del modelo.<br>**1- Objetivo:**<br>Proyectar Varias vistas del modelo después de realizado un corte sobre el mismo<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para permitir realizar las vistas de secciones se debe tener en cuenta los siguientes aspectos:<br>- Tener abierta una plantilla de trabajo.<br>- Tener el modelo seleccionado.<br>**3- Flujo de la acción a realizar:**<br>- Se selecciona el botón SectionView, se selecciona el tipo de corte que se desea hacer y se presiona en aceptar.<br>- Se muestra en el área de trabajo la imagen de la proyección.<br>- Si se selecciona el botón Cancel se cierra la ventana. | |
| Observaciones: | |
| **Prototipo de interfaz:** | |

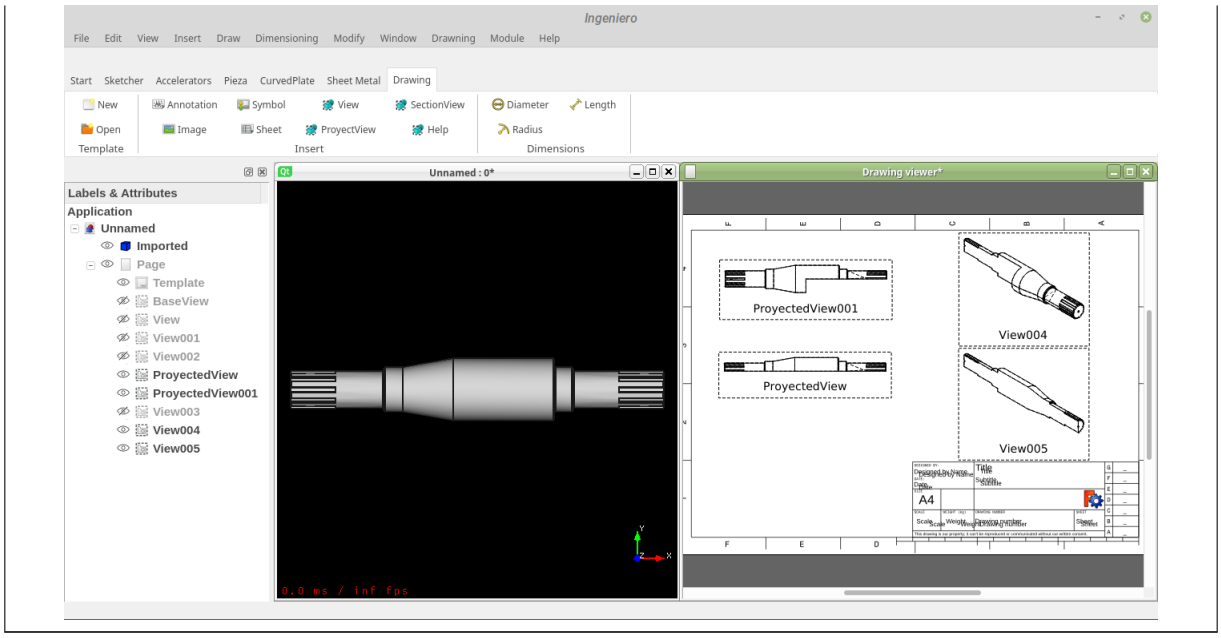



Tabla 7. Historia de Usuario realizar la vista de detalle

| Número: 5 | Nombre del requisito: Realizar la vista de detalle |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1era |
| Prioridad: Alta | Tiempo Estimado: 2 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 1, 5 semanas |
| **Descripción:**<br>**1- Objetivo:**<br>Permitir realizar la vista de detalle.<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para realizar la vista de detalle se debe tener en cuenta los siguientes aspectos:<br>- Una proyección sobre la plantilla<br>**3- Flujo de la acción a realizar:**<br>- Se selecciona la vista y el objeto del cual es obtenido se presiona el click derecho y se selecciona detalle | |
| Observaciones: | |
| **Prototipo de interfaz:** | |

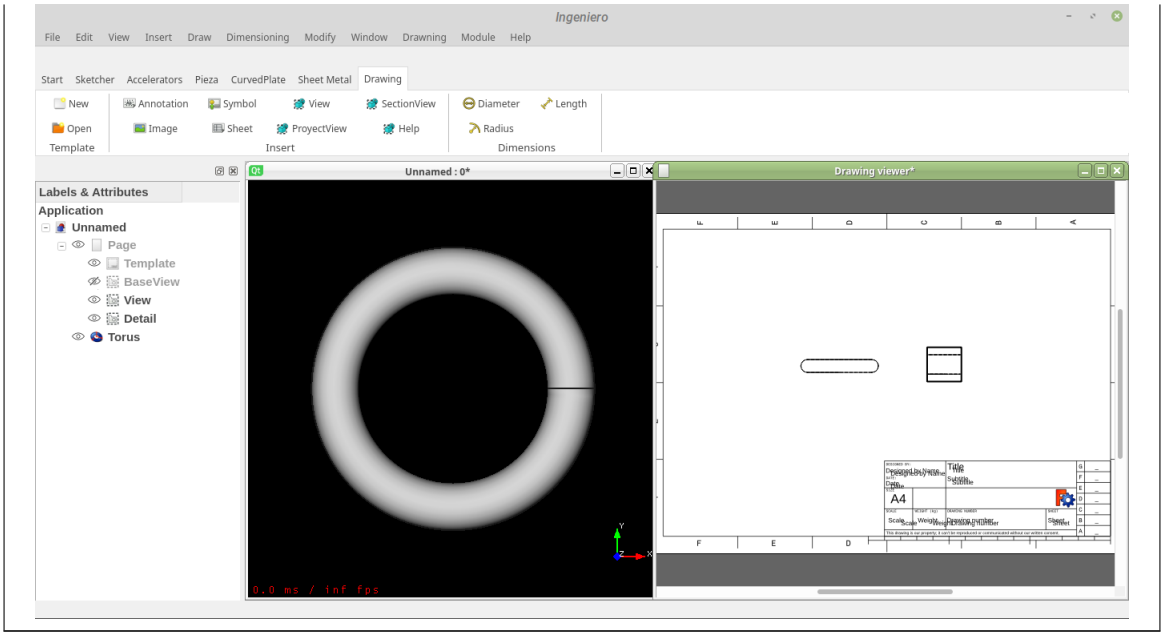



Tabla 8. Historia de Usuario actualizar una vista

| Número: 6 | Nombre del requisito: Actualizar una vista |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1era |
| Prioridad: Alta | Tiempo Estimado: 2 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 1, 5 semanas |
| **Descripción:**<br>**1- Objetivo:**<br>actualizar una vista de un modelo<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para actualizar una vista de un modelo se debe tener en cuenta los siguientes aspectos:<br>- Una proyección del objeto<br>- La clase SketchObject.<br>- Un objeto con modificaciones u otro objeto en el visor tridimensional<br>**3- Flujo de la acción a realizar:**<br>- Se modifica el objeto en el visor tridimensional.<br>- Se selecciona la vista a modificar y el objeto modificado o nuevo que se encuentra .<br>- Se presiona el click derecho y se selecciona Actualizar. | |
| Observaciones: | |
| **Prototipo de interfaz:** | |

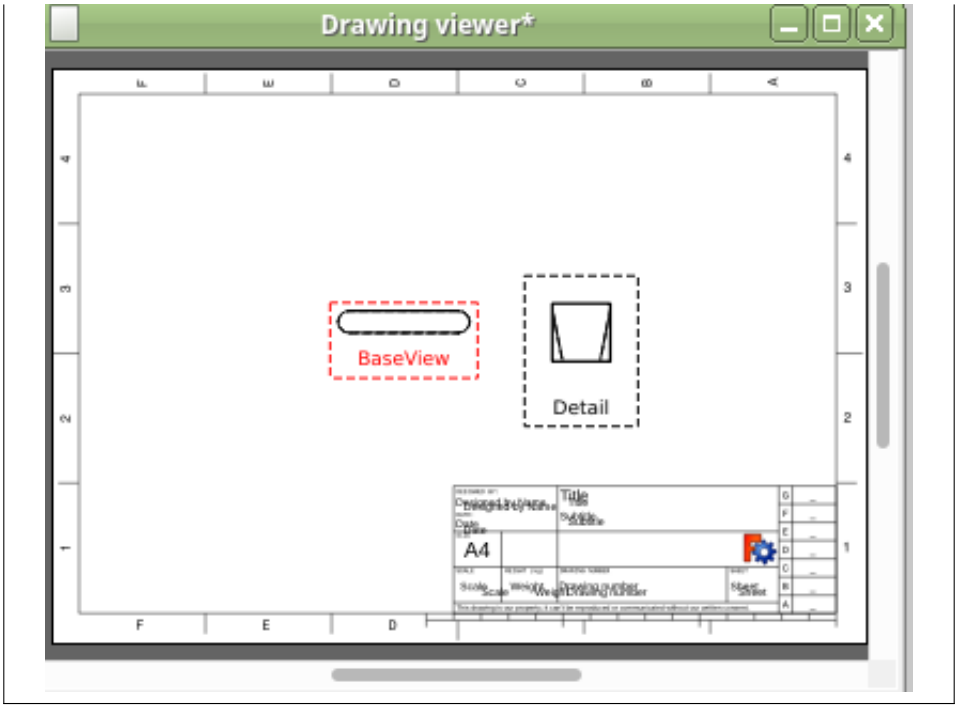
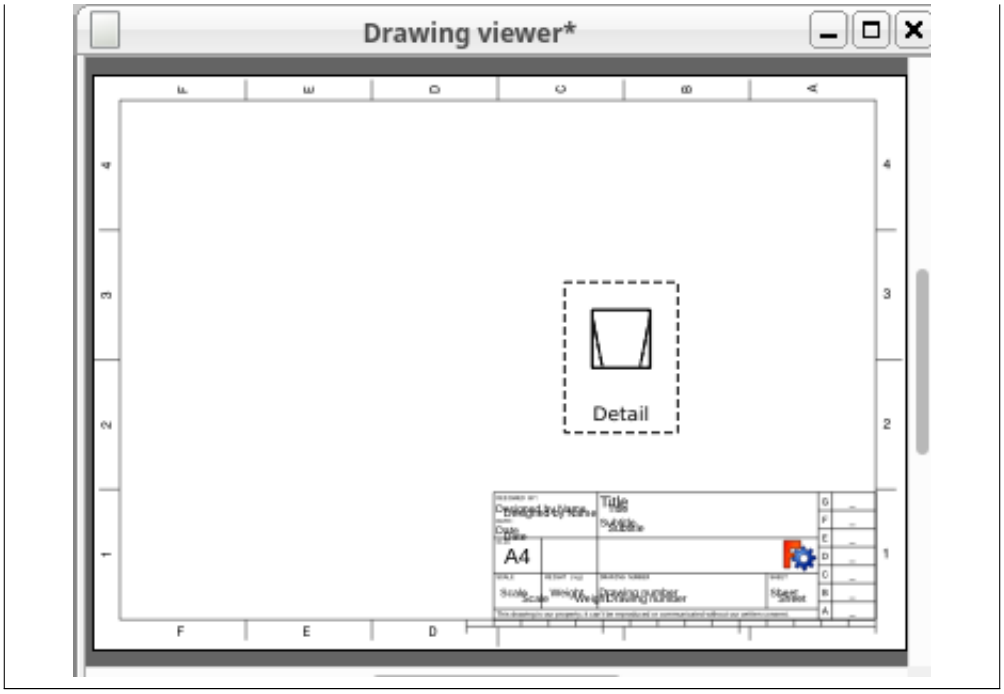



Tabla 9. Historia de Usuario Modelar las líneas de los contornos del casco de buque

| Número: 7 | Nombre del requisito: Eliminar vista |
|---|---|
| Programador: Mario Sánchez Regueira | Iteración Asignada: 1era |
| Prioridad: Alta | Tiempo Estimado: 2 semanas |
| Riesgo en Desarrollo: N/A | Tiempo Real: 1, 5 semanas |
| **Descripción:**<br>**1- Objetivo:**<br>Permitir eliminar una vista una vez proyectada.<br><br>**2- Acciones para lograr el objetivo (precondiciones y datos):**<br>Para eliminar una vista una vez proyectada se debe tener en cuenta los siguientes aspectos:<br>- Tener abierta una hoja de trabajo abierta<br>- Al menos una vista<br>**3- Flujo de la acción a realizar:**<br>- Seleccionar la vista que se desea eliminar<br>- Presionar click derecho<br>- Seleccionar eliminar | |
| Observaciones: | |
| **Prototipo de interfaz:** | |




## .2. Casos de Prueba

Tabla 10. Diseño de Casos de Prueba RF 2

| Descripción general | | | |
|---|---|---|---|
| Permitir proyectar vista base de un modelo | | | |
| SC 1 Proyectar vista base | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Vista base | Selecciona el botón view el cual mostrará un diálogo donde se selecciona la escala de proyección . | La escala representa la relación de tamaño entre el objeto y el dibujo. | DlgEscala /DrawingViewPart /Drawing |
| EC 1.2 Opción posición | Entra en los campos correspondientes las coordenadas de ubicación de la proyección. | Crea la superficie con los datos de entrada. | DlgEscala /DrawingViewPart /Drawing |
| EC 1.3 Opción de aceptar | Selecciona la opción de aceptar | Crea la vista base de la figura con los datos introducidos y cierra la ventana. | DrwaPage /DrawingViewPart /Drawing |
| EC 1.4 Opción de Close | Selecciona la opción de Close | Cierra la ventana. | DrwaPage /DrawingViewPart /Drawing |

Tabla 11. Diseño de Casos de Prueba RF 3

| Descripción general | | | |
|---|---|---|---|
| Permitir realizar varias vistas proyectadas | | | |
| SC 1 Vistas Proyectadas | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Opción Proyectada | Selecciona la opción Proyectada se presiona el botón aceptar se arrastra la vista base hacia la dirección en que se desea obtener la nueva vista. | Se crea una nueva vista conocida como vista auxiliar para entender mejor como es el objeto. | DlgVistaProyectada /DrawingViewPart /Drawing |
| EC 1.2 Opción HalfSectionView | Selecciona la opción HalfSectionView para proyectar la figura con un corte de 90 grados. | Se muestra la proyección en la dirección seleccionada con un corte de 90 grados permitiendo ver el interior. | DlgVistaProyectada /DrawingViewPart /Drawing |
| EC 1.3 Opción de FullSectionView | Selecciona la opción FullSectionView para proyectar la figura con un corte de 180 grados. | Se muestra la proyección en la dirección seleccionada con un corte de 180 grados permitiendo ver el interior. | DlgVistaProyectada /DrawingViewPart /Drawing |
| EC 1.4 Opción de Close | Selecciona la opción de Close | Cierra la ventana. | DlgVistaProyectada /DrawingViewPart /Drawing |

Tabla 12. Diseño de Casos de Prueba RF 4

| Descripción general | | | |
|---|---|---|---|
| Permitir realizar vista de sección | | | |
| SC 1 Vista de sección | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Botón SectionView | Presiona el botón SectionView | Se levanta un diálogo con dos opciones para seleccionar. | DlgSectionView /DrawingViewPart /Drawing |
| EC 1.2 Opción HalfSectionView | Selecciona la opción HalfSectionView para mostrar la figura con un corte de 90 grados. | Se muestra la proyección con un corte de 90 grados permitiendo ver el interior. | DlgSectionView /DrawingViewPart /Drawing |
| EC 1.3 Opción de FullSectionView | Selecciona la opción FullSectionView para mostrar la figura con un corte de 180 grados. | Se muestra la proyección con un corte de 180 grados permitiendo ver el interior. | DlgSectionView /DrawingViewPart /Drawing |

| | | | |
|---|---|---|---|
| EC 1.4 Opción de Close | Selecciona la opción de Close | Cierra la ventana. | DlgSectionView /DrawingViewPart /Drawing |

Tabla 13. Diseño de Casos de Prueba RF 5

| Descripción general | | | |
|---|---|---|---|
| Permitir realizar la vista de detalle | | | |
| SC 1 Vista de detalle | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Opción Detalle | Se presione el click derecho sobre la plantilla, se selecciona la opción de detalle. | Se genera una nueva vista conocida como vista de detalle para observar mejor un detalle del objeto. | DlgDetailView /MDIViewPart /Drawing |

Tabla 14. Diseño de Casos de Prueba RF 6

| Descripción general | | | |
|---|---|---|---|
| Permitir actualizar la vista | | | |
| SC 1 Actualizar vistas | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Opción Actualizar | Se presione el click derecho sobre la plantilla, se selecciona la opción de actualizar. | Se genera una nueva vista del objeto actual o modificado. | QGIView /MDIViewPart /Drawing |

Tabla 15. Diseño de Casos de Prueba RF 7

| Descripción general | | | |
|---|---|---|---|
| Eliminar vista | | | |
| SC 1 Eliminar vistas | | | |
| Escenario | Descripción | Respuesta del sistema | Flujo central |
| EC 1.1 Opción Eliminar | Se presione el click derecho sobre la plantilla, se selecciona la opción de eliminar. | Se elimina la vista del objeto sobre la plantilla. | QGIView /MDIViewPart /Drawing |

Tabla 16. Diseño de Casos de Prueba RF 9

| Descripción general |
|---|
| Exportar como SVG |
| SC 1 Exportar |

| Escenario | Descripción | Respuesta del sistema | Flujo central |
|---|---|---|---|
| EC 1.1 Opción Exportar | Se presione el click derecho sobre la plantilla, se selecciona la opción de exportar. | Se guarda todo el trabajo sobre la plantilla en formato SVG. | QGIView /MDIViewPart /Drawing |