Universidad de las Ciencias Informáticas
Facultad 4

Módulo para automatizar el diseño de engranaje de tornillo sin fin

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

Autor: Leandro Texidor Basterrechea

Tutores: DrC. Augusto César Rodríguez Medina

Ing. Gustavo García González

La Habana, 2018

"Not everething that counts can be counted and not everething that's counted truly conunts"

Albert Einstein

Dedicatoria

Agradecimientos

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 _____.

Leandro Texidor Basterrechea
Autor

DrC. Augusto César Rodríguez Medina
Tutor

Ing. Gustavo García González
Tutor

Resumen

El presente trabajo tiene como objetivo desarrollar un módulo, con las tecnologías disponibles de código abierto, para automatizar el diseño de engranaje de tornillo sin fin. En el mercado internacional existen sistemas para el diseño asistido por computadora que poseen módulos que aceleran el diseño de entidades mecánicas complejas, como es el caso de los engranajes de tornillo sin fin; pero estas aplicaciones comerciales no pueden ser adquiridas de forma legal a causa del bloqueo impuesto por parte de los Estados Unidos contra Cuba y los altos precios de estas herramientas. Son innumerables los usos industriales de esta entidad mecánica, los cuales van desde los sistemas de dirección de los automóviles hasta transportadores helicoidales. La solución propuesta fue desarrollada con los framework Open CASCADE y Qt, y para guiar el proceso de desarrollo se empleó como metodología AUP en su versión UCI. Con el módulo obtenido se reduce el tiempo de modelado de los engranajes de tornillo sin fin y se logró una alternativa ante el uso de sistemas comerciales en la industria cubana.

Palabras clave: engranaje de tornillo sin fin, sistemas de diseño asistido por computadora, componente, módulo, acelerador de diseño.

Índice general

Índice de figuras

Índice de tablas

Introducción

El presente trabajo de diploma posee sus bases en el proyecto del Grupo de Investigación Soluciones informáticas para la ingeniería y la industria (SIPII) perteneciente a la Facultad 4, en la Universidad de las Ciencias Informáticas; el documento es el resultado del proceso de investigación y desarrollo asociado a la creación de un módulo de engranajes de tornillo sin fin.

El Grupo de Investigación trabaja en la integración de módulos para la conformación de un sistema informático, con el objetivo de automatizar procesos industriales de alto impacto ingenieril en nuestro país, específicamente en el área de las tecnologías para el diseño asistido por computadora, asociados a las limitaciones del bloqueo económico y comercial impuesto por el gobierno de los Estados Unidos de América contra Cuba.

Para avalar los niveles de independencia y soberanía en el producto previsto, se utilizaron tecnologías de código abierto como la versión comunitaria de Open CASCADE, el marco de trabajo Qt y el código fuente de la aplicación FreeCad con el objetivo de reutilizar algunas funcionalidades; el proceso se formó transitando desde aspectos más simples hacia los más complejos creando un empleo sistemático, en lo fundamental, del método teórico de análisis y síntesis y el empírico de observación.

En el proceso de recopilación de la información, necesario para fundamentar el trabajo, se consultaron numerosas fuentes electrónicas, digitales e impresas; particular importancia tuvo la observación de animaciones, que se muestran en ejecución en los sistemas de diseño asistido por computadoras .

Estructura del trabajo de diploma:

El documento está estructurado en tres capítulos de la forma siguiente:

- **Capítulo 1 Fundamentos de la investigación, metodología y tecnologías.**
  Se exponen los aspectos generales de fundamentación teórica del trabajo, iniciando con la conformación del perfil de la investigación; se incluyen además los aspectos estudiados sobre los requerimientos de los mecanismos de sin fines en aplicaciones de diseño asistido por computadora, así como la metodología de software a emplear y las tecnologías para el desarrollo del módulo.
- **Capítulo 2 Descripción de la solución propuesta.**
  Se muestra un mapa conceptual que encapsula los conceptos fundamentales a tratar en la conformación del módulo, se exponen los requisitos capturados, se realiza la descripción del funcionamiento, se muestran las historias de usuarios, se define la arquitectura y se establecen los patrones de diseño.

- **Capítulo 3 Evaluación de la solución propuesta.**
  Se presentan los resultados más importantes de la etapa implementación y prueba como los estándares de codificación, diagrama de componente y las pruebas realizadas a las funcionalidades.

CAPÍTULO 1

Fundamentos teórico de la investigación, metodología y tecnologías

En el presente capítulo se exponen las consideraciones fundamentales acerca del estado del arte en los componentes de *software* destinados a acelerar el diseño de los engranajes de tornillo sin fin en los sistemas Diseño Asistido por Computadoras (CAD, por sus siglas en inglés); concluyendo así con los resultados de la investigación.

1.1. Motivación del proceso de investigación

Los procesos de producción comienzan con la etapa de concepción y diseño, que gracias a los avances científicos y tecnológicos de los últimos cincuenta años, han sufrido transformaciones a la hora de llevarse a cabo, comenzando con la elaboración de planos en la mesa de dibujo a partir de implementos para el trazado y continuando con el desarrollo de tecnologías como las aplicaciones informáticas, logrando así altos niveles de automatización, estas se conocen como herramientas CAD.

Uno de los problemas que enfrenta la industria nacional, son los elevados precios de las tecnologías CAD y la imposibilidad de acceder a ellas a causa de las restricciones del bloqueo económico, comercial y financiero impuesto por los Estados Unidos contra Cuba; esto se evidencia en las ramas de la ingeniería y el diseño, al no poder acceder de forma legal a herramientas comerciales y tener que utilizarlas con dudosa procedencia, lo que representa un riesgo para la seguridad informática; en tales condiciones es imposible fomentar un desarrollo tecnológico sustentable en el campo de la ingeniería, así como brindar servicios científico-técnicos en el exterior, por lo que aún si desapareciera el bloqueo los elevados precios de las herramientas de diseño asistido por computadoras comerciales sería difícil de sustentar.

Este problema también afecta el sector educacional cubano, los estudiantes de especialidades de ingeniería y arquitectura realizan sus trabajos en herramientas comerciales, lo que representa un asunto de seguridad nacional; los investigadores que utilizan las herramientas de diseño asistido por computadora para la elaboración de trabajos científicos, le resulta peligroso la divulgación de estos, a causa de posibles demandas por violación de leyes internacionales establecidas.

Los sistemas comerciales son distribuidos mediante licencias de uso, un ejemplo de ello es el *software*

Autodesk Inventor Professional 2018, herramienta de uso específico para la ingeniería mecánica, con un costo promedio de 2.110,0 euros por estación de trabajo, lo que representa alrededor de 21.100,0 euros por solo 10 profesionales del sector en solo un año. En la tabla 1.1 se muestran ejemplos de los precios de algunas de estas licencias de aplicaciones comerciales.

Tabla 1.1. Tabla de precios de licencias de herramientas comerciales (sitio-oficial-novedge, 2018).

Software	Precio	Tiempo de licencia
AutoCad 2019	1.715 euros	12 meses
Autodesk Inventor Professional 2018	2.110 euros	12 meses
SOLIDWORKS Standard	3.400 euros	12 meses
SOLIDWORKS Professional	4.350 euros	12 meses
SOLIDWORKS Premium	5.600 euros	12 meses
Solid Edge design and drafting	75 USD	1 meses
Solid Edge Foundation	185 USD	1 meses
Solid Edge Classic	230 USD	1 meses
Solid Edge Premium	329 USD	1 meses

Una alternativa ante las aplicaciones comerciales son las herramientas CAD de código abierto que nos permiten estudiar algoritmos y funcionalidades ya implementadas para reutilizarlas y contribuir a la obtención de soluciones propias, consiguiendo así aumentar la soberanía tecnológica y la no dependencia de los sistemas de código cerrado; pero estas tecnologías no cuentas con el desarrollo de sus homólogos comerciales, esto se pone de manifiesto por parte de los profesionales del sector de la industria de nuestro país, que rechazan la utilización de herramientas de código abierto y declaran su conformidad ante el uso de sistemas comerciales, a pesar de las limitaciones y riesgos que estas representan para la economía y la seguridad nacional.

En la Universidad de las Ciencias Informáticas el grupo de investigación SIPII perteneciente al Departamento de Ciencias Básicas de la facultad 4 trabaja en módulos CAD para automatizar el proceso de diseño de piezas complejas como engranajes cilíndricos de dientes rectos, resortes helicoidales de tracción, extensión y torsión, árboles escalonados y perfiles de viga normalizados o no, engranajes de tornillo sin fin, engranajes cónicos y helicoidales, entre otros. Para lograr estos módulos se estudiaron las tendencias en el desarrollo de los sistemas CAD/Ingeniería Asistida por Computadora desde su surgimiento con los sistemas más primitivos en las décadas del 50 y 60 del pasado siglo, así como el estudio y reutilización de cientos de líneas de código escritas en lenguaje C++ de las aplicaciones FreeCAD, LibreCad y Salome-Meca; también fue necesario asimilar las tecnologías Open CASCADE, de las que se han utilizado algoritmos y funciones para el modelado y Coin3D, que es una implementación de la API Open Inventor, para la visualización. Con los módulos realizados queda demostrado que si se puede desarrollar un sistema CAD que cumpla con las necesidades del sector industrial de nuestro país es posible, pero constituye una tarea de gran escala, que requiere recursos, esfuerzos, organización y cooperación entre diferentes entidades del país.

El presente trabajo de diploma se basa específicamente en un módulo para acelerar el diseño del engrana-

je de tornillo sin fin; este componente mecánico está diseñado para transmitir grandes esfuerzos, movimiento rotatorio y como reductor de velocidad; generalmente trabajan en ejes que se cortan a 90 grados y tiene la desventaja de no ser reversible el sentido de giro, sobre todo para grandes relaciones de transmisión. La rotación de la rueda sin fin sobre su eje, se produce por la acción de cuña del helicoide del tornillo debido a la propia geometría de la transmisión, el movimiento de rotación del helicoide del tornillo se traduce en un efecto de sucesivas cremalleras cuyos perfiles avanzan en la dirección axial del mismo; al ponerse en movimiento el engranaje, se originan líneas de contacto que progresan desde la cabeza hasta el pie del diente de la rueda. En construcciones de mayor calidad la corona está fabricada de bronce y el tornillo sin fin de acero templado, con el objetivo de reducir el rozamiento, ya que este mecanismo transmite grandes esfuerzos, es necesario que esté muy bien lubricado para matizar los desgastes por fricción (Radzevich, 2013).

Este tipo de mecanismo tiene una gran utilización en las diferentes ramas de la industria nacional, esto se debe a su gran uso en los sistemas de puertas automáticas, en los instrumentos musicales de cuerda, en los elevadores, en sistemas de dirección de los automóviles y fundamentalmente como reductores de velocidad siendo esta última una de las principales ventajas de este tipo de engranaje.

Teniendo en cuenta lo antes expuesto se elabora el siguiente **problema de investigación:** ¿Cómo automatizar el diseño de engranajes de tornillo sin fin empleando tecnologías de código abierto?

El **objeto de estudio** de la investigación se centra en el gráfico de computadoras para aplicaciones industriales, teniendo como **campo de acción**, el desarrollo de un componente CAD para los engranajes de tornillo sin fin.

Para la solución del problema investigación se formula como **objetivo general:** Desarrollar un componente de *software* para automatizar el diseño de engranajes de tornillo sin fin, mediante el empleo de tecnologías de código abierto.

De este objetivo general se derivan los siguientes **objetivos específicos**:

1. Caracterizar los sistemas de código abierto en relación con sus potencialidades y limitaciones para el modelado de engranajes de tornillo sin fin.
2. Identificar las características esenciales y las funcionalidades básicas de carácter general, necesarias para conformar el módulo de engranaje de tornillo sin fin.
3. obtener un módulo informático para acelerar el modelado de engranajes de tornillo sin fin.

Para dar cumplimiento a estos objetivos específicos se proponen las siguientes **tareas de investigación**:
Marco Teórico.

- Definición del perfil y diseño de la investigación.
- Búsqueda y revisión bibliográfica sobre el objeto de investigación.

Análisis.

- Fundamentación teórico-matemática de los engranajes de tornillo sin fin.

- Estudio de las funcionalidades de los módulos para el diseño de engranajes de tornillo sin fin, existentes en aplicaciones comerciales para los sistemas CAD.
- Análisis de los códigos fuente de las aplicaciones FreeCAD y Salome-Meca con la finalidad de identificar y evaluar la reutilización de sus funcionalidades, así mismo, identificar las funcionalidades de la tecnología OpenCascade que serán utilizadas en el proceso de desarrollo.
- Proceso de síntesis (obtención de conclusiones, resultados, definición de las formas de hacer y conformación de la estructura del componente).
- Estudio de los aspectos de la metodología de desarrollo de software (AUP-UCI) que se aplicarán en la investigación.
- Obtención de requisitos a partir de los módulos de pieza existentes en sistemas propietarios, de código abierto, así como de las consideraciones de los potenciales usuarios.

Diseño.

- Definición de la arquitectura del componente, lo que implica además definir los patrones de diseño con los que cumplirá.
- Elaboración de la estructura de clases del componente.
- Diseño de la interfaz gráfica del componente.

Implementación.

- Implementación del módulo destinado a la automatización del proceso de diseño de engranajes de tornillo sin fin.
- Realizar pruebas al módulo (de las diferentes funcionalidades).

Para resolver y dar cumplimiento a los objetivos y las tareas propuestas se emplearon los siguientes **métodos de investigación:**

Métodos teóricos:

- **Análisis y síntesis**: se empleó para la construcción y desarrollo de la teoría, profundización en el tema y la sistematización del conocimiento.
- **Modelado**: se empleó la generación de los artefactos que permitieron una mejor comprensión entre el equipo de desarrollo y las personas relacionadas.

Métodos empíricos:

- **Observación**: se empleó para caracterizar las soluciones, teniendo en cuenta distintos datos de otras aplicaciones y así establecer los requisitos con las principales funcionalidades de estos.

1.2. Generalidades de los engranajes de tornillo sin fin

Una transmisión de tornillo sin fin consta de un cilindro con dientes en forma de espiral con un paso determinado, este cuerpo se le llama tornillo sin fin. Al desplazar una vuelta completa el tornillo sin fin, el

engranaje helicoidal se desplaza en un diente completo, este tipo de engranaje es aplicado cuando se requiere de una gran relación de transmisión. Gracias al acople deslizante de los dientes, las transmisiones de tornillo sin fin trabajan sin hacer ruido y pueden absorber cargas de choque.

A grandes relaciones de transmisión, los mecanismos de tornillo sin fin cuentan con la propiedad de frenado; el arrastre por fuerza de estos engranajes es sólo posible en dirección del tornillo sin fin al engranaje helicoidal. El rozamiento de deslizamiento que surge debido al movimiento helicoidal es considerablemente mayor que en engranajes cilíndricos o cónicos. Para poder conducir el calor de rozamiento, estas transmisiones deben lubricarse especialmente bien, para la conducción de calor pueden utilizarse adicionalmente aletas de refrigeración o ventiladores. El material del tornillo sin fin debe ser más duro que el material del engranaje helicoidal, de esa forma se pueden obtener buenas propiedades de deslizamiento.

1.2.1. Movimiento relativo entre el tornillo sin fin y la rueda

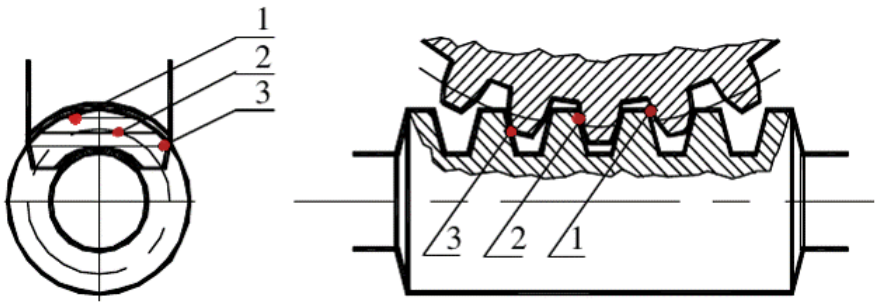
El tornillo sin fin es un mecanismo de trasmisión circular compuesto por dos elementos, el tornillo o piñón que actúa como entrada y el engranaje o corona que actúa como elemento de salida, es usado por su gran rapidez de reducción ya que, para que la rueda gire una vuelta completa es necesario que el tornillo gire tantas veces como dientes tenga el engranaje. Los ejes son normalmente perpendiculares, aunque es posible que tenga otros ángulos. El tornillo es siempre el conductor del movimiento en los reductores de velocidad presentando un bajo rozamiento y una marcha silenciosa, el mismo se cierra automáticamente cuando el engranaje no puede conducir al tornillo sin fin, esto ocurre cuando la tangente del ángulo principal es menor que el coeficiente de fricción, si el ángulo principal del engranaje corre por debajo de los 2 grados este no podrá ser cerrado automáticamente.

El mínimo número de dientes en los engranajes y la proporción de reducción determina el número de hilos para el tornillo; generalmente se usan de 1 a 6 hilos, pero en casos especiales requiere un mayor número. La rueda dentada cilíndrica con dentado helicoidal, presenta una garganta con centro de curvatura coincidente con el eje del sin fin (diente cóncavo), de este modo el contacto entre los dientes de la rueda y la hélice del sin fin es lineal, lo que permite transmitir potencias elevadas.Cuando existe movimiento en el engranaje, aparecen las líneas de contacto que comienzan desde la cabeza hasta el pie del diente de la rueda (ver secuencia de la figura 1.1), donde se han representado las líneas mediante rectas para mayor claridad (Rivero Llerena, 2002).

Si las líneas de contacto están ubicadas por encima del círculo de cresta de la rueda se producen una acción de acercamiento hacia este círculo, en caso contrario ocurre una acción de alejamiento. En cada intervalo pueden existir de 2 a 3 dientes en contacto con lo que la carga total queda repartida.

La interacción entre los filetes del tornillo y los dientes de la rueda se establece a través de un movimiento relativo entre ambos en el que predomina el deslizamiento. Para que el movimiento relativo entre los flancos sea uniforme, se requiere que ambas superficies sean conjugadas, lo que brinda una marcha suave con muy poco ruido, pero requiere de atenciones especiales como son una gran precisión de elaboración y montaje, además del establecimiento de un régimen de lubricación adecuado para mantenerlos dentro de

Figura 1.1. Secuencia del contacto de los dientes de la rueda.



límites reducidos, las pérdidas de potencia y el desgaste. La geometría bien conjugada garantiza un buen funcionamiento al propiciar curvaturas de las líneas de contacto, típicas de cada perfil, que dan lugar a la cuña lubricante entre los dientes de la rueda y los filetes del tornillo (Rivero Llerena, 2002).

El auto frenado es una de las peculiaridades más importantes a destacar en este tipo de trasmisión, el cual se favorece cuando el paso axial del tornillo es más pequeño. El ángulo de rozamiento entre las superficies conjugadas es uno de los parámetros esenciales para la evaluación del auto frenado, lo cual no se puede medir con suficiente exactitud por su carácter interdependiente por lo que algunos autores recomiendan dar crédito a la condición de auto frenado mediante evaluaciones prácticas.

1.2.2. Aspectos tribológicos en las transmisiones por tornillo sin fin

La trasmisión sin fin, a diferencia de otros tipos de engranajes posee su propia cinemática lo cual permite un mayor deslizamiento entre la rodadura de los flancos del tornillo y la rueda. Para disminuir efectos no deseados ha existido un extenso proceso de perfeccionamiento de la geometría, materiales y tecnología empleados para su obtención. Este desarrollo también ha estado unido a los progresos manejados en la ciencia de la tribología , todo lo cual ha permitido obtener valores aceptables en el rendimiento de la transmisión.

El mayor deterioro presente en esta trasmisión esta dado cuando se origina la velocidad de deslizamiento, surgen así grandes fuerzas de fricción y un aumento de la temperatura, lo que conduce al desgaste del mecanismo engranado. Existen diferentes elementos por los que ocurre el desgate mecánicos los cuales pueden estar evidenciados por:

- Tipo y magnitud del esfuerzo.
- Tipo y cantidad de lubricante.
- Temperatura de las superficies en contacto.
- Acabado superficial.
- Tiempo de trabajo.
- Velocidad.
- Medio ambiente.
- Propiedades mecánicas de los materiales.

Los materiales a emplear se definen según la capa superficial en contacto, en presencia del lubricante, por los que no resulta conveniente emplear ambos materiales duros debido a la flexión del tornillo, por lo que generalmente la rueda se fabrica de un material antifricción relativamente más blando (Rivero Llerena, 2002).

1.2.3. Aplicaciones del engranaje de tornillo sin fin

Uno de los principales y más usados mecanismos de transmisión en cualquier proyecto mecánico es el tornillo sin fin, que actúa como un elemento motriz o de entrada, se destaca por su sencillo funcionamiento y ganancia mecánica. Por lo que son de uso frecuente en las maquinarias de las industrias, a continuación, pasaremos a ver algunos ejemplos de su utilización.

Entre las aplicaciones del tornillo sin fin se destacan:

- **Sistema de puertas automáticas:** En estos sistemas es necesario el movimiento en una dirección (derecha a izquierda) y un bloqueo del sistema al final de la carrera; gracias a este mecanismo es posible que el sistema de apertura y cierre de las puertas; cada vez que los dos tornillos se encuentran la puerta queda cerrada, por lo que el sistema queda bloqueado gracias al engrane helicoidal.

- **Instrumentos musicales:** Los tornillos sin fines son de uso habitual en los sistemas de ajuste de guitarras, violines y otros instrumentos de cuerda. Su gran fuerza mecánica permite tensionarlas con muy poco esfuerzo.

- **Elevadores o transportadores helicoidales:** Estos equipos incluyen un tornillo helicoidal de paso amplio, fijo o variable que gira dentro de una carcasa desplazando el producto desde la boca de entrada a la compuerta de salida.

- **Sistemas de dirección de automóvil:** En los mecanismos de dirección del automóvil se tiene al sin fin como uno de sus componentes básicos. En estos sistemas el tornillo engrana constantemente con una rueda dentada, a su vez, el sin fin se une al volante mediante la columna de dirección, y las ruedas lo hacen al brazo de mando. Gracias a este mecanismo, por cada vuelta del volante, la rueda del coche gira un cierto ángulo, que depende según la relación de reducción efectuada.

- **Reductores de velocidad:** Los reductores tipo sin fin o de 90 grado tienen gran presencia en diferentes aplicaciones industriales debido a que ofrecen importantes radios de reducción en poco espacio. Por otro lado, la transmisión del movimiento se hace a 90 grados, lo que permite realizar la transmisión en un ángulo recto. Como desventaja, los reductores de engranaje tornillo sin fin están sometidos a una mayor fricción por su deslizamiento lo que provoca altas temperaturas de operación y mayores gastos energéticos.

1.2.4. Geometría de la transmisión de tornillo sin fin

En muchos casos el desarrollo de nuevas tecnologías de elaboración ha sido aprovechado como factor determinante en la concepción de la configuración geométrica final de la transmisión, de la que depende en buena medida, el comportamiento en servicio del par engranado. En la geometría externa los parámetros de más trascendencia para la eficacia de la transmisión son dos:

- **Distancia interaxial (aw):** Este es uno de los parámetros de mayor influencia en la transmisión de momento torsional, aunque esto no aparece explícitamente en las formulaciones generales.
- **Diámetro de fondo del tornillo (df1):** Este parámetro en combinación con la distancia entre los cojinetes del tornillo (L1) incide en el grado de rigidez, que influye en la cantidad de energía que se pierde por causa de las deformaciones elásticas. Del valor del diámetro de fondo depende el diámetro de referencia (d1) y el momento de fricción que tiene influencia en la eficiencia de la transmisión.

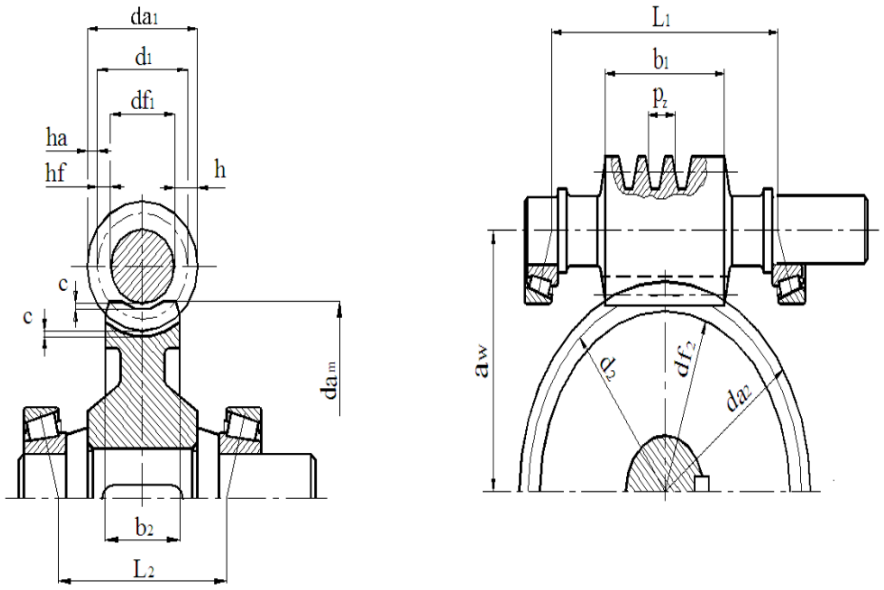


Figura 1.2. Parámetros geométricos de la transmisión por tornillo sin fin.

1.2.5. Metodología de cálculo de los engranajes de tornillo sin fin

Este mecanismo al igual que muchos en la rama de la mecánica, se encuentran normalizados por estándares internacionales como ISO(americana), AGMA(americana), DIN(alemana) y la GOST (soviética), la metodología de cálculo que se expone a continuación es la que plantea AGMA, ya que esta es una de las normas que se imparten en la facultad de ingeniería mecánica en nuestro país.

Z_w**:** Es el número de hilos que tiene el tornillo. Se simboliza como **(Z)**. Es fundamental para calcular la relación de transmisión.

Z_2**:** Es el número de dientes que tiene el engranaje. Se simboliza como **(Z)**. Es fundamental para calcular la relación de transmisión. El número de dientes de un engranaje no debe estar por debajo de 18 dientes cuando el ángulo de presión es 20 grados ni por debajo de 12 dientes cuando el ángulo de presión es de 25 grados.

m_n**:** Módulo normal es una magnitud expresada en milímetros de la relación que existe entre el diámetro primitivo y el número de dientes. Valores típicos: 1,1.25,1.50,2,2.25,2.50,3,4,5,6,8,10,12,16,20,25.

α_n**:** Es el ángulo que se utiliza para el trazado del envolvente de circulo para la creación del perfil de los dientes.

d_1**:** Es el diámetro de referencia del tornillo, sirve para la creación del círculo de referencia, que es el círculo base de los demás circunferencias de construcción y siempre tangente a la circunferencia del engranaje.

X_{n_2}**:** El coeficiente de corrección de la rueda, es el valor de separación que existe entre el piñón y la corona.

Tabla 1.2. Ecuaciones de los engranajes de tornillo sin fin (G. Budynas y Keith Nisbett, 2011),(Michalec, 2015).

Símbolo	Nombre del Símbolo	Fómula	Observaciones
U	Razón del engranaje	$U = \frac{Z_2}{Z_w}$	
γ	Ángulo medio del cilindro de referencia	$\arcsin\left(\frac{m_n \cdot Z_w}{d_1}\right)$	
d_2	Diámetro de referencia de la rueda	$d_2 = \frac{Z_2 \cdot m_n}{cos(\gamma)}$	
a	Distancia entre centro	$a = \frac{d_1 + d_2}{2} + X_{n2} \cdot m_n$	
h_{a1}	Addendum del tornillo	$h_{a1} = 1.00 \cdot m_n$	

h_{a2}	Addendum de la rueda	$h_{a2} = (1.00 + X_{n2}) \cdot m_n$	
h	Profundidad del diente	$h = 2.25 \cdot m_n$	
d_{a1}	Diámetro de cresta del tornillo	$d_{a1} = d_1 + 2 \cdot h_{a1}$	
d_{a2}	Diámetro de cresta de la rueda	$d_{a2} = d_2 + 2 \cdot h_{a2} + m_n$	
d_t	Diámetro de la garganta de la rueda	$d_t = d_2 + 2 \cdot h_{a2}$	
r_i	Radio de la superficie de garganta de la rueda	$r_i = \frac{d_1}{2} - h_{a1}$	
d_{f1}	Diámetro de fondo del tornillo	$d_{f1} = d_{a1} - 2 \cdot h$	
d_{f2}	Diámetro de fondo de la rueda	$d_{f2} = d_t - 2 \cdot h$	
b_1	Ancho de la zona tallada del tornillo	$b_1 = \pi \cdot m_n \cdot (4.5 + 0.02 \cdot Z_2)$	
b_2	Ancho de la rueda	$b_2 = 2 \cdot m_n \cdot 0.5 + \sqrt{\frac{d_1}{m_n} + 1}$	

1.3. Opciones para el modelado de los mecanismos de tornillo sin fin

Con las mejoras de *hardware* y *software* de las computadoras y la evolución continua en las ciencias informáticas, se han creado un grupo de sistemas informáticos para satisfacer las necesidades de los ingenieros y diseñadores en su trabajo diario, facilitándoles así el proceso de diseño en las múltiples áreas en que se desarrollan y dando pie a la reducción del tiempo de trabajo y el aumento de la calidad. En este apartado se abordarán algunos de estos sistemas, los cuales pueden ser de escritorio o de plataforma web.

1.3.1. Módulos existentes en los sistemas de diseño asistido por computadora

Con el objetivo de reducir los costos de producción y agilizar el proceso de diseño de cualquier proyecto mecánico, existen variedades de herramientas para el apoyo a este campo de gran importancia en las industrias.

Aplicación interactiva tridimencional asistida por computadora (CATIA, por sus siglas en inglés)

Esta herramienta informática fue creada por la compañía *Dassault Systèmes*, con el objetivo de proporcionar apoyo desde la concepción del diseño hasta la producción y el análisis de productos, se encuentra disponible para múltiples plataformas, *Microsoft Windows*, *Solaris*, *IRIX* y *HP-UX*.

En sus inicios se desarrollaba para servir en la industria aeronáutica en el diseño de los prototipos de los aviones para compañías como: *DassaultAviation* (*Airplane*, *France*), *Grumman* (*Airplane*, *USA*), *SNECMA* (*JetEngine*, *France*); actualmente se utiliza en la industria automovilística por compañías como: VW (*Volkswagen*, *Audi*, *SEAT* y *Škoda)*, *BMW*, *Renault*, *Peugeot*, *Daimler AG*, *Chrysler*, *Smart* y *Porsche*, para el diseño y desarrollo de componentes de carrocería. CATIA ofrece la posibilidad, no solo de modelar cualquier producto, sino de hacerlo en el contexto de su comportamiento en la vida real (Systèmes, 2017).

Aunque CATIA no cuenta con acelerador de diseño para los engranajes de tornillo sin fin; porque el sistema está pensado para que los ingenieros utilicen sus funcionalidades y diseñe de manera libre, sin embargo, si se puede modelar este tipo de mecanismo, lo que para hacerlo se necesita un diseñador mecánico que domine todas las funcionalidades de esta aplicación.

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 Systèmes*, mejor conocida por su *software* CATIA, adquirió la compañía y actualmente posee el cien por ciento de sus acciones.

El SolidWorks es un modelador de sólidos paramétrico que permite modelar piezas y mecanismos, extraer de ellos tanto planos técnicos como información necesaria para la producción de componentes, el programa posee un grupo de productos con distintas funcionalidades:

- **Soluciones CAD 3D:** permite a las empresas acelerar el desarrollo de productos, reducir los costes de fabricación, y mejorar la calidad y fiabilidad del producto en una gran variedad de sectores y aplicaciones (EULALIA IZARD, 2017).
- **Visualización:** proporciona un conjunto de herramientas de *software* independientes que combinan las funciones de renderizado líderes del sector con unas características y flujos de trabajo orientados al diseño, las cuales facilitan la creacion fácil y rápida de contenido visual a diseñadores, ingenieros, expertos de marketing y otros creadores de contenidos (SolidWorks-Corp., 2017c).
- **Simulación:** brinda una herramienta para realizar pruebas con una amplia variedad de parámetros (durabilidad, respuesta dinámica y estática, movimiento del ensamblaje, transferencia de calor, dinámica de fluidos y moldeo de plásticos por inyección) durante el proceso de diseño (SolidWorks-Corp., 2017d).
- **Gestión de datos de productos:** permite tener control sobre los datos de diseño y mejorar considerablemente la manera en que sus equipos administran y colaboran en el desarrollo de productos (SolidWorks-Corp., 2017b).
- **Diseño eléctrico:** proporciona una serie de funcionalidades de diseño de sistemas eléctricos para satisfacer las necesidades de los profesionales (SolidWorks-Corp., 2017a).

La aplicación no tiene un acelerador de diseño para los engranajes de tornillo sin fin, pero si contiene una biblioteca de diseño que almacena una serie de piezas mecánicas construidas, las cuales pueden ser cargadas y nos dan la posibilidad de editar sus propiedades para obtener un nuevo modelo mecánico. Entre las piezas que encontramos están diferentes tipos de tornillos, engranajes de dientes rectos, cónicos, helicoidales entre otros.

Autodesk Inventor

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, así como el comportamiento del producto antes que sea construido (Autodesk, 2017).

Inventor utiliza formatos específicos para importar y exportar los diseños realizados, ejemplo de estos son: archivo para las piezas (*.IPT*), ensamblajes (*.IAM*), vista del dibujo (*.IDW y .DWG*) y presentaciones (*IPN*); en las últimas versiones de Inventor se utiliza un gestor de formas (*Shape Manager*) como kernel de modelado geométrico el cual pertenece a *Autodesk*; en la versión profesional incluye herramientas necesarias para crear piezas de plástico y sus respectivos moldes de inyección, cuenta con análisis de tensión por elementos finitos y análisis dinámico además de utilizar tecnologías como *iPart*, *iAssembly*, *iMates*, *iCopy*,

iLogic hacen que el diseño sea más rápido y eficiente y su combinación con *Autodesk Vault* y *Autodesk 360* la hacen líder en el mercado del diseño mecánico (Autodesk, 2017).

Inventor si cuenta con un acelerador de diseño para los engranajes de tornillo sin fin, el mismos se encuentra en el modo *Assemble* en la pestaña *Design* en el apartado *Power Transmission* se selecciona *Worm Gears* y se activa la ventana del acelerador donde se pueden introducir parámetros como relación de trasmisión, número del hilos y de dientes, diámetro de referencia, módulo, entre otras propiedades los cuales sirven para la creación del engranaje de tornillo sin fin.

Solid Edge

Fue lanzado al mercado por primera vez en 1996 inicialmente desarrollado por *Intergraph*, hoy en día pertenece y es desarrollado por *Siemens AG*.Es un programa parametrizado CAD de piezas tridimensionales. Permite el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros.

Solid Edge es una herramienta de *software* fácil de usar para el proceso de desarrollo de productos diseño en 3D, simulación, fabricación, gestión del diseño ya que combina la velocidad y simplicidad del modelado directo con la flexibilidad y el control de diseño paramétrico hecho posible con *synchronous technology* (Siemens, 2017).

Solid Edge al igual que Inventor posee un acelerado de diseño en cual se encuentra en el modo *Assemble* en *Engineering Reference* dentro se selecciona *Worm Gear Designer* y se activa la ventana del acelerador donde se pueden introducir parámetros como relación de trasmisión, número del hilos y de dientes, diámetro de referencia, módulo, entre otras propiedades los cuales sirven para la creación del engranaje de tornillo sin fin.

FreeCAD

Es un modelador CAD 3D desarrollado completamente en código abierto con Licencia Pública General Reduccida (LGPL, por sus siglas en inglés), para la asistencia en ingeniería mecánica y el diseño de elementos mecánicos, aunque también se emplea en amplia gama de usos en la ingeniería y la arquitectura . 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 código abierto de la biblioteca de Inventor abierto, Qt y Python.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

"FreeCAD cuenta con herramientas similares a CATIA, SolidWorks o Solid Edge, y por lo tanto también cae en la categoría de Diseño Asistido por Computadoras orientado a la Mecánica (MCAD, por sus siglas en inglés), PLM, CAx y Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés). 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 (Riegel y Mayer, 2017)."

La versión 0.17 no posee, al igual que CATIA y SolidWork, un acelerador gráfico para el modelado y cálculo del engranaje de tornillo sin fin. El procedimiento en esta versión sería realizar una hélice y un cilindro en el módulo "Part". Posteriormente, se pasa a trabajar en el módulo "Part Design", se crea un plano y el perfil del alambre en el mismo; para ubicar este en el inicio de la hélice se puede desplazar con las restricciones . Finalmente, con la herramienta "Sweep" del módulo "Part" se selecciona el camino y el perfil, se marca "crear sólido" y al aplicar queda conformado la rosca del tornillo.

1.3.2. Herramientas para el cálculo de los engranajes de tornillo sin fin

Con el objetivo de agilizar el trabajo de diseño se han desarrollado distintas herramientas que automatizan el proceso de los cálculos de las propiedades del engranaje de tornillo sin fin. Algunas de estas son MITCalc y eFunda.

MITCalc

Es una aplicación de escritorio creada para calcular las propiedades de distintos sistemas mecánicos, pudiendo usar diferentes estándares de cálculos como son ANSI/AGMA 6022-C93 (Revisión de AGMA 341.02), ANSI/AGMA 6034-B92 (Revisión de ANSI/AGMA 6034-A87), DIN 3996, DIN 3975-1, DIN 3975-2, entre otros.

"Esta herramienta brinda, además de los cálculos, las verificaciones comunes de diseño de disímiles elementos mecánicos como: engranajes rectos, los de tornillo sin fin, los cónicos, los helicoidal, engranaje planetario, correa de distribución, cojinetes, resortes, viga, pandeo, placas, conchas, y muchos otros (MITCalc, 2017)."

Permite la salida directa a disímiles sistemas CAD propietarios, como:

- *AutoCAD* (12-2014)
- *AutoCAD LT* (95-2.014)
- *IntelliCAD*
- *Ashlar Graphite*
- *TurboCAD*
- *BricsCAD*

En la versión 1.73 del MITCalc permite realizar todos los cálculos asociados a los engranajes de tornillo sin fin, mediante la metodología de cálculo que se seleccione; la hoja de cálculo esta compuesta por las siguientes pestañas, *Options of basic input parameters*,*Options of material, loading conditions, operational and production parameters*, *Parameters of the tooth profile*, *Design of a geometry of toothing*, entre otras. En los Anexos B.1 se muestra un ejemplo de una iteración en el MITCalc.

La aplicación brinda una buena precisión en la realización de los cálculos de los engranajes de tornillo sin fin resolviendo las siguientes tareas: cálculo de las dimensiones de los engranajes, diseño de trasmisiones automática con mínimo de entras requeridas, cálculos de parámetros geométricos, cálculos de los parámetros de fuerza y control de seguridad y cálculos auxiliares de calentamiento y ejes de diseño. Estos valores son mostrados en una tabla con los resultados obtenidos, la cual puede ser exportada a Excel o a cualquieras de las aplicaciones antes mencionadas.

eFunda

Es una calculadora existente en la plataforma web que tiene como objetivo crear para la comunidad de ingenieros una herramienta que sea capaz de registrar cálculos anteriormente realizados, para poder ser utilizados por otros ingenieros y minimizar el tiempo de trabajo. eFunda brinda los conceptos básicos que cubren los materiales de nivel universitario de las escuelas de ingeniería. También, abarca exactamente en qué condiciones se aplican las fórmulas de esta rama y de los sustentos matemáticos relacionados a la misma. Uno de sus valores agregados es que brinda soporte a una serie de calculadoras on-line de finanzas, matemática, mecánica y diseño (eFunda, 2017).

Realiza los cálculos de todos los parámetros geométricos de los engranajes de tornillo sin fin, así como la fuerza y resistencia que puede tener el mecanismo, los cálculos son realizados de una manera rápida y una de las desventajas con la que cuenta es que no exporta el modelo calculado.

1.4. Metodología de desarrollo

La metodología para el desarrollo de *software* es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de *software* desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

"Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decidió hacer una variación de la metodología AUP (Proceso Unificado Ágil), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (Rodriguez Sanches, 2015)."

- **Fases:** La Metodología AUP variante UCI contiene tres fases las cuales son inicio, ejecución y cierre porque así se adapta mejor al ciclo de vida de los proyectos de la UCI. Para una mayor comprensión se muestra la figura A.1 del apéndice (ibíd.).

- **Disciplinas:** La AUP variante UCI propone siete disciplinas las cuales son modelado de negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación y pruebas de aceptación. Para la gestión de la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y

PMC (Monitoreo y control de proyecto). Para una mayor comprensión se muestra la figura A.3 del apéndice (Rodriguez Sanches, 2015).

- **Roles:** AUP propone 9 roles (Administrador de proyecto, Ingeniero de procesos, Desarrollador, Administrador de BD, Modelador ágil, Administrador de la configuración, *Stakeholder*, Administrador de pruebas y Probador), se decide para el ciclo de vida de los proyectos de la UCI tener 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros. Para una mayor comprensión se muestra la figura A.2 del apéndice (ibíd.).

Esta versión de AUP define cuatro escenarios en los que se puede ubicar el desarrollo de una aplicación de acuerdo a sus características, los cuales son (ibíd.):

- **Escenario 1:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que pueden modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta.
- **Escenario 2:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.
- **Escenario 3:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.
- **Escenario 4:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos.

Siguiendo la política de desarrollo de *software* de la institución, se define como metodología a emplear la AUP-UCI en el escenario número 4, debido a que para representar el contexto no se modela el negocio ni el dominio, solamente historias de usuario, porque se asume que al ser seleccionado se tiene plena confianza y experiencia en lo que se quiere implementar, es decir, se dominan las tecnologías de desarrollo, se dispone de poco tiempo y el equipo de desarrollo siempre va estar acompañado del cliente.

Por tanto, esta parte del negocio que se puede representar utilizando modelo de negocio o dominio, se conoce y domina con total plenitud, por lo que se obvia estas representaciones y se pasa directo a encapsular las funcionalidades de los requisitos mediante historias de usuarios.

1.5. Herramientas y lenguajes para el desarrollo

En la actualidad debido a la evolución de las tecnologías informáticas y lenguajes de desarrollo, ha sido posible acelerar la realización de nuevas técnicas computacionales, que satisfagan las necesidades del

hombre moderno, provocando que cada vez sea necesario el estudio de estas para la realización un sistema determinado.

A continuación, se presenta una descripción de las herramientas y lenguajes empleados en la realización de este módulo.

1.5.1. Visual Paradigm

Visual Paradigm es una herramienta Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) que brinda un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente y la documentación de los programas. Esta herramienta emplea el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés).

Se caracteriza por (S. PRESSMAN, 2003):

- Disponibilidad en múltiples plataformas (*Windows*, *Linux*,*MacOS*).
- Diseño centrado en casos de uso y enfocado al negocio que generan un *software* de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, con diferentes especificaciones.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web.
- Varios idiomas.
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.

Se ha decidido su empleo para la confección de la propuesta de solución debido a que esta herramienta genera códigos en varios lenguajes, posee una licencia gratuita y emplea el lenguaje de modelado visual UML que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*.

1.5.2. Open Cascade Technology

La Tecnología Open Cascade (OCCT, por sus siglas en inglés), es una plataforma de desarrollo de *software* que proporciona servicios para superficies 3D y modelado de sólidos, el intercambio de datos CAD, y la visualización. La mayor parte de la funcionalidad OCCT se encuentra disponible en forma de bibliotecas de C ++. La biblioteca puede ser aplicada en el desarrollo de *software* cuyo objetivo sea el modelado 3D (CAD), fabricación o medición de la Fabricación Asistida por Computadora (CAM, por sus siglas en inglés) o simulación numérica (CAE) . Es una tecnología de *software* libre que se puede distribuir y/o modificar bajo los términos de la Licencia Pública General de GNU (LGPL) versión 2.1, con excepción adicional (OpenCASCADETechnology, 2017).

Alternativamente, *Open CASCADE* puede ser utilizada bajo los términos de licencia comercial o acuerdo contractual. Está diseñada para ser altamente portátil y es conocida por trabajar en una amplia gama de plataformas (*UNIX*, *Linux*, *Windows*, *Mac OS X*, *Android*). La versión (7.0.0. beta) está certificada oficialmente en las plataformas *Windows* (IA-32 y x86-64), *Linux* (x86-64), *MAC OS X* (x86-64) y *Android* (4.0.4 ARMv7).

Edición Comunitaria de Open Cascade (OCE, por sus siglas en inglés) es una versión de OCCT en la que la comunidad del *software* libre aporta sus experiencias y recomendaciones de optimización a las versiones liberadas mediante foros o la página oficial de desarrollo de esta tecnología (http://dev.opencascade.org/).

Se decide el empleo en la presente investigación de *Open Cascade Community Edition* por todas las características antes mencionadas de la tecnología de *Open Cascade*, por ser de *software* libre y poseer un acelerado desarrollo por parte de la comunidad.

1.5.3. Lenguaje C++

Es un lenguaje de programación diseñado a mediados de los años 80 por *Bjarne Stroustrup*. Su creación fue con el objetivo de extender al lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. El nombre de C++ fue propuesto por *Rick Mascitti* en el año 1983 ya que hasta el momento se le conocía como C con clases.

"C++ es un lenguaje compilado, es decir, para correr un programa su código tiene que ser procesado por un compilador produciendo archivos objetos, los cuales son combinados por un vínculo a un ejecutable del programa. El ejecutable es creado por una combinación específica de *hardware* y *software*; no es portable, dígase, desde *Mac* a *Windows*. Cuando se habla de la portabilidad de programas en C++, usualmente significa portabilidad del código fuente; en otras palabras, el código fuente puede ser compilado satisfactoriamente y ejecutado en una variedad de sistemas. Esto implica que C++ es suficientemente expresivo y eficiente para los sistemas de mayor demanda de tareas de programación. C++ es usado por millones de programadores en cada dominio de aplicación. Billones de líneas de C++ están actualmente desarrolladas. Muchos sistemas operativos poseen partes esenciales escritas en C++, como Windows, iOS, Linux, entre otros. Este lenguaje no fue diseñado con la computación numérica en mente, sin embargo, muchos programas de ingeniería, numéricos y científicos son realizados en C++. Este puede coexistir con código escrito en otro lenguaje. C++ es soportado por una variedad de librerías y conjuntos de herramientas, tales como Boost[1], QT, OpenCV[2], entre otras (STROUSTRUP, 2013)."

Se elige este lenguaje de programación para el desarrollo de la propuesta de solución a causa de todas las características antes mencionadas y por su uso en la tecnología de Open Cascade.

[1]Líbreras portables.

[2]Procesamiento de imágenes en tiempo real

1.5.4. Framework QT

Fue creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas de QT las cuales son ampliamente usadas para desarrollar aplicaciones con interfaces gráficas de usuarios.

"Qt es un marco de desarrollo de aplicaciones multiplataforma basado en C++, dentro de las que se encuentran: *Linux*, *OS X*, *Windows*, *VxWorks*, *QNX*, *Android*, *iOS*, *BlackBerry*, *Sailfish OS* y otras." Qt está disponible bajo varias licencias: licencia comercial y para software libre con varias versiones de la GPL y la LGPL. Es mucho más que un conjunto de herramientas de Interfaces Gráficas de Usuario (GUI, por sus siglas en inglés). Proporciona módulos para el desarrollo multiplataforma en las áreas de redes, bases de datos, OpenGL, tecnologías web, sensores, protocolos de comunicación (Bluetooth, puertos serie), XML y procesamiento JSON, impresión, la generación de PDF, y mucho más (QtWiki, 2017)".

Se decide el uso de Qt en la presente investigación por ser un framework para el desarrollo de aplicaciones basado en C++ y por poseer una gran cantidad de módulos para disímiles tareas.

1.5.5. Sistema de control de versiones

Git es un sistema de control de versiones distribuido, libre y de código abierto, diseñado para manejar proyectos pequeños o muy grandes con rapidez y eficiencia. La característica Git que realmente hace que sea parte de casi todas Administración de Código Fuente (SCM, por sus siglas en inglés) es su modelo de ramificación que permite tener múltiples ramas locales que pueden ser totalmente independientes entre sí. La creación, la fusión y la supresión de esas líneas de desarrollo toma segundos. Esto significa que se pueden ejecutar acciones como las indicadas en (TORVALDS y HAMANO, 2010):

- **Conmutación de contexto sin fricción:** Crear una rama para probar una idea, comience varias veces, cambie de nuevo a la rama de donde se ramificó, aplique un parche, cambie de nuevo a la rama donde está experimentando y fíjarlo.
- **Reglas Basadas en el Rol:** Tener una rama que siempre contiene solo lo que va a la producción, otra que se fusiona en el trabajo para la prueba, y varias más pequeñas para el trabajo diario.
- **Flujo de trabajo basado en funciones:** Crear nuevas ramas para cada nueva función en la que esté trabajando, de modo que pueda cambiar sin problemas entre ellas y, a continuación, suprimir cada una de las ramas cuando se fusione en su línea principal.
- **Experimentación desechable:** Crear una rama para experimentar, darse cuenta de que no va a funcionar, y solo eliminarlo, abandonar el trabajo.

Se elige Git-Lab como sistema de control de versiones durante el desarrollo de la propuesta de solución, debido a que este sistema pertenece a la institución y presenta las mismas características antes mencionadas y por indicaciones del proyecto al que está integrado la presente investigación.

1.6. Conclusiones del capítulo

Del proceso de investigación realizado concluye:

1. El estudio de las herramientas comerciales permitió constatar que estas poseen buenas prestaciones, pero no pueden ser utilizadas en nuestro país de forma legal.
2. El análisis de las fuentes consultadas permitió constatar que no existe precedente de un módulo similar en aplicaciones de código abierto.
3. El estudio de las herramientas de cálculos permitió constatar que estas no dan la posibilidad de obtener el modelo del mecanismo de tornillo sin fin, aunque el MITCalc nos da salida directa para algunas aplicaciones comerciales.
4. La selección de las tecnologías y herramientas, C++ como lenguaje de programación, Qt como *framework* de desarrollo, Open CASCADE como biblioteca para modelado en tres dimensiones, Visual Paradigm como herramienta para el modelado de artefactos ingenieriles y el Git-Lab como sistemas de control de versiones, permitió obtener un marco de trabajo adecuado para el desarrollo de las funcionalidades requeridas.

CAPÍTULO 2

Descripción de la solución propuesta

En el presente capítulo se presenta la descripción de la propuesta de solución, que incluye un mapa conceptual, la descripción de las funcionalidades, la captura de requisitos, las historias de usuario, así como los aspectos del diseño arquitectónico y los patrones orientados a objetos empleados.

2.1. Mapa Conceptual

El mapa conceptual se utiliza para apoyar la descripción de la solución con el objetivo de hacer comprensible para cualquier persona la relación de los conceptos en él. En la figura 2.1 se puede observar todos los conceptos que son manejados en la solución.

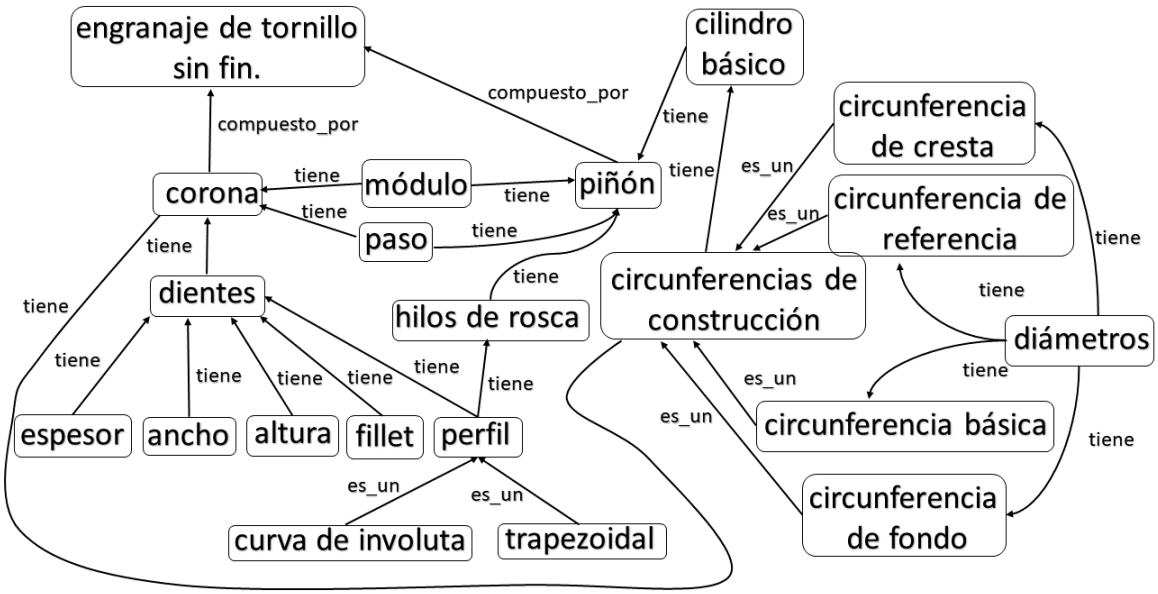


Figura 2.1. Mapa conceptual.

2.2. Descripción de las funcionalidades

El módulo será diseñado con las funcionalidades necesarias para modelar geométricamente los engranajes de tornillo sin fin y realizará los cálculos asociados a su diseño en correspondencia con las normas establecidas.

La interfaz poseerá una ventana principal con una pestaña que muestra los campos para introducir los datos iniciales, la franja superior cuenta con una ilustración representativa del engranaje de tornillo sin fin con los datos fundamentales para su creación.

En la pestaña principal se podrá elegir el tipo de pieza a visualizar, se introducirán los parámetros comunes del mecanismo como son, la relación de trasmisión, el módulo normal y el ángulo de presión; cada uno de los componentes del mecanismo de tornillo sin fin contienen parámetros para su creación, en el caso del tornillo son, números de hilos y el diámetro de referencia y en la rueda se le introducirá solo el número de dientes, estos parámetros están de acuerdo a la metodología de cálculo utilizada; la pestaña contendrá también tres botones principales, cancelar, calcular y modelar, los cuales al ser seleccionado ejecutan distintas funciones, por ejemplo, el calcular realiza todos los cálculos descritos en la norma usada y los muestra en la ventana principal; el modelar da paso a la creación del modelo en 3D en el visor; el cancelar cierra la ventana del acelerador del mecanismo de tornillo sin fin.

2.3. Requisitos

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información (S. PRESSMAN, 2006). A continuación se exponen los requisitos funcionales y no funcionales identificados.

2.3.1. Requisitos funcionales

Los aspectos funcionales permiten expresar una especificación detallada de las responsabilidades del sistema que se propone, además determinan una manera clara del comportamiento del mismo.

A continuación se listan los requisitos a cumplir por el componente:

RF 1. Insertar parámetros comunes de la rueda y tornillo sin fin.

RF 2. Insertar parámetros de la rueda.

RF 3. Insertar parámetros del tronillo sin fin.

RF 4. Seleccionar el tipo de engranaje de tornillo sin fin.

RF 5. Mostrar los resultados de los cálculos.

RF 6. Realizar modelado de la rueda.

RF 7. Realizar modelado del tornillo sin fin.

RF 8. Visualizar el ensamble del engranaje de tornillo sin fin.

2.3.2. Requisitos no funcionales

Los requerimientos no funcionales permiten definir las cualidades que el *software* debe tener, con el objetivo de brindar a los usuarios una mayor usabilidad, disponibilidad y rendimiento.

A continuación se listan los requisitos a cumplir por el componente:

Software:

RNF 1. Debe ejecutarse en cualquier distribución de GNU-Linux de 64 bits.

Restricciones de diseño e implementación:

RNF 2. Lenguajes C++, QT, bibliotecas de OCE comunitaria y la arquitectura en capa.

2.4. Historias de usuarios

Las historias de usuario son tarjetas de papel en las cuales el cliente describe brevemente las características funcionales o no con que el sistema va a contar. Permitiendo así que su utilización sea dinámica y flexible, con el objetivo de que en cualquier momento puedan ser modificadas, reemplazadas por otras más generales o específicas, o simplemente sustituirlas por unas nuevas; deben ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en poco tiempo (JEFFRIES R. y HENDRICKSON, 2001).

Las tablas 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, se muestran las historias de usuarios realizadas en la propuesta de solución.

Tabla 2.1. Historia de usuario #1

Historia de usuario	
Número: 1	**Nombre:** Insertar parámetros comunes de la rueda y tornillo sin fin
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 4 horas

Continúa en la próxima página

Tabla 2.1. Continuación de la página anterior

Descripción: 1- Objetivo: Permitir la inserción de los parámetros comunes del mecanismo de tornillo sin fin.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros comunes del mecanismo de tornillo sin fin hay que:

- Tener en cuenta los siguientes datos de entrada: relación de trasmisión,módulo normal y ángulo de presión.
- La relación de trasmisión debe ser del tipo de dato "*double*" y sus valores generalmente están entre 1-40.
- El módulo normal debe ser del tipo de dato "*double*" y sus valores generalmente están entre 1-25.
- El ángulo de presión debe ser del tipo de dato "*double*" cuando se introduce por la interfaz, luego para realizar los demás cálculos hay que convertirlo a grado, esto generalmente se hace multiplicando su valor por $\pi \div 180.0$ y sus valores típicos son 14.5,17.5,20,22.5,25,30.

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Estos datos tienen una relación directa con la selección de la opción "*Calculate*", esta permite que se inicialicen los demás parámetro de la geometría de este tipo de engranaje y se muestren en la ventana "*Result*".

Prototipo elemental de interfaz gráfica de usuario

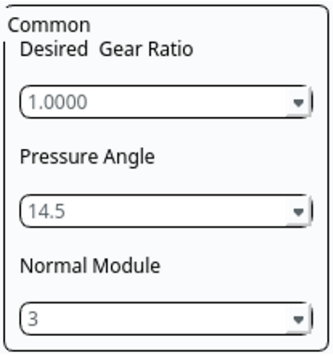


Tabla 2.2. Historia de usuario #2

Historia de usuario	
Número: 2	**Nombre:** Insertar parámetros del tornillo sin fin
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 3 horas

Continúa en la próxima página

Tabla 2.2. Continuación de la página anterior

Descripción:
1- Objetivo: Permitir la inserción de los parámetros del tornillo sin fin.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros del tornillo sin fin hay que:

- Tener en cuenta los siguientes datos de entrada: el diámetro de referencia del tornillo y los números de hilos.
- El diámetro de referencia del tornillo debe ser del tipo de dato "*double*" su valor no puede ser 0.
- Los números de hilos debe ser del tipo de dato "*double*" y sus valores por lo general están entre 1-6.

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Estos datos tienen una relación directa con la selección de la opción "*Calculate*",esta permite que se inicialicen los demás parámetros de la geometría de este tipo de engranaje y se muestren en la ventana "*Result*".

Prototipo elemental de interfaz gráfica de usuario:

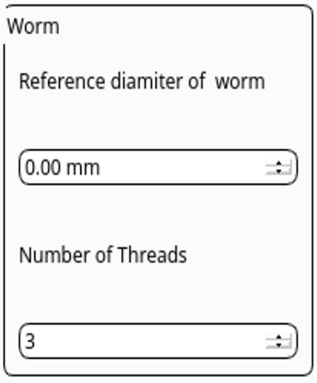


Tabla 2.3. Historia de usuario #3

Historia de usuario	
Número: 3	**Nombre:** Insertar parámetros de la rueda
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo: N/A	**Tiempo Real:** 0.5 horas

Continúa en la próxima página

Tabla 2.3. Continuación de la página anterior

Descripción:
1- Objetivo: Permitir la inserción de los parámetros para la construcción de la rueda.

2- Acciones para lograr el objetivo (precondiciones y datos): Para insertar parámetros de la rueda hay que:

- Tener en cuenta el número de dientes que tendrá la misma.
- El número de dientes debe ser del tipo de dato "*double*" y tiene que ser mayor igual a 19.

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y si selecciona la opción "*Calculate*" y estos tienen errores, se le debe advertir para que pueda rectificarlos.
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Estos datos tienen una relación directa con la selección de la opción "*Calculate*",esta permite que se inicialicen los demás parámetro de la geometría de este tipo de engranaje y se muestren en la ventana "*Result*".

Prototipo elemental de interfaz gráfica de usuario:

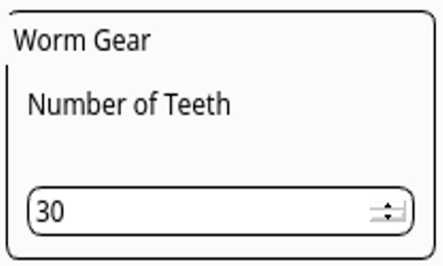


Tabla 2.4. Historia de usuario #4

Historia de usuario	
Número: 4	**Nombre:** Seleccionar el tipo de engranaje de tornillo sin fin
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 1.5 horas

Continúa en la próxima página

Tabla 2.4. Continuación de la página anterior

Descripción: 1- Objetivo: Permitir la selección del tipo de engranaje de tornillo sin fin.

2- Acciones para lograr el objetivo (precondiciones y datos): Para la selección del tipo de engranaje de tornillo sin fin hay que:

- Seleccionar el "*radioButton*" correspondiente a la pieza que se desee visualizar.

3- Flujo de la acción a realizar:

- El usuario debe seleccionar la pieza que quiere visualizar ejemplo, el tornillo sin fin, la rueda o el mecanismo de acoplado .
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones: Estos datos tienen una relación directa con la selección de la opción "*Calculate*", esta permite que se inicialicen los demás parámetros de la geometría de este tipo de engranaje y se muestren en la ventana "*Result*", también tiene una relación directa con la selección de la opción "*Model*", y esta permite que se visualice la pieza seleccionada.

Prototipo elemental de interfaz gráfica de usuario:

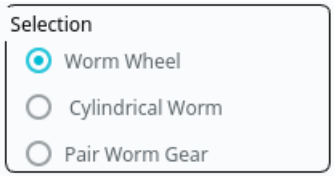


Tabla 2.5. Historia de usuario #5

Historia de usuario	
Número: 5	**Nombre:** Mostrar los resultados de los cálculos
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 3 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 4 horas

Continúa en la próxima página

Tabla 2.5. Continuación de la página anterior

Descripción:
1- Objetivo: Permitir que se muestren los cálculos de las propiedades de los engranajes de tornillo sin fin.

2- Acciones para lograr el objetivo (precondiciones y datos): Para que se muestren los cálculos de las propiedades de los engranajes de tornillo sin fin hay que:

- Tener en cuenta los siguientes datos de entrada: diámetro de referencia del tornillo, número de hilos, relación de trasmisión, módulo normal y ángulo de presión.
- Estos valores que se le introducen tiene que ser del tipo de dato " *doubel*".

3- Flujo de la acción a realizar:

- El usuario introducirá correctamente los datos necesarios y seleccionara la opción "*Calculate*", si tienen errores se le debe advertir al usuario para que pueda rectificarlos y luego poder ser mostrados en la ventana "*Result*".
- Si se selecciona el botón "*Cancel*" se cierra la ventana.

Observaciones:

Prototipo elemental de interfaz gráfica de usuario:

Tabla 2.6. Historia de usuario #6

Historia de usuario	
Número: 6	**Nombre:** Realizar modelado de la rueda
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 300 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 420 horas

Descripción:
1- Objetivo: Permitir realizar el modelado de la rueda

2- Acciones para lograr el objetivo (precondiciones y datos): Para realizar el modelado de la rueda hay que:

- Tener los datos introducido en la ventana principal sin errores.

3- Flujo de la acción a realizar:

- El usuario debe seleccionar el "*Worm Wheel*" para poder acceder a la función que crea la rueda.
- Selecciona el botón "Calculate" para que los datos introducidos en la interfaz, puedan inicializar los parámetros de la geometría.
- Seleccionar el botón "Model" se cierra la ventana principal y en el visor se muestra la rueda.

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 2.7. Historia de usuario #7

Historia de usuario	
Número: 7	**Nombre:**Realizar modelado del tornillo sin fin
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 240 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 300 horas
Descripción: **1- Objetivo:** Permitir realizar el modelado del tornillo sin fin **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar el modelado del tornillo sin fin hay que: • Tener los datos introducido en la ventana principal sin errores. **3- Flujo de la acción a realizar:** • El usuario debe seleccionar el "*Cyleindrical Worm*" para poder acceder a la función que crea el tornillo sin fin . • Selecciona el botón "Calculate" para que los datos introducidos en la interfaz, puedan inicializar los parámetros de la geometría. • Seleccionar el botón "Model" se cierra la ventana principal y en el visor se muestra el tornillo sin fin.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 2.8. Historia de usuario #8

Historia de usuario	
Número: 8	**Nombre:** Visualizar el ensamble del engranaje de tornillo sin fin
Programador: Leandro Texidor Basterrechea	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 15 horas

Continúa en la próxima página

Tabla 2.8. Continuación de la página anterior

Descripción: 1- Objetivo: Permitir visualizar el ensamble de la rueda con el tornillo en tres dimensiones **2- Acciones para lograr el objetivo (precondiciones y datos):** Para Visualizar el ensamble de la rueda con el tornillo en tres dimensiones hay que: • Tener los datos introducido en la ventana principal sin errores. **3- Flujo de la acción a realizar:** • El usuario debe seleccionar el "*Pair Worm Gear*" para poder acceder a las funciones que crean el tornillo sin fin y la rueda. • Selecciona el botón "Calculate" para que los datos introducidos en la interfaz, puedan inicializar los parámetros de la geometría del engranaje. • Seleccionar el botón "Model" se cierra la ventana principal y en el visor se muestra el engranaje de tornillo sin fin.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

2.5. Diseño

El proceso de diseño tiene asociado la decisión del tipo de arquitectura y los patrones de diseño que se emplean en un sistema, así como la confección de los distintos diagramas que favorezcan el trabajo en la fase de implementación.

2.5.1. Estilo y patrón arquitectónico del software

Un estilo arquitectónico determina el vocabulario de los componentes y relaciones que pueden ser utilizadas en instancias de este estilo determinado, con un conjunto de restricciones en las descripciones arquitectónicas.

"(...) 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 cambios fundamentales en la estructura del *software*, incluida una reasignación de funcionalidad del componente(...) (S. PRESSMAN, 2006)"

Para el desarrollo de *software* existen variedades de estilos y patrones arquitectónicos entre los que podemos encontrar: cliente/servidor, basado en componentes, arquitectura en capas (N-Layer), presentación

desacoplada, arquitectura en capas (N-Tier), arquitectura orientada al dominio (DDD) u orientada a objetos (OO) y bus de servicios(Mensajes), entre otras.

Para darle solución a este trabajo se escoge como estilo arquitectónico el de Llamada y Retorno, pues permite que un diseñador de *software* obtenga una estructura de programa que resulta relativamente fácil modificar y variar el tamaño. Miembros de esta familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los orientados a objeto y los jerárquicos en capas.

2.5.2. Arquitectura en Dos Capas

La arquitectura se encuentra organizada mediante un sistema de 2 capas, la Gui y APP. Este modelo soporta el desarrollo incremental del sistema, los cambios y la portabilidad, además cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades a una de ellas solo se ven afectadas las capas adyacentes. Entre las desventajas de este modelo encontramos que la estructuración de los sistemas puede resultar difícil y el rendimiento puede verse afectado, pues se puede incurrir en que una funcionalidad debe pasar por muchas capas para alcanzar la capa superior que solicitó su servicio (SitioOpenGLOverview, 2018).Se decide el empleo de esta arquitectura en la confección del módulo, porque soporta bien los cambios y el desarrollo incremental.

2.5.3. Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces.

Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP, por sus siglas en inglés) definidos en las funcionalidades son:

- **Experto:** La responsabilidad de la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para hacerlo (LARMAN y VALLE, 2003). Este patrón se evidencia en las clases "*WormWheel*"y *Screw* que se encargan de la creación de superficies.
- **Creador:** Ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos (ibíd.). Este patrón esta presente en "*DlgCreateWormGear*" y cuentan con la información necesaria para la creación de objetos tipo "*WormGear*'.
- **Polimorfismo:** Se usa cuando varía el tipo de alternativas o comportamientos relacionados; permite asignar la responsabilidad del comportamiento a los tipos en que varía el mismo (ibíd.). Se evidencia en la clase "*WormGear*" que reimplementa los métodos "*execute*" y "*mustExecute*"que hereda "*Feature*".
- **Alta cohesión:** Es una medida de cuán relacionadas o enfocadas están las responsabilidades de una clase (ibíd.). Se evidencia en "*WormGear*".

- **Bajo acoplamiento**:Asigna las responsabilidades a los objetos de modo se reduzca el impacto de los cambios y no incremente el acoplamiento, que es una medida de la fuerza en que las clases se relacionan (LARMAN y VALLE, 2003). Se evidencia en "*WormWheel*", "*Screw*".

Los patrones Gang of Four (GOF) definido en las funcionalidades son:

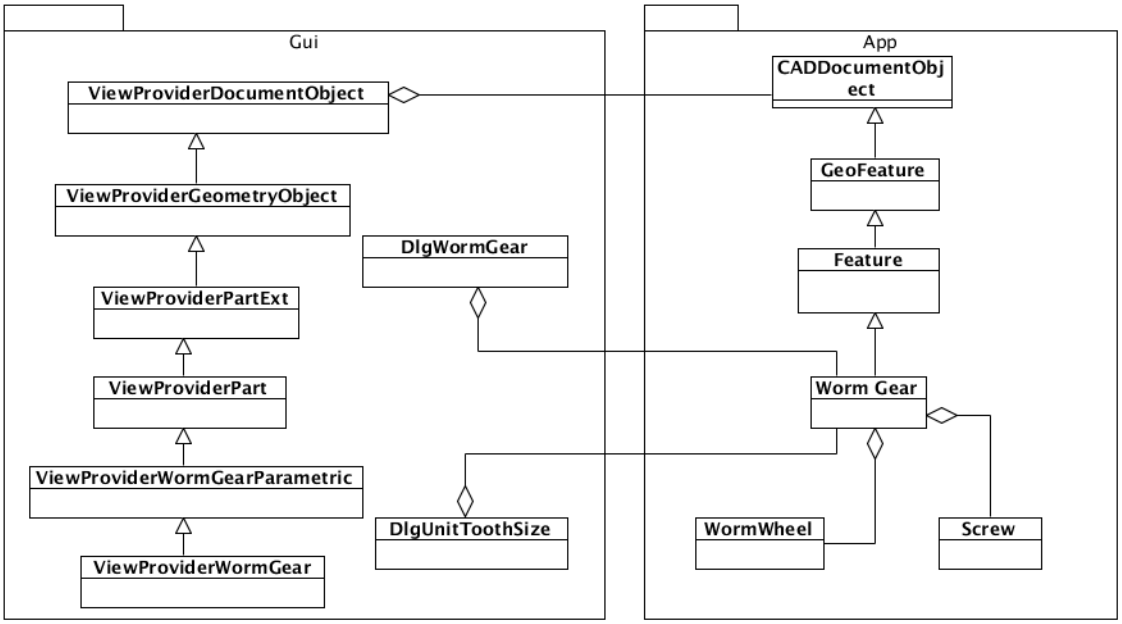
- **Observador:**Construye una dependencia entre un sujeto y sus observadores de modo que cada modificación del sujeto sea notificada a los observadores para que puedan actualizar su estado (ibíd.). Este patrón se evidencia con la función "*mustExecute*" la que permite que las propiedades de las piezas modeladas, puedan ser editadas.

2.5.4. Diagrama de clase del diseño

Un diagrama o modelo de clases en UML, es un tipo de diagrama de estructura estática que describe la organización de un sistema, mostrando las clases del mismo, sus atributos, operaciones (o métodos) y las relaciones entre los objetos (herencia, agregación, asociación, entre otras).

En la figura 2.2 se muestra el diagrama que responde a la necesidades del módulo de engranajes de tornillo sin fin.

Figura 2.2. Diagrama de clase del diseño.



2.5.5. Construcción de engranaje de tornillo fin fin

Metodología de cálculo.

Los mecanismos de tornillo sin fin son una pieza mecánica que trasmiten movimiento entre ejes que están ubicados perpendicularmente, mediante la interacción de dos piezas: la corona con dentado helicoidal y el piñón.

Creación de las circunferencias de construcción.

Para crear la rueda se necesitan cuatro circunferencia de construcción:

- **Circunferencia de Referencia.**
  Es la circunferencia guía de las demás circunferencias de construcción y siempre es tangente a la circunferencia de referencia del tornillo; se crea posicionando un punto como centro, en este caso el punto (0,0,0) y su diámetro se calcula por la siguiente expresión:

$$reference\ diameter\ of\ wheel = \frac{Z_2 \cdot m_n}{cos(\gamma)} \tag{2.5.1}$$

  Donde:
  Z_2: número de dientes.
  m_n : módulo normal.
  $cos(\gamma)$: coseno del ángulo de referencia del cilindro guía.

- **Circunferencia de cresta.**
  La circunferencia de cresta es la encargada de señalar el valor de altura al diente, cortando a su perfil y en algunos casos es tangente a la circunferencia básica del tornillo; se crea posicionando un punto como centro, en este caso el punto (0, 0, 0) y su diámetro se calcula por la siguiente expresión:

$$tip\ diameter\ of\ wheel = reference\ diameter\ of\ wheel + (2.0 \cdot addendum\ of\ wheel) + m_n \tag{2.5.2}$$

  Donde:
  m_n : módulo normal.
- **Circunferencia base.**
  La circunferencia básica sirve como base al trazado del perfil del diente y en algunos casos es tangente a la circunferencia de cresta del tornillo; se crea posicionando un punto como centro, en este caso el punto (0, 0, 0) y su diámetro se calcula por la siguiente expresión:

$$basic\ diameter\ of\ wheel = reference\ diamiter\ of\ wheel \cdot cos(\alpha) \tag{2.5.3}$$

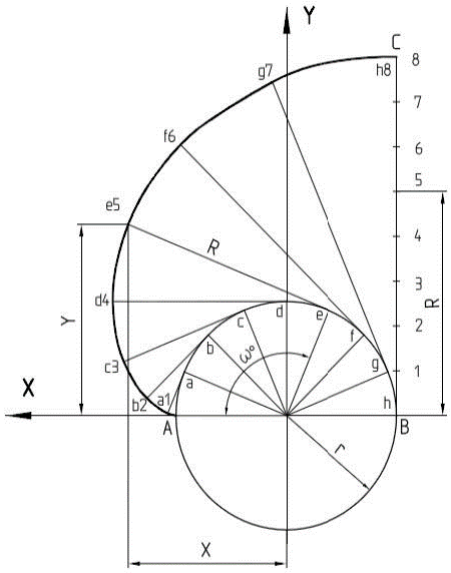
  Donde:
  α: ángulo de presión.

- **Circunferencia de Fondo.**
  La circunferencia de fondo es la encargada de representar el núcleo de la rueda; si el diámetro de la circunferencia de fondo es menor que el diámetro de la circunferencia básica, se traza una línea radial desde el inicio del perfil del diente en la circunferencia básica, hasta el centro de la circunferencia, para darle al diente el cuerpo bajo el perfil; por otro lado, si el diámetro de la circunferencia de fondo es mayor que el diámetro de la circunferencia básica, el perfil del diente se crea a partir de la circunferencia de fondo; se crea posicionando un punto como centro, en este caso el punto (0, 0, 0) y su diámetro se calcula por la siguiente expresión:

$$root\ diameter\ of\ wheel = throat\ diamiter - 2 \cdot tooth\ depth \tag{2.5.4}$$

- **Perfil del diente de la rueda.**
  El envolvente de circulo o involuta, es el perfil que utilizaremos para la construcción del diente de la rueda, se desarrolló mediante una curva que describe cualquier punto de una recta, que rueda sin resbalar sobre una circunferencia de radio *r* (construcción del hilo tenso que se desenrolla de un tambor cilíndrico). Ver imagen 2.3

Figura 2.3. Trayectoria del envolvente de circulo o Involuta.



Para crear el tornillo se necesitan tres circunferencia de construcción:

- **Circunferencia de referencia.**
  Es la circunferencia guía de las demás y siempre es tangente a la circunferencia de referencia de la rueda, se crea posicionando el punto como centro, en este caso el punto (0, *Distancia entre Centro*, 0) y su diámetro se introduce por la interfaz.

- **Circunferencia de cresta.**
  La circunferencia de cresta es la encargada de señalar el valor de altura al hilo de rosca, en algunos casos es tangente a la circunferencia básica de la rueda; se crea posicionando un punto como centro, en este caso el punto (0, *Distancia entre Centro*, 0) y su diámetro se calcula por la siguiente expresión:

$$tip\ diameter\ of\ worm = reference\ diamiter\ of\ worm + 2 \cdot addendum\ of\ worm \tag{2.5.5}$$

- **Circunferencia de fondo.**
  La circunferencia de fondo es la encargada de representar el núcleo de la rueda; el perfil del diente se crea a partir de la circunferencia de fondo; se crea posicionando un punto como centro, en este caso el punto (0, *Distancia entre Centro*, 0) y su diámetro se calcula por la siguiente expresión:

$$root\ diameter\ of\ worm\ = tip\ diameter\ of\ worm - 2 \cdot tooth\ depth \tag{2.5.6}$$

- **Construcción del núcleo de la rueda.**
  Para la creación del núcleo se necesitan utilizar las circunferencias antes construida y realizar una figura geométrica como la de la imagen 2.4 a la izquierda, la cual se puede crear de la siguiente manera:
  Este perfil se dibujara en el plano XY realizando cuatro punto y un arco.

  Funciones y topología de Open CASCADE que pueden ser utilizadas:

  - **gp_Pnt** crea un punto en tres dimensiones.
  - **gp_Pnt2d** crea un punto en dos dimensiones.
  - **gp_Plane** crea un plano en tres dimensiones en el espacio.
  - **gp_Ax2d** crea un eje representando en el eje X un objeto de referencia en el sistema de coordenada.
  - **gp_Circ2d** describe un circulo en dos dimensiones en el plano el mimo esta definido por un radio y una posición en el plano.
  - **Geom2d_Curve** es una clase abstracta que permite realizar una curva en el espacio en 2 dimensiones.
  - **GCE2d_MakeSegment** su constructor da la posibilidad de crear un segmento entre dos puntos en dos dimensiones.
  - **GCE2d_MakeArcOfCircle** su constructor da la posibilidad de crear un arco entre dos puntos estos puntos son obtenidos por la intercepción entre el circulo construido y los segmentos.
  - **TopoDS_Wire** es una topología de las biblioteca Open CASCADE.

  Después de tener construido el perfil pasaremos a convertirlo en cara y luego lo haremos un sólido mediante funciones de Open CASCADE, en la figura 2.4 se puede apreciar una representación de lo antes descrito.

Funciones y topología de Open CASCADE que pueden ser utilizadas:

- **TopoDS_Face** es una de las topología presente en las biblioteca de Open CASCADE, la cual tiene como objetivo hacer una cara de un *Wire* cerrado.
- **TopoDS_Shape** topología presente en las biblioteca de Open CASCADE, la cual tiene como objetivo hacer un solido; "*BRepPrimAPI_MakeRevol*" es una de las funciones que permite revolucionar un *Face* con una dirección y obtener un sólido.

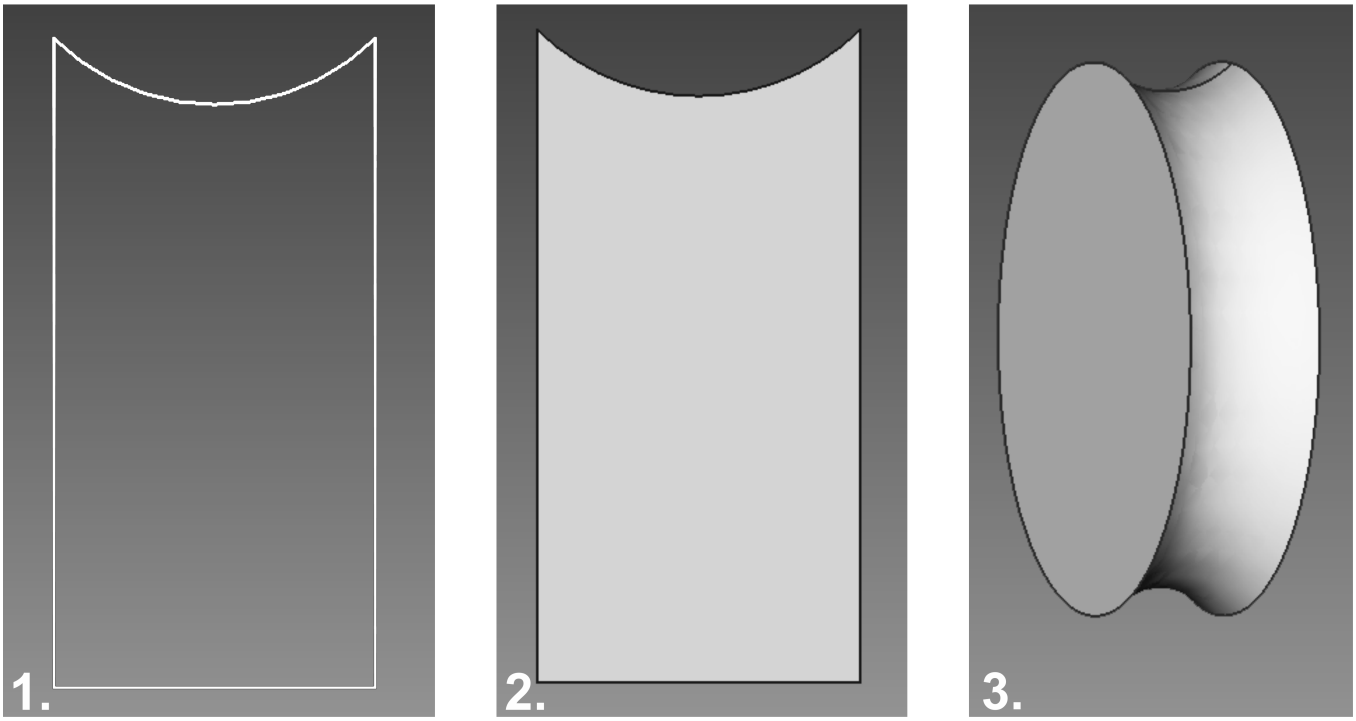


Figura 2.4. Secuencia de imágenes del núcleo.

- **Construcción de la hélice para colocar el diente.**
  Para su construcción se obtendrá el punto medio de la garganta del perfil y que es de donde van a salir las dos hélices con direcciones diferentes, pero al final la convertiremos en una sola para darle así la inclinación al diente. En la figura 2.5 se puede apreciar una representación de lo antes descrito.
- **Ubicación y construcción sólida del diente.**
  Después de haber creado el envolvente de círculo y la hélice que pasa por la garganta, se aplicarán algunas funciones de la biblioteca Open CASCADE para generar el diente en 3 dimensiones. La figura 2.5 se puede apreciar una representación de lo antes descrito.
  - Lo primero que se realizará será la ubicación del perfil en el punto medio de la hélice, que en este caso es el primer punto de las dos hélices construidas.

- Después de tener ubicado el perfil en el punto medio, se comprueba que el mismo sea tangente al núcleo en ese punto, para que no ocurran deformaciones; luego se pasa a realizar el "*Swep*" a lo largo de la hélice para obtener un diente en tres dimensiones.

Funciones de Open CASCADE que pueden ser utilizadas :

- **gp_Trsf** es una función que nos da la posibilidad de realizar transformaciones en el espacio tridimensional como traslación,rotación,escalar.
- **SetDisplacement** es una función que modifica una transformación y nos permite crear un nuevo sistema de coordenada.
- **BRepFill_PipeShell** es una función que permite realizar el "*Swep*" y en este caso se utiliza para que el diente quede solido.
- **SetRotation** al igual que el "*setDisplacent*" una función que se utiliza dentro de una transformación, pero esta sirve para rotar, en este caso se utilizó para rotar el diente solido 360 grados para conformar la corona.
- **BRepAlgoAPI_Cut** es una función que realiza una operación *boolean* para realizar un corte sobre dos sólidos.

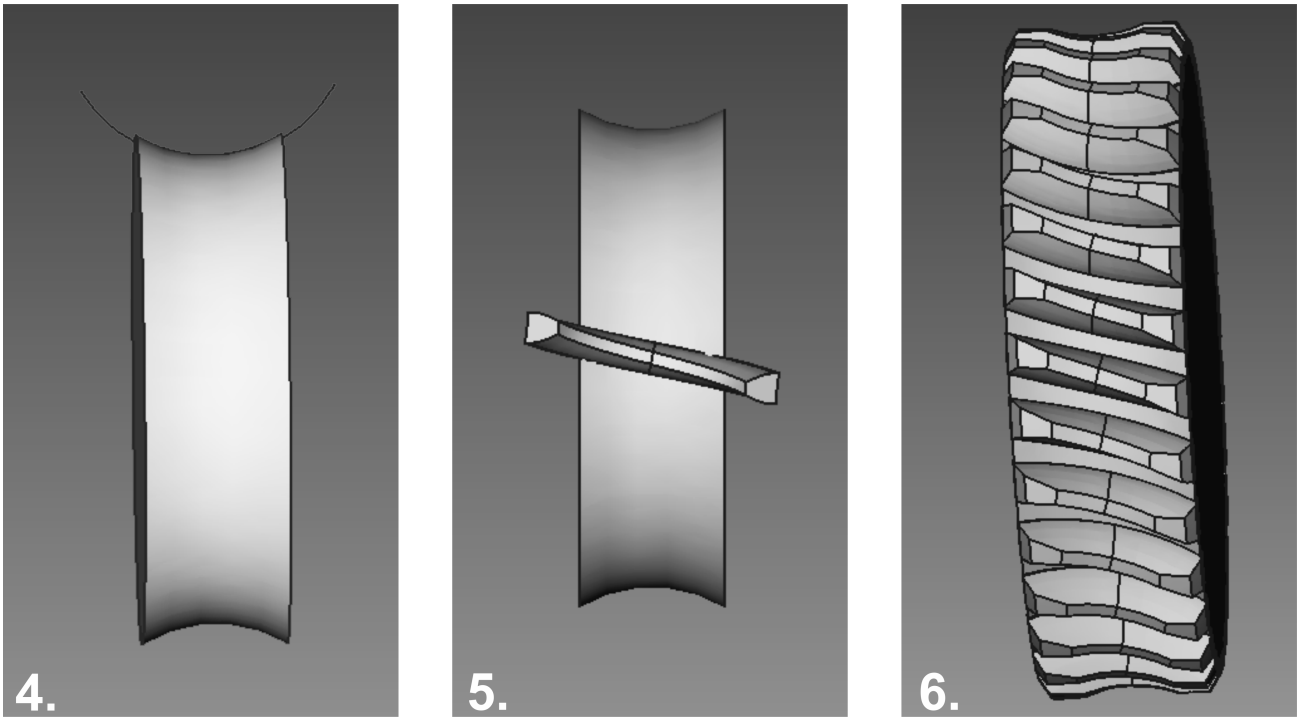


Figura 2.5. Secuencia del la creación de la corona

- **Construcción el núcleo del tornillo sin fin.**
  Debido a que el núcleo es cilíndrico y que se tienen las circunferencias de construcción, solo queda

aplicar una de las funciones de Open CASCADE.

Funciones de Open CASCADE que pueden ser utilizadas :

- **BrepPrimAPI_MakeCylender** es una de las funciones de figuras primitivas, esta en particular nos da la posibilidad de realizar un cilindro y recibe un eje, radio y una altura.

- **Perfil que pasará por los hilos del tornillo sin fin.**
  Se construirá un perfil trapezoidal, el cual pasará por cada una de las hélices que serán creada en el núcleo cilíndrico del tornillo sin fin, para poder conformar los hilos de rosca.
  Esto será posible creando cuatro puntos, que luego serán unidos mediante segmentos y utilizando las funciones trigonométrica y valores de la metodología de cálculo este perfil podrá ser construido correctamente. Ver imagen 2.6
  Ecuaciones de la metodologías que se utilizaron:

$$tooth\ depth = 2.25 \cdot m_n \tag{2.5.7}$$

$$addendum\ of\ worm = 1.00 \cdot m_n \tag{2.5.8}$$

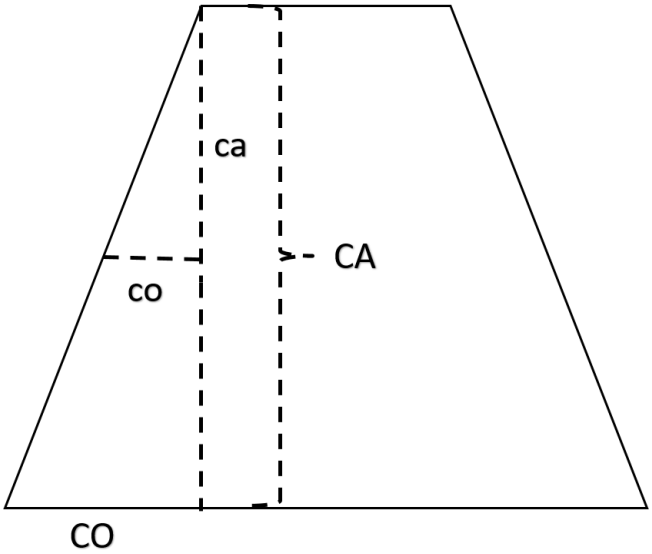
$$normal\ tooth\ thickneess\ of\ worm = \pi \cdot m_n \div 2.0 \tag{2.5.9}$$

  Donde:
  m_n : módulo normal.
  Funciones y topología de Open CASCADE que pueden ser utilizadas:

  - **gp_Pnt** crea un punto en tres dimensiones.
  - **TopoDS_Wire** es una de las topologías de la biblioteca Open CASCADE.
  - **gp_Trsf** es una función que nos da la posibilidad de realizar transformaciones en el espacio tridimensional, como traslación, rotación, escalar.
  - **BRepBuilderAPI_MakeWie** es una de las topologías de la biblioteca Open CASCADE.
  - **BRepBuilderAPI_MakeEdge** es una de las topologías de la biblioteca Open CASCADE, la cual tiene como función crear un borde.
  - **BRepBuilderAPI_Transform** construye un esquema para aplicar la geometría.
  - **BRepFill_PipeShell** es una función que permite realizar el "*Swep*", en este caso se utiliza para hacer solido los hilos de rosca.
  - **TopoDS_Compound** es una función que permitir unir varios "'*Shape*, para poder trabajar con ellos.

- **Construcción de las hélices del tornillo sin fin.**
  Después de tener construido el cilindro de la base del tornillo sin fin y el perfil del diente, se pasa a realizar las demás hélices que tendrá el mismo, y así crear las roscas; en las bibliografías consultadas

Figura 2.6. Representación de la trigonométrica.



$$ca = tooth\ depth - addendum\ of\ worm \tag{2.5.10}$$

$$co = co \cdot \tan(\alpha) \cdot \pi \div 180.0 \tag{2.5.11}$$

$$CO = tooth\ depth \cdot \tan(\alpha) \cdot \pi \div 180.0 \tag{2.5.12}$$

Donde: α: ángulo de presión.

se plantea que el tornillo sin fin debe tener de 1 a 6 hilos de rosca. Este proceso fue realizado de la siguiente manera:

- se crea una hélice con paso igual a la siguiente expresión:

$$pitch\ helix\ of\ worm = \pi \cdot m_n \cdot number\ of\ thereads \div cos(reference\ cylinder\ angle) \tag{2.5.13}$$

- se definen la cantidad de vuelta con la siguiente expresión:

$$truns = face\ width\ of\ worm \div pitch\ helix\ of\ worm \tag{2.5.14}$$

Funciones y topología de Open CASCADE que pueden ser utilizadas:

- **TopoDS_Wire** es una topología de la biblioteca Open CASCADE.
- **BRepBuilderAPI_MakeWire** es una topología de la biblioteca Open CASCADE.
- **BRepBuilderAPI_MakeEdge** es una de las topologías presente en las biblioteca de Open CAS-

CADE, la cual tiene como función crear un borde.

- **BRepFill_PipeShell** es una función que permite realizar un "*Swep*", y que los hilos de rosca queden solido.
- **BrepLib** convierte una curva en 2D en una 3D.
- **BRepBuilderAPI_Transform** construye un esquema para aplicar la geometría.
- **gp_Trsf** es una función que nos da la posibilidad de realizar transformaciones en el espacio tridimensional, como traslación, rotación, escalar.
- **SetRotation** es una de las función de una transformación; pero esta sirve para rotar, se utilizó para rotar la hélice alrededor del cilindro, para obtener la cantidad de hilos que el usuario desee, la expresión para el ángulo que se utilizo fue:

$$angle\ of\ rotation = 2.0 \cdot \pi \div number\ of\ thereads \tag{2.5.15}$$

2.6. Conclusiones del capítulo

1. La selección del estilo de *software* llamada y retorno, permitió que la estructura del programa sea relativamente fácil de modificar y variar su tamaño.
2. La selección de la arquitectura en capa permitió, que el sistema soporte bien los cambios y el desarrollo incremental.
3. La realización del diagrama de clase del diseño, permitió una visión más clara de las relaciones entre las clases y los patrones que serán empleados en la fase de implementación.
4. El diseño de las historias de usuario, permitió una visión más clara de los requisitos funcionales solicitado por el cliente.

CAPÍTULO 3

Evaluación de la solución propuesta

En el presente capítulo se abordan cada uno de los componentes que integran la etapa de implementación y prueba de la aplicación a desarrollar. Se expone el estándar de codificación, los diseños de los casos de pruebas, así como los resultados obtenidos del proceso de verificación de la calidad.

3.1. Implementación

La implementación es la elaboración de una aplicación o ejecución de un plan, idea o algoritmo en donde el usuario se involucra en su desarrollo.

La fase de implementación del *software* es la elaboración de una especificación técnica que posee como entradas los artefactos de la fase anterior (diseño), como: diagramas de clases, especificación de arquitectura y patrones a emplear en el sistema. En esta fase 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. Detalles técnico de la implementación

Siguiendo la descripción propuesta en el capítulo anterior, para el desarrollo de los engranajes de tornillo sin fin se emplearon los criterios conceptuales manejados durante el proceso de Investigación-Desarrollo, bibliotecas externas y fragmentos del código fuente de la aplicación FreeCAD, que permitieron: la creación de objetos para su posterior visualización; en las líneas que siguen se expone en forma resumida la descripción del módulo y clases principales que componen la solución:

Clase *WormGear*:

- Es la clase principal que interactúa directamente con la interfaz y le provee a las clases *Screw* y *WormWheel* la información necesaria para la construcción del engranaje de tornillo sin fin.
- **Contiene las siguientes funciones:**

- ***calculation***: se encarga de realizar todos los cálculos según la metodología utilizada.
- ***mustExecute***: esta clase hereda de "*Feature*" y es la encargada de que si el usuario quiere editar los valores de la geometría y ver esa edición en el visor
- ***execute***:esta clase hereda de "*Feature*" es la encargada de visualizar la pieza en el visor
- ***loadWormGear***:

- Esta clase tiene una relación de agregación con las clases *Screw* y *WormWheel*.
- Esta clase recibe los siguientes parámetros de la interfaz, module, números de dientes y de hilos, ángulo de presión y diámetro de referencia del tornillo.

Clase *Screw*:

- Es la clase que se encarga de construir el tornillo sin fin.
- Recibe los siguientes valores por parámetro, module, números de dientes y de hilos, ángulo de presión y diámetro de referencia del tornillo.
- **Contiene las siguientes funciones:**
  - ***cylindricalScrew***: es la función que crea el tornillo cilíndrico.

Clase *WormWheel*:

- Recibe los siguientes valores por parámetro, module, números de dientes y de hilos, ángulo de presión y diámetro de referencia del tornillo.
- **Contiene las siguientes funciones:**
  - ***WormWheelMesh***: es la función que crea la corona para el conformar el engranaje de tornillo sin fin.
  - ***involuteProfile***: es la función que crea el perfil de involuta.

Clase *DlgCreateWormGear*:

- Es la clase en que se desarrolla todas las implementaciones referente a la interfaz.
- **Contiene las siguientes funciones:**
  - **on-cancelar-clicked**: es la implementación del botón "*cancel*" y es el encargado de cerrar la ventana principal.
  - **on-aceptar-clicked**: es la implementación del botón "*model*" y es el encargado de mandar a crear en el visor el engranaje de tornillo sin fin.
  - **on-calcularDesing-clicked** es la implementación del botón "*caculate*" y es el encargo, cuando se selecciona de mostrar todo los cálculos en la ventana "*Result*".

Para realizar el módulo de engranaje de tornillo sin fin se implementaron 5 clases , con 16 funciones para un total de 1511 lineas de códigos.

3.1.2. Estándar de codificación

Los beneficios que tiene la codificación estandarizada en la industria del desarrollo de *software* son diversos. Algunas empresas tienen estos documentos porque es un requisito para obtener certificaciones, pero se ha observado que los ingenieros de *software* tienden a no tomarlos en cuenta al momento de hacer su trabajo. Se puede inferir que este descuido se debe a la falta de información de los desarrolladores de los beneficios de la codificación estandarizada, situación que puede corregirse desde su formación profesional. No obstante, lo anterior, pocos son los libros de texto de programación de computadoras que abordan estos temas para acostumbrar a los estudiantes, y futuros ingenieros de *software*, a adherirse a estos estándares.

Los estándares de codificación son pautas a cumplir por el código realizado en determinado sistema, con el objetivo de garantizar que todos los participantes lo puedan entender en menor tiempo y que el código en consecuencia se pueda mantener; sea fácil de transformar para añadirle nuevas características o modificar las existentes, depurar errores, o mejorar el rendimiento. En la imagen siguiente se muestra el estándar de codificación de la solución.

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{
	cuerpo de la clase
	}
	class FooFirst{
	cuerpo de la clase
	}

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() <Tipo dato retorno> funcionCompuesta() <Tipo dato retorno>funcionDobleCompuesta()

Figura 3.1. Continuación en la siguiente página

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; <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;
	x == y;
	idFuncion.miembro();
	idFuncion->miembro();
	array[];

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 :
	<estructura control>(condición){
	tarea a ejecutar
	}
	Para los bucles while, for, do while y otros:
	<bucle>(condiciones){
	tarea a ejecutar
	}

Comentarios en el código según el estándar de C++.	
Comentarios pequeños.	/* comentario sencillo */
Otros comentarios.	/* *Comentario */
Comentario de versión, descripción de clase y otras características de la clase o paquete.	/* ********** *Comentario amplio * ********** */

Figura 3.2. Estándar de codificación (E. SHIGLEY y R. MISCHKE, 1996),(Documentation, 2018)

3.1.3. Diagrama de componente

Un diagrama de componente representa como es dividido en componentes un sistema de *software*; muestra dependencias entre los componentes, que 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.

Estos diagramas pueden contener paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías (Cillero, 2018).

En la figura 3.3 se muestra el diagrama de componentes de la solución.

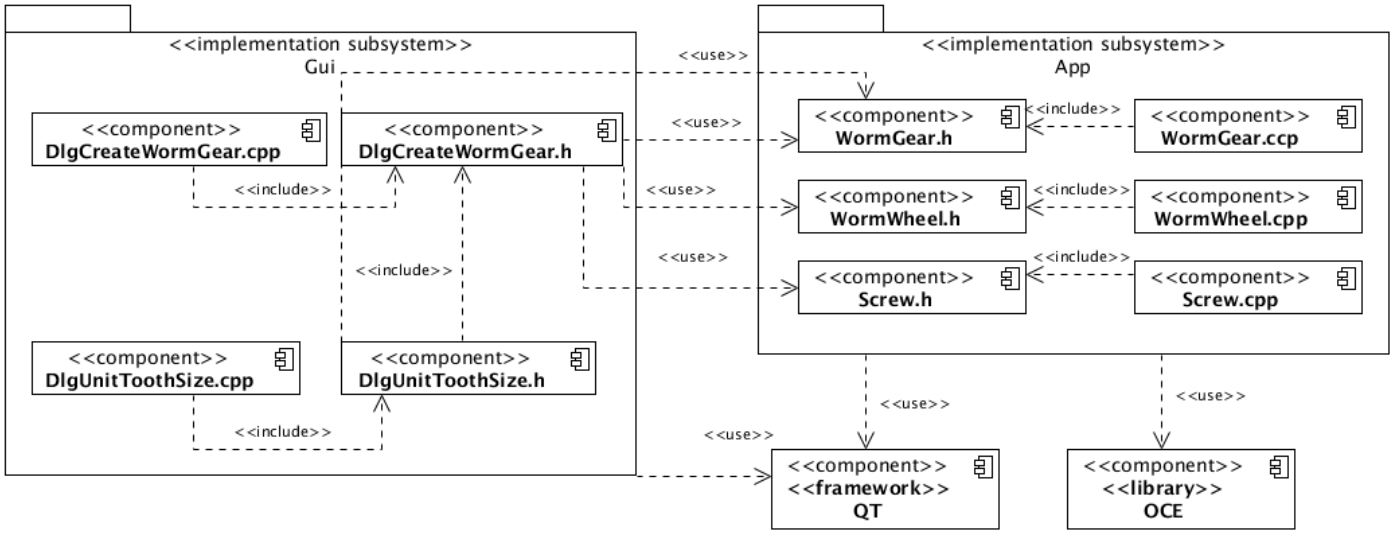


Figura 3.3. Diagrama de componentes

3.2. Pruebas

Durante las fases anteriores de definición y de desarrollo, se intenta construir el *software* partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, llegan las pruebas donde se crean una serie de casos de prueba que intentan destruir el *software* construido.

Sommerville en su libro Ingeniería del *software* plantea que el proceso de prueba de *software* tiene dos objetivos distintos (SOMMERVILLE, 2006).

- Demostrar al desarrollador y al cliente que el *software* satisface sus requisitos. Esto significa que debería haber al menos una prueba para cada requerimiento o característica que se incorporará a la entrega del producto.

- Descubrir defectos en el *software* en el que el comportamiento de este es incorrecto, no deseable o no cumple su especificación. La prueba de defectos está relacionada con la eliminación de todos los tipos de comportamientos del sistema no deseables, tales como caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrupción de datos.

La metodología AUP-UCI define tres etapas para la realización de las pruebas (Rodriguez Sanches, 2015).

- **Pruebas internas:** se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de pruebas ejecutables para automatizar las pruebas.
- **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables[1] de los proyectos antes de ser entregados al cliente para su aceptación.
- **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el *software* está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el *software* fue construido.

3.2.1. Método de prueba

Las pruebas en sí mismas deben mostrar un conjunto de características que logren la meta de encontrar la mayor cantidad de errores con el mínimo esfuerzo. Para llevar a cabo este objetivo, se emplearán los dos métodos de prueba (S. PRESSMAN, 2006).

- **Prueba de caja blanca:** se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del *software* y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:
    - Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
    - Revisen todas las decisiones lógicas en sus lados, verdadero y falso.
    - Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.
    - Revisen estructuras de datos internas para garantizar su validez
- **Prueba de caja negra:** se refiere a las pruebas que se llevan a cabo en la interfaz del *software*, examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del *software*. Intentan encontrar errores en las siguientes categorías:
    - Funciones incorrectas o faltantes.

[1] Los productos entregables son aquellos que se entregan oficialmente al cliente como parte del desarrollo en fechas previamente acordadas (TORO y JIMÉNEZ, 2000)

- Errores de interfaz.
- Errores en las estructuras de datos o en el acceso a bases de datos externas.
- Errores de comportamiento o rendimiento.
- Errores de inicialización y terminación.

Para validar el correcto funcionamiento del núcleo se tienen las pruebas unitarias, las pruebas funcionales y las pruebas de aceptación realizadas por el cliente.

3.2.2. Diseño de caso de pruebas

Para validar los requisitos funcionales de la aplicación se diseñan casos de prueba a partir de las historias de usuario, con el objetivo fundamental de encontrar la mayor cantidad posible de deficiencias existentes en las funcionalidades implementadas. En la tabla 3.1 se muestra el caso de prueba correspondiente al requisito mostrar los resultados de los cálculos.

Tabla 3.1. Diseño de Casos de Prueba de la Historia de Usuario " Mostrar los resultados de los cálculos".

Descripción general			
Permitir la visualización de los resultados del cálculo de las propiedades los engranajes de tornillo sin fin.			
SC 1 Mostrar los resultados de los cálculos			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción mostrar los resultados del cálculo de las propiedades del engranaje de tornillo sin fin.	Selecciona la opción "*Calculate*".	La opción muestra en la ventana "*Result*" todos los cálculos geométricos.	Accelerators/ WormGear /Calculate.
EC 1.2 Verificarlos cálculos	Selecciona la opción "*Calculate*".	Se mostraran mensajes de error hasta que los datos no estén correctos y luego se podrán visualizar los valores en la ventana "*Result*"	Accelerators/ WormGear /Calculate.

3.2.3. Prueba unitarias

Para la realización de las pruebas unitarias se utiliza el "***Qt Test***", un *framework* para realizar pruebas unitarias a aplicaciones y bibliotecas basadas en Qt. El "***Qt Test***" proporciona todas las funcionalidades comúnmente encontradas en los *framework* de pruebas, así como extensiones para probar interfaces gráficas de usuario(QtTest, 2018). Los resultados de la ejecución de las pruebas se muestran en figura 3.4.

Código fuente 3.1. Código para ejecutar ***Qt Test***

```
#include <QtTest>
#include <typeinfo>
#include <bits/stdc++.h>
#include  <Worgear.h>;
using namespace std;
class TestUnitTest : public QObject
{
    Q_OBJECT
private Q_SLOTS:
    void calculationAddendum();
    void calculationAngle();
    void calculationReferenceDiameterWheel();
    void calculationCenterDistance();
    void calaculationCoefficientOfProfileShift();
    void calculationToothDepth();
    void calculationTipDiameter();
    void calculationThroatDiameter();
    void calculationThroatSurfaceRadius();
    void calculationRootDiameter();
    void calculationdb();
    void calculationfacewidthOfWorm();

private:
    WormGear *wormgear = new WormGear( 3.0 , 2.0 , 30.0, 20.0 * M_PI
        /180.0, 44.0 );
};

  void TestUnitTest::calculationAddendum(){
    wormgear->calaculationAddendum();
    QVERIFY2(wormgear->getAddendumWorm()== 1.* 3.,"No coincide
        AddendumWorm");
    QVERIFY2(wormgear->getAddendumWheel() == 3.0 ,"No coincide
        AddendumWheel");}

   void TestUnitTest::calculationAngle(){
    wormgear->calculationAngle();
    QVERIFY2(wormgear->getReferenceCylinderLeadAngle()== asin((3.0 *
        2.0) / (44.0)),"No coincide Reference Cylinder LeadAngle");
    QVERIFY2(wormgear->getHelixAngle() == (M_PI/2)-asin((3.0 *  2.0) /
        (44.0)) ,"No coincide HelixAngle");}

   void TestUnitTest::calculationReferenceDiameterWheel(){
    wormgear->calculationReferenceDiameterWheel();
    QVERIFY2(wormgear->getReferenceDiameterWheel()== 30. * 3.0/cos(asin
```

```
    ((3.0 *  2.0) / (44.0))),"No coincide Reference Diameter Wheel")
    ;}

void TestUnitTest::calculationCenterDistance(){
 wormgear->calculationCenterDistance();
 QVERIFY2(wormgear->getCenterDistance()== (( 44. + 30. * 3.0/cos(
    asin((3.0 *  2.0) / (44.0))) /
  2.0),"No coincide Center  Distance");}

 void TestUnitTest::calaculationCoefficientOfProfileShift(){
 wormgear->calaculationCoefficientOfProfileShift();
 QVERIFY2(wormgear->getCoefficientOfProfileShift()==(2.0 * (( 44. +
    30. * 3.0/cos(asin((3.0 *  2.0)
 / (44.0))) / 2.0) - 44.-30. * 3.0/cos(asin((3.0 *  2.0) / (44.0)))
    )/ (2.0 * 3.0) ,"No coincide Coefficient Of Profile Shift ");}

void TestUnitTest::calculationToothDepth(){
 wormgear->calculationToothDepth();
 QVERIFY2(wormgear->getToothDepth()==2.25 * 3. ,"No coincide Tooth
    Depth");}

void TestUnitTest::calculationTipDiameter(){
 wormgear->calculationTipDiameter();
 QVERIFY2(wormgear->getTipDiameterWorm()==44. + (2.0 * 3.),"No
    coincide Tip Diameter Worm");
 QVERIFY2(wormgear->getTipDiameterWheel()==30. * 3.0/cos(asin((3.0 *
      2.0) / (44.0))) + (2.0 *
 3.)+3.,"No coincide Tip Diameter Wheel");}

void TestUnitTest::calculationThroatDiameter(){
 wormgear->calculationThroatDiameter();
 QVERIFY2(wormgear->getThroatDiameter()==30. * 3.0/cos(asin((3.0 *
    2.0) / (44.0))) + (2.0 * 3.)," no coincide Throat Diameter");
    }

void TestUnitTest::calculationThroatSurfaceRadius(){
 wormgear->calculationThroatSurfaceRadius();
 QVERIFY2(wormgear->getThroatSurfaceRadius()==(44. / 2.0) - 3.," no
    coincide Throat Surface Radius");
 }

 void TestUnitTest::calculationRootDiameter(){
 wormgear->calculationRootDiameter();
 QVERIFY2(wormgear->getRootDiameterWheel()==30. * 3.0/cos(asin((3.0
    *  2.0) / (44.0))) + (2.0 *
```

```
3.)-(2.0 *2.25 * 3.)," no coincide Root Diameter Wheel");
QVERIFY2(wormgear->getRootDiameterWorm()==44. + (2.0 * 3.)-(2.0
    *2.25 * 3.)," no coincide Root Diameter Worm");
}

void TestUnitTest::calculationdb(){
wormgear->calculationdb();
QVERIFY2(wormgear->getdb()== 30. * 3.0/cos(asin((3.0 *  2.0) /
    (44.0))) * cos(20* M_PI/180. * M_PI/180.),"no coincide db");
}

void TestUnitTest::calculationfacewidthOfWorm(){
 wormgear->calculationfacewidthOfWorm();
 QVERIFY2(wormgear->getFaceWidthOfWorm()==M_PI * 3. * ( 4.5 + 0.02 *
     30),"no coincide con facewidth  Of Worm");}

QTEST_APPLESS_MAIN(TestUnitTest)
#include "qtest.moc"
```

Figura 3.4. Iteración con ***Qt Test***

```
Starting /home/leandro/Documents/University/ProyectoCAD/Codigos/build-qtest-Desktop-Debug/qtest...
********* Start testing of TestUnitTest *********
Config: Using QtTest library 5.5.1, Qt 5.5.1 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC
5.4.0 20160609)
PASS   : TestUnitTest::initTestCase()
PASS   : TestUnitTest::calculationAddendum()
PASS   : TestUnitTest::calculationAngle()
PASS   : TestUnitTest::calculationReferenceDiameterWheel()
PASS   : TestUnitTest::calculationCenterDistance()
PASS   : TestUnitTest::calaculationCoefficientOfProfileShift()
PASS   : TestUnitTest::calculationToothDepth()
PASS   : TestUnitTest::calculationTipDiameter()
PASS   : TestUnitTest::calculationThroatDiameter()
PASS   : TestUnitTest::calculationThroatSurfaceRadius()
PASS   : TestUnitTest::calculationRootDiameter()
PASS   : TestUnitTest::calculationdb()
PASS   : TestUnitTest::calculationfacewidthOfWorm()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 14 passed, 0 failed, 0 skipped, 0 blacklisted
********* Finished testing of TestUnitTest *********
/home/leandro/Documents/University/ProyectoCAD/Codigos/build-qtest-Desktop-Debug/qtest exited with code 0
```

3.2.4. Resultados de las pruebas

La realización de pruebas funcionales permitió identificar la presencia de no conformidades, las cuales se listan en la tabla 3.2

Tabla 3.2. No conformidades detectadas en las pruebas

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	La relación de trasmisión no funciona.	Media	Resuelta
2	RF 4	Los "*radioButtom*" no permiten la selección.	Alta	Resuelta

3	RF 5	Los valores de los cálculos no se muestran.	Media	Resuelta
4	RF 5	No se actualizan los valores de los cálculos al cambiar un valor.	Baja	Resuelta
5	RF 5	Si hay errores en algunos de los campos, si modifico los valores a otros correctos, siguen marcándose en color rojo.	Baja	Resuelta
6	RF 5	No se muestran las unidades de medidas.	Baja	Resuelta

Al concluir cada una de las iteraciones planificadas para el desarrollo de la propuesta de solución, fueron realizadas las pruebas pertinentes. En la figura 3.5 se muestra el resultado obtenido:

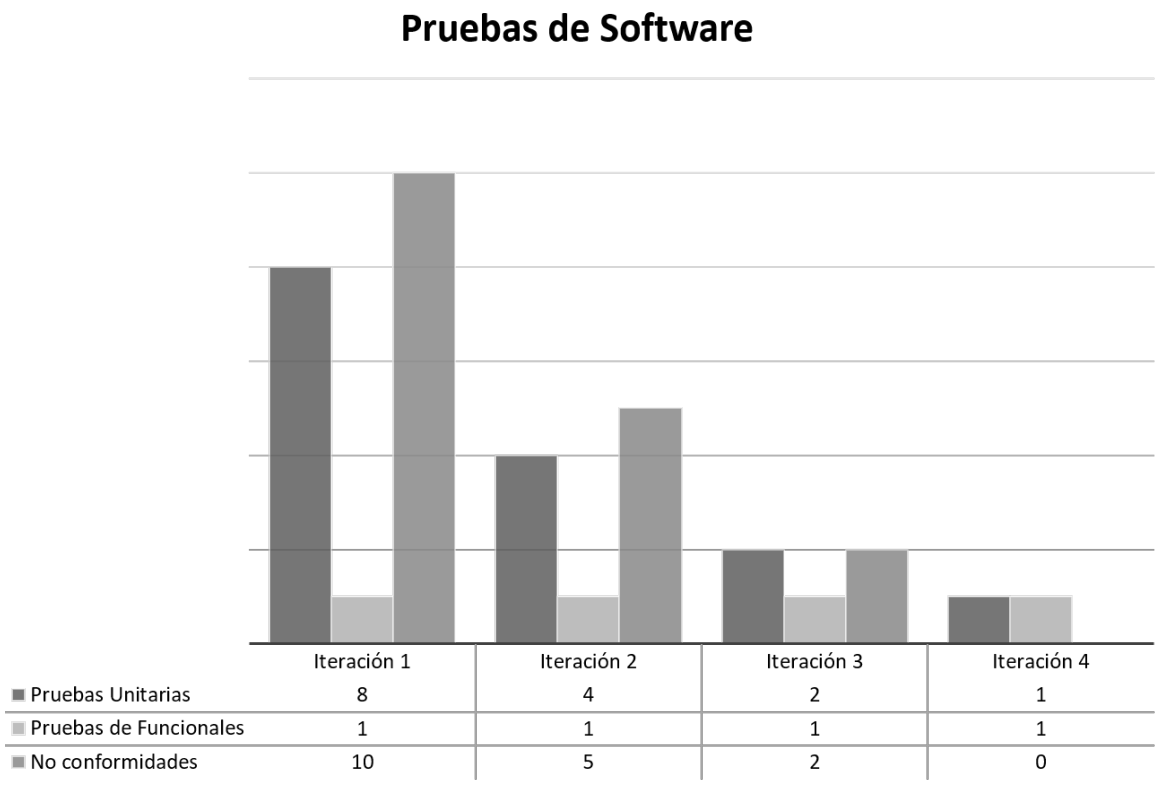


Figura 3.5. Resultados de las pruebas aplicadas al *software* en las distintas iteraciones

En la primera iteración se realizó una prueba de aceptación, ocho unitarias y quedaron pendientes diez pruebas por realizar en las demás iteraciones; en la segunda iteración se realizó una prueba de aceptación, cuatro unitarias y quedaron pendientes para las próximas iteraciones cinco pruebas; en la tercera iteración se realizó una prueba de aceptación, dos pruebas unitarias y quedaron pendientes para la última iteración dos pruebas; en la cuarta iteración se realizó una prueba unitaria y una prueba de aceptación, cumpliéndose todas las pruebas de manera satisfactoria sin ninguna pendiente, dando por terminado dicho proceso de pruebas.

3.3. Conclusiones

1. La utilización de la técnica de caja negra, permitió identificar un total de 17 no conformidades, las cuales fueron corregidas garantizando una mejor calidad de la solución.
2. La realización del diagrama de componente, permitió dividir el componente en partes modulares, remplazables y sustituible que encapsularan las implementaciones del software.

Conclusiones

De la investigación realizada se concluye:

1. El desarrollado del módulo permite acelerar el proceso de diseño del engranaje de tornillo sin fin, reduciendo de esta manera el tiempo de modelado con respecto a procedimientos tradicionales.
2. Las implementaciones de las funcionalidades del módulo permiten el modelado de rasgos complejos en la topología del engranaje, así como la posibilidad de modelar la corona o el tornillo sin fin de forma independiente.
3. La caracterización de los sistemas de código abierto permitió identificar las potencialidades y limitaciones para el modelado de engranajes de tornillo sin fin, lo que permitió proyectar la idea de un módulo para acelerar el diseño de dicho mecanismo.

Recomendaciones

A partir del trabajo realizado y luego de haber analizado los resultados obtenidos se sugieren los siguientes elementos a tener en cuenta para un futuro perfeccionamiento:

1. Incluir los cálculos de resistencia por la norma utilizada, así como por las demás normas existentes.
2. Incluir la selección de materiales normalizado por las diferentes normas de cálculo.

Glosario

flancos (Es la cara interior del diente es su zona de rozamiento.). 8

framework (Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.). 21

GPL (Garantiza la libertad de compartir y modificar todas las versiones de un programa que se encuentre bajo esta.). 20

JSON (Es un formato de intercambio de datos ligero.). 20

LGPL (Es un tipo de licencia que garantiza la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios.). 19, 20

OpenGL (Es un entorno de desarrollo para aplicaciones gráficas en 2D y 3D.). 20

superficies sean conjugadas (Existen dos tipos de conjugación.

- **Geométrica:** Vincula la geometría de la fresa madre con la del tornillo, exigiendo la coincidencia entre el diámetro y el ángulo de referencia del helicoide de ambos.
- **De perfil:** Exige la coincidencia de perfiles entre la fresa madre que elabora los dientes de la rueda y el tornillo que engranará con la misma.

). 8

tribología (Ciencia que estudia la fricción, desgaste y la lubricación que existe durante el contacto en superficies sólidas en movimiento.). 8

XML (Es un formato de texto simple y muy flexible.). 20

Acrónimos

CAD Diseño Asistido por Computadoras. 3–5, 13, 15, 16, 19
CAE Ingeniería Asistida por Computadora. 4, 15, 19
CAM Fabricación Asistida por Computadora. 19
CASE Ingeniería de Software Asistida por Computadora. 18
CATIA Aplicación interactiva tridimencional asistida por computadora. 13

GOF Gang of Four. 33
GRASP Patrones Generales de Software para Asignación de Responsabilidades. 32
GUI Interfaces Gráficas de Usuario. 20

LGPL Licencia Pública General Reduccida. 15

MCAD Diseño Asistido por Computadoras orientado a la Mecánica. 15

OCCT La Tecnología Open Cascade. 19
OCE Edición Comunitaria de Open Cascade. 19, 24

SCM Administración de Código Fuente. 21
SIPII Soluciones informáticas para la ingeniería y la industria. 1, 4

UML Lenguaje Unificado de Modelado. 19

Referencias bibliográficas

Autodesk (2017). *SEMCO: Autodesk Inventor*. URL: http://www.semco.com.pe/web/curso_autodesk_inventor.jsp (vid. págs. 14, 15).

Cillero, Manuel (2018). *diagrama-de-componentes*. URL: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/ (vid. pág. 47).

Documentation, QGIS (2018). *Guía de desarrolladores para QGIS*. URL: https://docs.qgis.org/2.18/es/docs/developers_guide/codingstandards.html (vid. pág. 46).

E. SHIGLEY, JOSEPH y CHARLES R. MISCHKE (1996). *Standard Handbook of Machine*. Ed. por Design. 2da Edición (vid. pág. 46).

eFunda (2017). *About Us*. URL: http://www.efunda.com/about/about.cfm (vid. pág. 17).

EULALIA IZARD, ANAYA (2017). *Ampliación de cálculo de elementos de máquinas.* URL: https://www.academia.edu/8585046/resortes_mec%C3%A1nicos (vid. pág. 14).

G. Budynas, Richard y J. Keith Nisbett (2011). *Shigley's Mechanical Engineering Design* (vid. pág. 11).

JEFFRIES R., ANDERSON y A. HENDRICKSON (2001). *Extreme Programming Installed.* (Vid. pág. 25).

LARMAN y VALLE (2003). *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Ed. por Pearson Educación. Fuera de colección Out of series. ISBN 978-84-205-3438-1. (vid. págs. 33, 34).

Michalec, George (2015). *Kohara Gear Company of Japan* (vid. pág. 11).

MITCalc (2017). *MITCalc - Mechanical, Industrial and Technical Calculations.* URL: http://www.mitcalc.com/ (vid. pág. 16).

OpenCASCADETechnology (2017). *Open CASCADE Technology: Overview.* URL: http://dev.opencascade.org/doc/overview/html/index.html#OCCT_OVW_SECTION_1 (vid. pág. 19).

QtTest (2018). *QtTest*. URL: http://doc.qt.io/qt-5/qttest-index.html (vid. pág. 49).

QtWiki (2017). *About Qt - Qt Wiki*. URL: https://wiki.qt.io/About_Qt (vid. pág. 21).

Radzevich, Stephen P. (2013). *Theory of GEARING Kinetics, Geometry, and Synthesis* (vid. pág. 5).

Riegel, Juergen y Werner Mayer (2017). *About FreeCad - FreeCad Documentation.* URL: http://www.freecadweb.org/wiki/index.php?title=About_FreeCAD (vid. pág. 16).

Rivero Llerena, Gabriel (2002). «Determinación del factor de contacto entre los flancos del engranaje por tornillo sin fin con perfil derivado del cono.» Tesis doct. Instituto superior politécnico Jose Antonio Echeveria, Facultad de ingeneria mecánica. (vid. págs. 7-9).

Rodriguez Sanches, Tamara (2015). *Metodología de desarrollo para la actividad productiva de la UCI* (vid. págs. 17, 18, 48).

S. PRESSMAN, ROGER (2003). *Ingeniería de Software. Un enfoque práctico* (vid. pág. 19).

— (2006). *Ingeniería del software. Un enfoque práctico.* Vol. 6ta Edición (vid. págs. 24, 32, 48).

Siemens, PLM Software (2017). *Siemens PLM Software*. URL: http://www.plm.automation.siemens.com/es_sa/products/solidedge/ (vid. pág. 15).

sitio-oficial-novedge (2018). *sitio-oficial-novedge*. URL: https://novedge.com/ (vid. pág. 4).

SitioOpenGLOverview (2018). *SitioOpenGLOverview*. URL: https://www.opengl.org/about/#1 (vid. pág. 33).

SolidWorks-Corp. (2017a). *Diseño Eléctrico-SolidWorks*. URL: http://www.solidworks.es/sw/products/electrical-design/packages.htm (vid. pág. 14).

— (2017b). *Gestión de datos-SolidWorks*. URL: http://www.solidworks.es/sw/products/product-data-management/packages.htm (vid. pág. 14).

— (2017c). *Productos de visualización de SOLIDWORKS*. URL: http://www.solidworks.es/sw/products/visualization/solidworks-visualization-overview.htm (vid. pág. 14).

— (2017d). *Simulación-SolidWorks*. URL: http://www.solidworks.es/sw/products/simulation/packages.htm. (vid. pág. 14).

SOMMERVILLE (2006). *Ingeniería de Software*. Vol. 8va Edición (vid. pág. 47).

STROUSTRUP, BJARNE (2013). *The C++ Programming Lenguage* (vid. pág. 20).

Systèmes, Dassault (2017). *No conformidades detectadas con las pruebas de caja negra*. URL: http://www.3ds.com/es/productos-y-servicios/catia%20Tabla%208%20No%20conformidades%20detectadas%20con%20las%20pruebas%20de%20caja%20negra.%20/ (vid. pág. 13).

TORO y JIMÉNEZ (2000). *Metodología para la elicitación de requisitos de sistemas software.* Facultad de Informática y Estadística Universidad de Sevilla (vid. pág. 48).

TORVALDS, L. y J. HAMANO (2010). *Git:Fast version control system*. URL: http://git-scm.com (vid. pág. 21).

Generado con LaTeX: 20 de junio de 2018: 3:31pm

Apéndices

APÉNDICE A

Metodología AUP variante UCI

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes el proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Figura A.1. Fases de la metodología AUP variante UCI.

Roles AUP	Roles Variación AUP-UCI	Responsabilidades Roles (Variación AUP-UCI)
Administrador de proyecto	Jefe de proyecto	Las habilidades y competencias de cada uno de los roles definidos para la Variación de AUP-UCI se pueden consultar en mejoras.prod.uci.cu
	Planificador	
Ingeniero de procesos	Analista Arquitecto de información	
Modelador ágil	(Opcional)	
Desarrollador	Desarrollador	
Administrador de la configuración	Administrador de la Configuración	
Stakeholder	Stakeholder (Cliente/Proveedor de requisitos)	
Administrador de pruebas	Administrador de calidad	
Probador	Probador	
Administrador de BD	Arquitecto de software (Sistema)	
	Administrador de BD	

Figura A.2. Roles de la metodología AUP variante UCI.

Disciplinas AUP	Disciplinas Variación AUP-UCI	Objetivos Disciplinas(Variación AUP-UCI)
Modelado	Modelado de negocio	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes: 1- Casos de Uso del Negocio (CUN). 2- Descripción de Proceso de Negocio (DPN). 3- Modelo Conceptual (MC). A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos.
	Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio. Ver 4.2.1 Escenarios para la disciplina Requisitos.
	Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
Implementación	Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Prueba	Prueba interna	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.
	Prueba de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Prueba de aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
Gestión de configuración	Se cubren con el área de procesos PP PMC y CM que propone CMMI-DEV v1.3. Las mismas son áreas de procesos de gestión y soporte respectivamente.	Consultar en mejoras.prod.uci.cu los libros de procesos de cada una de estas áreas.
Gestión de Proyecto		
Entorno		

Figura A.3. Disciplinas de la metodología AUP variante UCI.

APÉNDICE B

Pruebas en herramientas comerciales

B.1. Iteración en el MITCalc

Worm gearing

			Worm	Gear	
i	**Check lines: 4.29;**				
ii	☑ **Project information**				
Author ☑		**Date** ☑		**Project No.** ☑	
John Doe		08/06/2018		111-111	
File name					
Gear4_02.xlsb					
Project Name ☑					
Worm Gear					
Basic Info ☑					
Worm gearing:; mn=4; z1=4; z2=160; n1=1500; n2=37,5; i=40; P1=2,14 [kW]					
Project Notes ☑					
Comments					

?	**Input section**				
1.0	☑ **Options of basic input parameters**				
1.1	Calculation units		SI Units (N, mm, kW...)		
1.2	Driven worm / worm gear		Gear		
1.3	Transferred power	Pw [kW]	2,137	3,000	
1.4	Speed (Worm / Worm gear)	n [/min]	1500,00	37,50	[/min]
1.5	Torsional moment (Worm / Gear)	Mk [Nm]	13,60	763,94	[Nm]
1.6	Transmission ratio / from table	i	40,00		
1.7	Actual transmission ratio / deviation	i	40,00	0,00%	
2.0	☑ **Options of material, loading conditions, operational and production parameters**				
2.0	Material identification according standard :		ANSI		
2.1	Material of the worm:	Alloy structural steel Gr.5120(ASTM A506) (Rm=785 MPa) case-hardened			
2.2	Material of the gear :	Bronze (centrifugal cast) CuSn12Ni2-C-GZ (DIN EN 1982) (Rm=300 MPa)			
2.3	Type of worm (profile type)		ZN (N) Wormgear		
2.4	Loading of the gearbox, driving machine - examples		A...Continuous		
2.5	Loading of gearbox, driven machine - examples		A...Continuous		
2.6	Type of lubrication		Worm oil bath lubrication		
2.7	Type of oil		Oil based on Polyglycols (PEG)		
2.8	Oil designation - selection		ISO VG - 220 (AGMA no 5)		
2.9	Kinematic viscosity for 40°C and 100°C	ν40,ν100	220,00	40,00	[mm^2/s]
2.10	Lubrication density at 15°C	ρoil15	1,060	[kg/dm^3]	
2.11	Roughness average value of the worm	Ra1	0,50	[microm]	
2.12	Application factor	KA	1,00	1,00 ☑	
2.13	Desired service life	Lh	25000	[h]	
2.14	**Requested coefficients of safety**				
2.15	Wear safety	SW	1,10	≥1.10	
2.16	Pitting safety	SH	1,00	≥1.00	
2.17	Worm deflection safety	Sδ	1,00	≥1.00	
2.18	Tooth strength safety	SF	1,10	≥1.10	
3.0	☑ **Parameters of the tooth profile**				
3.1	Addendum - Coefficient of the height of the tooth head	ha*	1,000	[modul]	
3.2	Unit head clearance	c*	0,250	[modul]	
3.3	Recommended coefficient of the root radius		0,38	[modul]	
3.4	Coefficient of the root radius	rf*	0,38 ☑	[modul]	
4.0	☑ **Design of a geometry of toothing**				
4.1	**Table of proper solutions**				
4.2	Check safety	SW ☑	SH ☑	Sδ ☑	SF ☑
4.3	Range of z1 from - to		1	3	
4.4	Range of q from - to		6	25	
4.5	Sort results according to parameter:		mass		

4.6	z1	z2	i	n2	q	m	DP	eta	gama	a	d1	d2	mass	SW	SH	Sd	SF	S
4.7	1	40	40.00	37.50	8.50	4.23	6.00	0.760	6.71	4.07	1.43	6.71	46.94	1.55	1.12	1.30	2.65	1.44

4.8	**Design of a geometry**					
4.9	Number of teeth Worm / Worm Gear	z1,z2	4	160		
4.10	Normal pressure angle	α	20,00	20	[°]	
4.11	Diameter quotient (q = d1 / m)	q	8,50	6 - 25	○	
4.12	Mean diameter of worm	d1	37,5766	~ 43,2	○	[mm]
4.13	Pitch angle	γ	25,2011	4	◉	[°]
4.14	Pitch direction		Right			
4.15	Module / Standardized value	mn	4,000			[mm]
4.16	Circular Pitch / Diametral Pitch	CP/DP	0,4947	6,3500		
4.17	Distance of left/right bearing (% of wheel diameter)	l1%,l2%	50,00	50,00		[% da2]

APÉNDICE C

Tablas de casos de pruebas

Tabla C.1. Diseño de Casos de Prueba de la Historia de Usuario " Insertar parámetros comunes de la rueda y tornillo sin fin".

Descripción general			
Permitir la inserción de los parámetros comunes de los engranajes de tornillo sin fin.			
SC 1 Insertar parámetros comunes de la rueda y tornillo sin fin.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Desired Gear Ratio*".	Selecciona el "*SpinBox*" "*Desired Gear Ratio*".	Brinda la posibilidad de seleccionar el valor de la "*Desired Gear Ratio*".	Accelerators/ WormGear /Common.
EC 1.2 Opción seleccionar "*Pressure Angle*".	Selecciona el "*SpinBox*" "*Pressure Angle*".	Brinda la posibilidad de seleccionar el valor de la "*Pressure Angle*".	Accelerators/ WormGear /Common.
EC 1.3 Opción seleccionar "*Normal Module*".	Selecciona el "*SpinBox*" "*Normal Module*".	Brinda la posibilidad de seleccionar el valor de la "*Normal Module*".	Accelerators/ WormGear /Common.

Tabla C.2. Diseño de Casos de Prueba de la Historia de Usuario " Insertar parámetros de la rueda".

Descripción general			
Permitir la inserción de los parámetros de la rueda del engranaje de tornillo sin fin.			
SC 1 Insertar parámetros de la rueda.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Number of Teeth*".	Selecciona el "*SpinBox*" "*Number of Teeth*".	Brinda la posibilidad de seleccionar el valor el "*Number of Teeth*".	Accelerators/ WormGear /Worm Gear.

Tabla C.3. Diseño de Casos de Prueba de la Historia de Usuario " Insertar parámetros del tronillo sin fin".

Descripción general

Permitir la inserción de los parámetros del tornillo sin fin.			
SC 1 Insertar parámetros del tronillo sin fin.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Reference diameter of worm*".	Selecciona el "*SpinBox*" "*Reference diameter of worm*".	Brinda la posibilidad de seleccionar el valor de la "*Reference diameter of worm*".	Accelerators/ WormGear / Worm.
EC 1.2 Opción seleccionar "*Number of Threads*".	Selecciona el "*SpinBox*" "*Number of Threads*".	Brinda la posibilidad de seleccionar el valor de la "*Number of Threads*".	Accelerators/ WormGear /Worm.

Tabla C.4. Diseño de Casos de Prueba de la Historia de Usuario " Seleccionar el tipo de engranaje de tornillo sin fin".

Descripción general			
Permitir la selección del tipo de pieza			
SC 1 Seleccionar el tipo de engranaje de tornillo sin fin.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Worm Wheel*".	Selecciona el "*RadiusButton*" "*Worm Wheel*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso la rueda.	Accelerators/ WormGear / Selection.
EC 1.2 Opción seleccionar "*Cylindrical Worm*".	Selecciona el "*RadiusButton*" "*Cylindrical Worm*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso el tornillo sin fin.	Accelerators/ WormGear /Selection.
EC 1.3 Opción seleccionar "*Pair Worm Gear*".	Selecciona el "*RadiusButton*" "*Pair Worm Gear*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este el acople entre el tornillo sin fin y la rueda dentada.	Accelerators/ WormGear /Selection.

Tabla C.5. Diseño de Casos de Prueba de la Historia de Usuario " Realizar modelado de la rueda".

Descripción general			
Permitir el modelado de las rueda			
SC 1 Realizar modelado de la rueda.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Worm Wheel*".	Selecciona el "*RadiusButton*" "*Worm Wheel*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso la rueda.	Accelerators/ WormGear/ Selection.

EC 1.2 Insertar los parámetros necesarios para la construcción.	Introducir parámetros como "*Number of Teeth*", "*Reference diameter of worm*", "*Number of Threads*", "*Pressure Angle*", "*Desired Gear Ratio*" y "*Normal Module*".	Brinda la posibilidad de introducir los parámetros necesarios.	Accelerators/ WormGear/ Parameter.
EC 1.3 Calcular los parámetros de la rueda.	Selecciona el botón"*Calculate*".	Brinda la posibilidad de calcular y visualizar los parámetros de la rueda dentada.	Accelerators/ WormGear/ Calculate
EC 1.3 Visualizar la rueda.	Selecciona el botón"*Model*".	Brinda la posibilidad visualizar el modelo en 3D en el visor.	Accelerators/ WormGear/ Model

Tabla C.6. Diseño de Casos de Prueba de la Historia de Usuario "Realizar modelado del tornillo sin fin".

Descripción general			
Permitir el modelado del tornillo sin fin			
SC 1 Realizar modelado del tornillo sin fin			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Cylindrical Worm*".	Selecciona el "*RadiusButton*" "*Cylindrical Worm*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso el tornillo sin fin.	Accelerators/ WormGear/ Selection.
EC 1.2 Insertar los parámetros necesarios para la construcción.	Introducir parámetros como "*Number of Teeth*", "*Reference diameter of worm*", "*Number of Threads*", "*Pressure Angle*", "*Desired Gear Ratio*" y "*Normal Module*".	Brinda la posibilidad de introducir los parámetros necesarios.	Accelerators/ WormGear/ Parameter.
EC 1.3 Calcular los parámetros del tornillo sin fin.	Selecciona el botón"*Calculate*".	Brinda la posibilidad de calcular y visualizar los parámetros del tornillo sin fin.	Accelerators/ WormGear/ Calculate
EC 1.4 Visualizar la el tornillo sin fin.	Selecciona el botón"*Model*".	Brinda la posibilidad visualizar el modelo en 3D en el visor.	Accelerators/ WormGear/ Model

Tabla C.7. Diseño de Casos de Prueba de la Historia de Usuario " Visualizar el ensamble del engranaje de tornillo sin fin".

Descripción general			
Permitir el modelado del tornillo sin fin y la rueda dentada			
SC 1 Visualizar el ensamble del engranaje de tornillo sin fin			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción seleccionar "*Pair Worm Gear*".	Selecciona el "*RadiusButton*" "*Pair Worm Gear*".	Brinda la posibilidad de seleccionar el tipo de pieza que va ser visualizada en este caso el ensamble de ambos.	Accelerators/ WormGear/ Selection.
EC 1.2 Insertar los parámetros necesarios para la construcción.	Introducir parámetros como "*Number of Teeth*", "*Reference diameter of worm*", "*Number of Threads*", "*Pressure Angle*", "*Desired Gear Ratio*" y "*Normal Module*".	Brinda la posibilidad de introducir los parámetros necesarios.	Accelerators/ WormGear/ Parameter.
EC 1.3 Calcular los parámetros del engranaje de tornillo sin fin.	Selecciona el botón"*Calculate*".	Brinda la posibilidad de calcular y visualizar los parámetros del engranaje de tornillo sin fin.	Accelerators/ WormGear/ Calculate
EC 1.4 Visualizar el ensamble del tornillo sin fin y la rueda dentada.	Selecciona el botón"*Model*".	Brinda la posibilidad visualizar el modelo en 3D en el visor.	Accelerators/ WormGear/ Model