Universidad de las Ciencias Informáticas
Facultad 4

Módulo para acelerar el diseño de engranajes cónicos de dientes rectos y helicoidales.

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

Autor: Carlos Alberto Gómez García.

Tutores: Dr. Augusto César Rodríguez Medina.

Ing. Gustavo García González.

La Habana, 2018

Nunca consideres el estudio como una obligación, si no como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

Dedicatoria

El presente trabajo se lo dedico a todas las personas que directa o indirectamente estuvieron involucrados en el desarrollo de mis estudios, en especial a mi familia, amigos y compañeros, que fueron la base para culminar esta meta.

Agradecimientos

El más sincero agradecimiento a todos los profesoes de la Universidad de las Ciencias Informáticas por brindarme la oportunidad de obtener una profesión y por ser una persona útil a la sociedad.

Mi más prefunda gratitud a todas las personas que directa o indirectamente estuvieron involucrados en el desarrollo de mis estudios, en especial a mi familia, amigos y compañeros, que fueron la base para culminar esta meta.

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 _____.

Carlos Alberto Gómez García.
Autor

Dr. Augusto César Rodríguez Medina.
Tutor

Ing. Gustavo García González.
Tutor

Resumen

El documento contiene los resultados de un trabajo en el que utilizando como base las normas (ANSI/AGMA 2005-D03), se obtuvo una solución informática de código abierto para automatizar el diseño y los cálculos de engranajes cónicos con dientes rectos o helicoidales. El resultado se logró empleando el marco de trabajo Qt, el lenguaje de programación C++ y haciendo uso de la tecnología Open CASCADE. Para guiar el proceso de desarrollo, se empleó como metodología el Proceso Ágil Unificado en su versión UCI. Con esta solución se eliminan problemas legales por posibles usos de sistemas propietarios de forma indebida y se da un paso más en la soberanía tecnológica.

Palabras clave: Engranajes cónicos, acelerador de diseño, diseño asistido por computadora.

Índice general

Índice de figuras

Índice de tablas

Introducción

Una de las carencias reconocibles en la ingeniería nacional está relacionada con los sistemas informáticos que actualmente se emplean en el diseño y la ingeniería; al revisar las tendencias de su desarrollo se aprecia que algunos de estos han incorporado módulos para acelerar el diseño de componentes complejos. Precisamente este documento agrupa los aspectos esenciales sobre el desarrollo de una solución informática para acelerar el proceso de diseño de engranajes cónicos con dientes rectos o helicoidales, empleando tecnologías y sistemas de código abierto.

El proceso de investigación y desarrollo asociado tiene sus bases en los trabajos del grupo de investigación Soluciones Informáticas para la Ingeniería y la Industria (SIPII) de la Facultad 4, que realiza investigaciones aplicadas y evalúa la factibilidad de implementar módulos informáticos para el Diseño Asistido por Computadoras (CAD, por sus siglas en inglés) con destino a los sectores de la industria nacional vinculados a la actividad de diseño.

El documento está estructurado en tres capítulos, en el primero se aborda la fundamentación teórica de la investigación, donde se incluye un estudio de los sistemas de diseño asistido por computadora homólogos, propiedades y características de los diferentes engranajes cónicos de dientes rectos y helicoidales, la metodología y herramientas para el desarrollo; en el segundo se incluye la propuesta de solución, se realiza el levantamiento de requisitos, la descripción del componente, la arquitectura, los patrones de diseño y las historias de usuarios; en el tercero se presentan elementos correspondientes a la implementación, como: los estándares de codificación y las pruebas realizadas al componente.

CAPÍTULO 1

Fundamentos de la investigación, metodología y tecnologías

En este capítulo se expone la situación problemática que dio origen al proceso de investigación. Se abordan los aspectos técnicos sobre engranajes cónicos de dientes rectos y helicoidales, así como sus normas cuyas formulaciones sirven de base al proceso de desarrollo; también se exponen los sistemas homólogos y se describen las diferentes tecnologías de desarrollo a utilizar.

1.1. Aspectos preliminares sobre la situación problémica y el proceso de investigación

Los engranajes son mecanismos utilizados para transmitir potencia mecánica entre las distintas partes de una máquina, están formados por dos ruedas dentadas, de las cuales a la mayor se le denomina corona y al menor piñón (R Marambedu, 2009).

Los engranajes tienen amplia aplicación en la maquinaria moderna y el procedimiento para su diseño en la actualidad, se lleva a cabo mediante computadoras y sistemas informáticos; en el caso de los engranajes cónicos de dientes rectos y helicoidales, que son los tratados en este trabajo.

Entre las aplicaciones típicas de los engranajes cónicos al campo automovilístico figuran las parejas planetarios-satélites del diferencial, el par piñón-corona del diferencial, más conocido como par cónico, y los pares de accionamiento del árbol de levas en cabeza, utilizados antes en lugar de las trasmisiones de cadenas.

Los engranajes cónicos efectúan la transmisión de movimiento entre ejes que se cortan en un mismo plano, generalmente en ángulo recto, por medio de superficies cónicas dentadas. Los dientes convergen en la intersección de los ejes. Son utilizados para efectuar reducción de velocidad con ejes en 90° (ibíd.).

Tras el surgimiento de los sistemas de diseño asistido por computadora, en la década del 50 (siglo XX) y su posterior desarrollo, han revolucionado las técnicas de dibujo, dieron un salto significativo con la aparición de las computadoras digitales (BALDASANO, 2016).

En la década del 90 del siglo XX empiezan a aparecer sistemas para el modelado en tres dimensiones, respondiendo a una nueva concepción en la filosofía de diseño, esto fue posible gracias a la implementación de

técnicas para el trazado paramétrico, que facilitaba la modificación de los objetos gráficos dinámicamente.
Debido al bloqueo económico impuesto por los Estados Unidos contra Cuba, sistemas comerciales como CATIA, Solid Edge, Solid Works, Autodesk Inventor, entre otros, destinados al cálculo y modelado de piezas mecánicas como engranajes cónicos, no se pueden utilizar de forma institucional debido a que al país se le prohíbe acceder a todo tipo de sistema que se considere de alta tecnología (BALDASANO, 2016).
Los sistemas informáticos comerciales, en general, se distribuyen bajo licencias de uso por un tiempo determinado, por lo que los ingenieros no son dueños del sistema, sino que reciben autorización para su empleo. Alguna de estas herramientas tienen módulos que automatizan la construcción de estructuras y piezas mecánicas complejas, como engranajes cónicos de dientes rectos y helicoidales, árboles escalonados, resortes helicoidales, entre otros. A estos módulos se le conoce, en algunas aplicaciones como aceleradores de diseño o generadores de componente, pero siempre su función es la misma, agilizar el trabajo al usuario del sistema durante el proceso de modelado.
También existen herramientas de código abierto para el diseño asistido por computadora, por ejemplo, LibreCAD es una aplicación para el diseño en dos dimensiones con cierta similitud al AutoCAD, el FreeCAD se aproxima más a sistemas como CATIA en sus funcionalidades para el modelado en tres dimensiones; estas no cuentan con ningún módulo que automatice la creación de componentes como los anteriormente mencionados.
Teniendo en cuenta la imposibilidad de adquirir sistemas de alta tecnología como las herramientas de diseño comerciales y la limitaciones de los sistemas de software libre como FreeCAD al no poseer un componente eficiente que acelere el modelado de engranajes cónicos de dientes rectos y helicoidales, como una forma de afrontar el problema existente en el país, en la Facultad 4 de la Universidad de las Ciencias Informáticas, existe un grupo de investigación llamado "Soluciones Informáticas para la Ingeniería y la Industria" **(SIPII)**, el cual se encuentra desarrollando una herramienta de diseño que satisfaga las necesidades de los ingenieros, arquitectos, diseñadores y otros vinculados a la actividad del diseño, que garantice la soberanía tecnológica y los disímiles usos en la industria de estos elementos mecánicos complejos, para esto se define como:

Partiendo de este análisis se formula el siguiente **problema de investigación: ¿Cómo modelar prototipos virtuales de engranajes cónicos de dientes rectos y helicoidales?**. El **objeto de estudio** de la investigación se centra en el modelado de entidades utilizando la tecnología Open CASCADE, teniendo como **campo de acción**, el modelado de engranajes cónicos de dientes rectos y helicoidales. Para solucionar el problema planteado se propone como **objetivo general**, desarrollar un componente para el modelado de engranajes cónicos de dientes rectos y helicoidales que funcione acoplado a una aplicación de diseño asistido por computadora basado en tecnologías de código abierto o de forma independiente mediante el empleo de tecnologías de código abierto. A partir del objetivo general definido, se derivan los siguientes objetivos específicos:

1. Diseñar las funcionalidades para el modelado de engranajes cónicos de dientes rectos y helicoidales.
2. Identificar las características y funcionalidades requeridas por un módulo para acelerar el modelado de engranajes cónicos de dientes rectos y helicoidales.

3. Obtener mediante implementación, un módulo informático para acelerar el modelado de engranajes cónicos.

Para dar cumplimiento a estos objetivos específicos se proponen las siguientes **tareas de la 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 cónicos de dientes rectos y helicoidales.
- Estudio de las funcionalidades de los módulos para el diseño de engranajes cónicos de dientes rectos y helicoidales, existentes en aplicaciones comerciales para los sistemas de diseño.
- 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 Open CASCADE 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 comerciales, 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.
- Modelado del flujo de datos del componente.
- Diseño de la interfaz gráfica del componente.

Implementación

- Implementación del componente destinado a la automatización del proceso de diseño de engranajes cónicos de dientes rectos y helicoidales.
- Integración del componente en la aplicación del proyecto.
- Realizar pruebas al componente (de las diferentes funcionalidades).

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

Métodos teóricos:

- **Análisis y síntesis**: se empleó para la construcción y desarrollo de la teoría, para la profundización en el tema y la sistematización del conocimiento.
- **Modelado**: se utilizó para la realización de los artefactos correspondientes al análisis, diseño e implementación del componente.

Métodos empíricos:

- **Observación**: permitió estudiar cómo funciona el proceso de creación de engranajes cónicos de dientes rectos y helicoidales con la tecnología Open CASCADE.

1.2. Generalidades sobre engranajes

Los engranes cónicos transmiten potencia, pero lo hacen exclusivamente entre ejes que no son paralelos, es decir, que se intersecan o se cortan entre sí. Generalmente, se emplean para ejes que forman 90º uno con respecto al otro, sin embargo, también pueden diseñarse para casi cualquier ángulo mayor o menor que el ángulo recto; a este tipo de engrane se le llama engrane cónico angular. Existe además el caso particular en el que el piñón y la rueda tienen la misma cantidad de dientes, por lo que los ángulos de la raíz de cada uno son de 45°; a estos engranes cónicos se les llama engranes de inglete (SHIGLE, 1979).

Dentro de los engranes cónicos encontramos tres tipos según la forma de sus dientes: recto, helicoidal o en espiral, y Zerol. El primero, es el más comúnmente usado y su forma es la más simple, siendo sus dientes cónicos tanto en espesor como en altura, con ángulos de presión usuales de 14,5° y 20°; el segundo, al igual que el engrane cilíndrico helicoidal, garantiza que más de un diente entre en contacto y que cubra extensamente a su correspondiente par, debido a la curvatura y oblicuidad de éstos; por último, y sólo como referencia, los engranes cónicos Zerol que son una forma especial de los engranes cónicos en espiral con dientes curvos cuyo ángulo medio de la espiral es cero (ibíd.).

Cinemáticamente son representados por conos de paso que giran sin deslizamiento a una relación de velocidad específica, y cuyo contacto es un elemento común llamado elemento de paso. En consecuencia, el elemento de paso es el eje instantáneo del movimiento relativo de un engrane respecto al otro. Los dos ejes y el eje instantáneo coinciden en el plano axial y se intersectan en el vértice (ver Figura 1.1) (ibíd.).

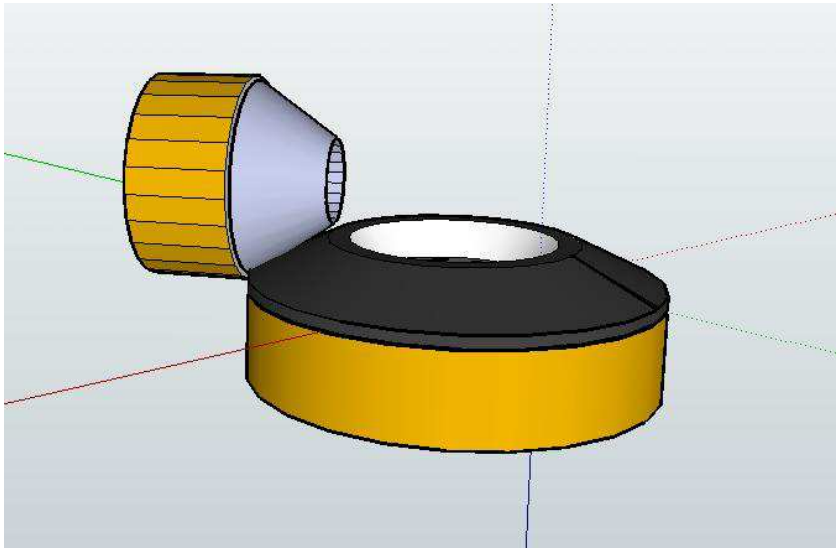


Figura 1.1. Estructura cónica primitiva de los engranajes.

1.3. Engranajes cónicos de dientes rectos

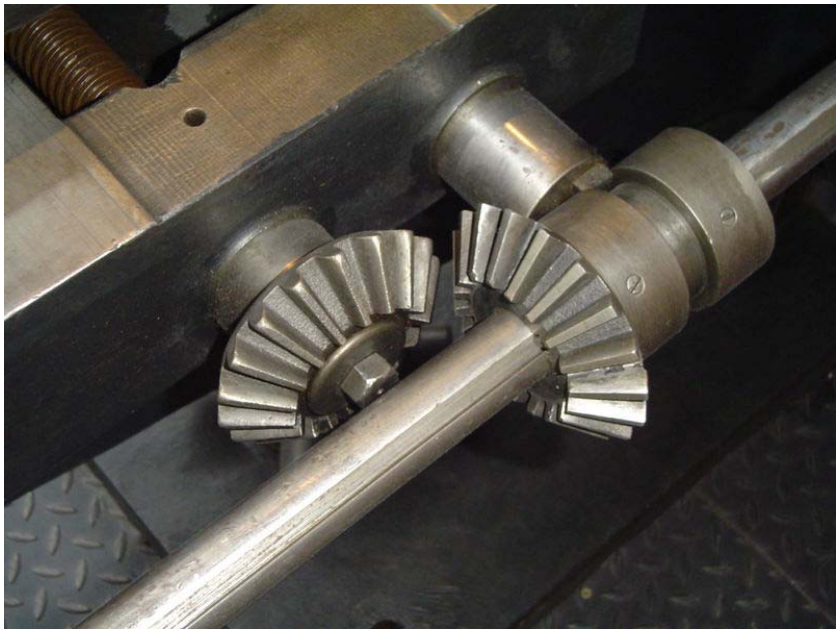


Figura 1.2. Engranajes cónicos de dientes rectos utilizados en la industria.

Efectúan la transmisión de movimiento de ejes que se cortan en un mismo plano, generalmente en ángulo recto, por medio de superficies cónicas dentadas. Los dientes convergen en el punto de intersección de los ejes. Son utilizados para efectuar reducción de velocidad con ejes en 90°. Estos engranajes generan más

ruido que los engranajes cónicos helicoidales. Los engranes cónicos rectos imponen cargas de empuje y radiales en sus soportes o cojinetes. Los ángulos de presión más comunes utilizados en los engranes cónicos rectos son de 14.5 y 20 grados (STANDARD, 2005) (ver Figura 1.2).

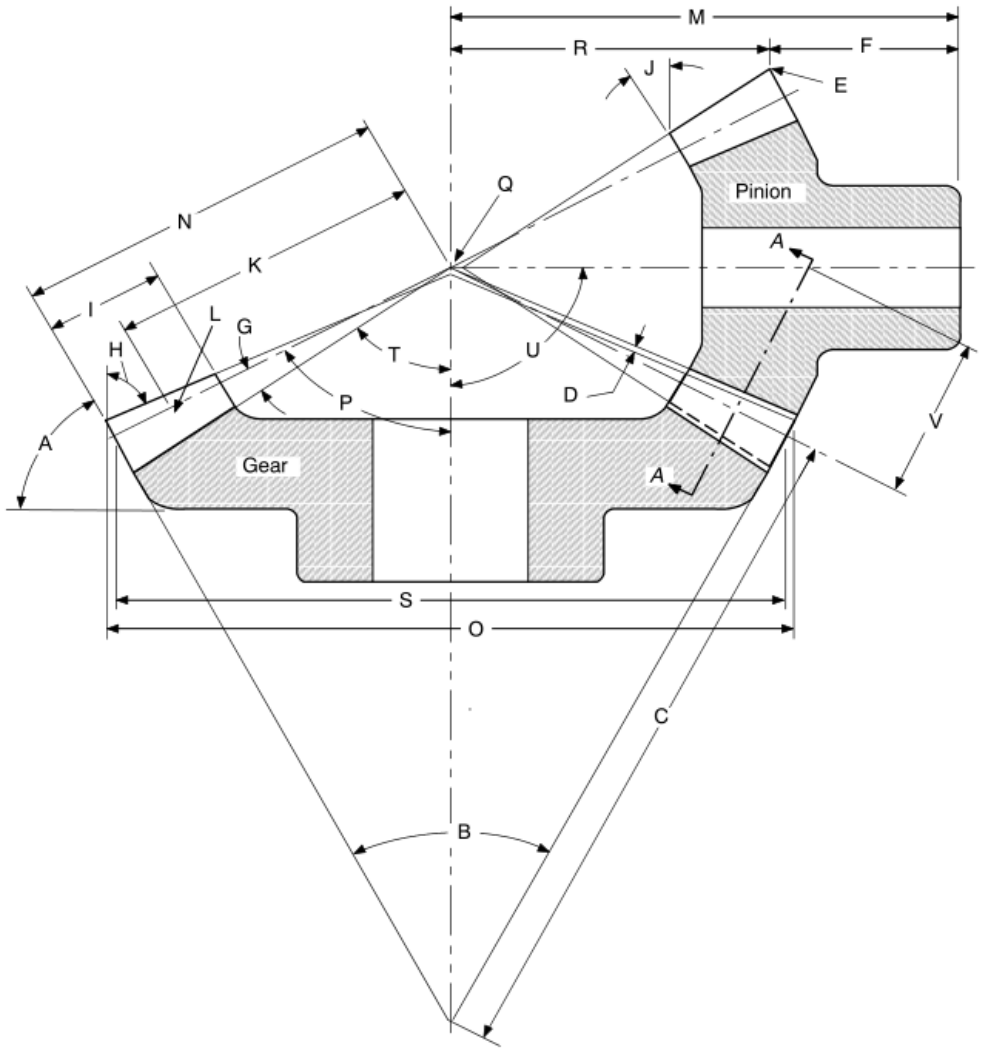
1.4. Engranajes cónicos de dientes helicoidales

Se utilizan para reducir la velocidad en un eje de 90°. La diferencia con el cónico recto es que posee una mayor superficie de contacto. Es de un funcionamiento relativamente silencioso. Además, pueden transmitir el movimiento de ejes que se corten. Los datos constructivos de estos engranajes se encuentran en prontuarios técnicos de mecanizado. Se mecanizan en fresadoras especiales y se utilizan en las transmisiones posteriores de camiones y automóviles (ibíd.) (ver Figura 1.3).

Figura 1.3. Engranajes cónicos helicoidales utilizados en la industria.

1.5. Fundamentación matemática para diseñar engranajes cónicos de dientes rectos y helicoidales

Para una mejor comprensión de los esquemas y formulaciones matemáticas que a continuación se exponen, es de gran importancia dar a conocer la tabla de símbolos con las terminologías de los engranajes cónicos de dientes rectos y helicoidales vigentes en la norma ANSI/AGMA 2005–D03,AMERICAN NATIONAL STANDARD (ibíd.).



A	Back angle	H	Face angle	P	Pitch angle
B	Back cone angle	I	Face width	Q	Pitch cone apex
C	Back cone distance	J	Front angle	R	Pitch cone apex to crown
D	Clearance	K	Mean cone distance	S	Pitch diameter
E	Crown point	L	Midface	T	Root angle
F	Crown to back	M	Mounting distance	U	Shaft angle
G	Dedendum angle	N	Outer cone distance	V	Equivalent pitch radius
		O	Outside diameter		

Figura 1.4. Engranaje cónico nomenclatura – plano axial

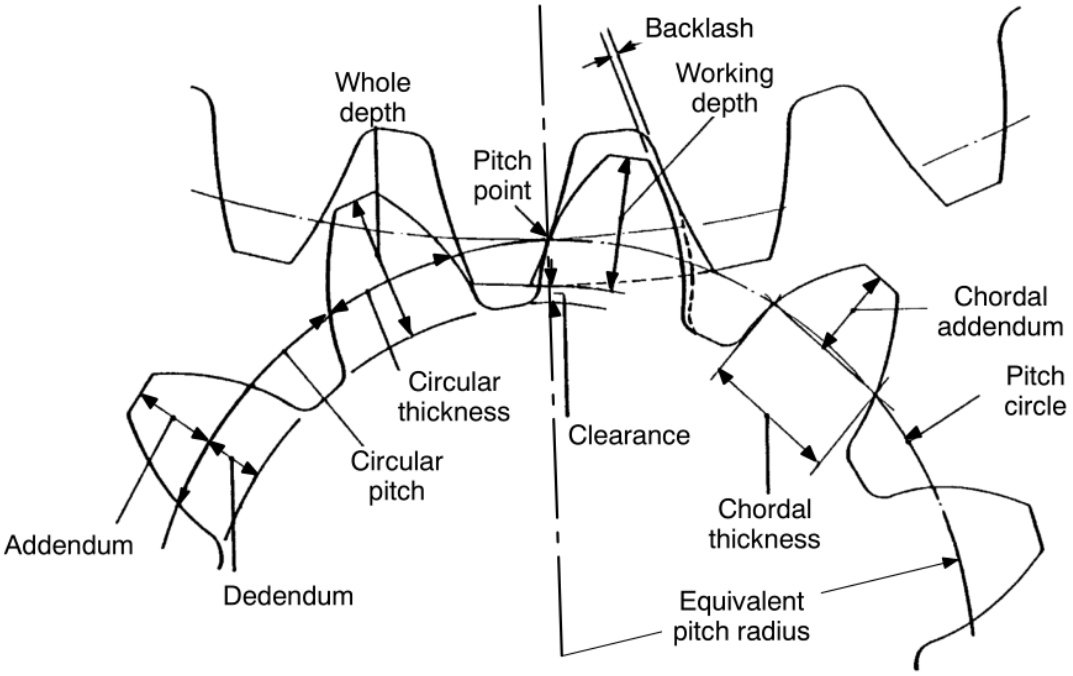


Figura 1.5. Engranaje cónico nomenclatura – sección media tramo A-A

Tabla 1.1. Símbolos y términos de engranajes cónicos de dientes rectos o helicoidales.

Símbolo	Término	Unidad
Am	Mean cone distance	in (mm)
Ao	Outer cone distance	in (mm)
aG	Gear mean addendum	in (mm)
aoG	Gear outer addendum	in (mm)
aoP	Pinion outer addendum	in (mm)
aP	Pinion mean addendum	in (mm)
bG	Gear mean dedendum	in (mm)
boG	Gear outer dedendum	in (mm)
boP	Pinion outer dedendum	in (mm)
bP	Pinion mean dedendum	in (mm)
c	Clearance	in (mm)
C_1	Mean addendum factor	- - -
D	Outer gear pitch diameter	in (mm)
d	Outer pinion pitch diameter	in (mm)
Do	Gear outside diameter	in (mm)
do	Pinion outside diameter	in (mm)
F	Face width	in (mm)
h	Mean working depth	in (mm)

hk	Outer working depth	in (mm)
ht	Outer whole depth	in (mm)
k1	Depth factor	- - -
K2	Clearance factor	- - -
m_{et}	Outer transverse module	in (mm)
mG	Gear ratio	in (mm)
m_{90}	Equivalent 90° ratio	in (mm)
N	Number of gear teeth	- - -
n	Number of pinion teeth	- - -
Pd	Outer transverse diametral pitch	in (mm)
Pdm	Mean diametral pitch	(in^{-1})
pm	Mean circular pitch	(in^{-1})
Q	Intermediate variable	in (mm)
rc	Cutter radius	in (mm)
Xo	Gear pitch cone apex to crown	in (mm)
xo	Pinion pitch cone apex to crown	in (mm)
αG	Gear addendum angle	deg (rad)
αP	Pinion addendum angle	deg (rad)
Γ	Gear pitch angle	deg (rad)
Γo	Gear face angle	deg (rad)
ΓR	Gear root angle	deg (rad)
γ	Pinion pitch angle	deg (rad)
γo	Pinion face angle	deg (rad)
γR	Pinion root angle	deg (rad)
δ_G	Gear dedendum angle	deg (rad)
δ_P	Pinion dedendum angle	deg (rad)
λ	First auxiliary angle	deg (rad)
Σ	Shaft angle	deg (rad)
$\Sigma\delta$	Sum of dedendum angles	deg (rad)
ψ	Mean spiral angle at pitch surface	deg (rad)

Tabla 1.2. Ecuaciones de los engranajes cónicos de dientes rectos o helicoidales.

No	Nombre del Símbolo	Fómula Piñón	Fómula común	Fómula Rueda
1	Pitch diameter	$d = n \cdot m_{et}$		$D = N \cdot m_{et}$

2	Pitch angle	$\gamma = \left[\frac{sin\Sigma}{\frac{N}{n} + cos\Sigma}\right]$		$\Gamma = \Sigma - \gamma$
3	Outer cone distance		$A_o = \frac{0.5 \cdot D}{sin\Gamma}$	
4	Mean cone distance		$A_m = A_o - 0.5 \cdot F$	
5	Mean working depth		$h = k_1 \cdot m_{et} \cdot \frac{A_m}{A_o} \cdot cos\psi$	
6	Clearance		$c = k_2 \cdot h$	
7	Mean whole depth		$h_m = h + c$	
8	Equivalent 90° ratio		$m_{90} = \sqrt{\frac{N}{n} \cdot \frac{cos\gamma}{cos\Gamma}}$	
9	Mean circular pitch		$pm = \pi \cdot m_{et} \cdot \frac{A_m}{A_o}$	
10	Mean addendum	$a_P = h - a_G$		$a_G = c_1 \cdot h$
11	Mean dedendum	$b_P = h_m - a_P$		$b_G = h_m - a_G$

12	Face angle	$\gamma_o = \gamma + \delta_G$		$\Gamma_o = \Gamma + \delta_P$
13	Root angle	$\gamma_R = \gamma + \delta_P$		$\Gamma_R = \Gamma + \delta_G$
14	Outer addendum	$a_{oP} = a_P + 0.5 \cdot F \cdot tan\delta_G$		$a_{oG} = a_G + 0.5 \cdot F \cdot tan\delta_P$
15	Outer dedendum	$b_{oP} = b_P + 0.5 \cdot F \cdot tan\delta_P$		$b_{oG} = b_G + 0.5 \cdot F \cdot tan\delta_G$
16	Outer working depth		$h_k = a_{oP} + a_{oG}$	
17	Outer whole depth		$h_t = a_{oP} + b_{oP}$	
18	Outside diameter	$d_o = d + 2 \cdot a_{oP} \cdot cos\gamma$		$D_o = D + 2 \cdot a_{oG} \cdot cos\Gamma$
19	Pitch cone apex to crown	$x_o = A_o \cdot cos\gamma - a_{oP} \cdot sin\gamma$		$X_o = A_o \cdot cos\Gamma \cdot a_{oG} \cdot sin\Gamma$
20	Mean diametral pitch		$P_{dm} = P_d \cdot \left(\frac{A_o}{A_m}\right)$	
21	Mean pitch diameter	$d_m = \frac{n}{P_{dm}}$		$D_m = \frac{N}{P_{dm}}$

1.6. Creando circunferencias de construcción

Para crear un engranaje se necesitan 4 circunferencias de construcción:

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

$$d = rbc \cdot 2 \tag{1.6.1}$$

Donde:

d.- diámetro de la circunferencia de referencia.

rbc.-radio de la distancia del cono trasero.

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 de la otra rueda; se crea posicionando un punto como centro, en este caso el punto (0; 0; 0) y su diámetro se calcula por la siguiente expresión:

$$da = (d + 2 \cdot Ao) \tag{1.6.2}$$

Donde:

da.- diámetro de circunferencia de cresta

d.- diámetro de la circunferencia de referencia.

Ao.-addendum externo

Circunferencia básica: sirve como base al trazado del perfil del diente y en algunos casos es tangente a la circunferencia de cresta de la otra rueda; se crea posicionando un punto como centro, en este caso el punto (0; 0; 0) y su diámetro se calcula por la siguiente expresión:

$$db = d \cdot cos(angle) \tag{1.6.3}$$

Donde:

db.- diámetro de circunferencia básica

d.- diámetro de la circunferencia de referencia.

angle.- ángulo de presión.

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:

$$df = d - 2 \cdot outer\_dedendum \tag{1.6.4}$$

Donde:

df.- diámetro de circunferencia de fondo

d.- diámetro de la circunferencia de referencia.

1.7. Creando perfil de diente para engranajes cónicos

Para crear un diente de un engranaje cónico hay que tener en cuenta varios elementos:

1.7.1. Perfil del diente

Para crear un perfil con una curva involuta lo primero que se necesita es obtener los puntos por los que pasa dicha curva, estos puntos se obtendrán con las siguientes ecuaciones paramétricas (SHIGLE, 1979):

$$x = \frac{db}{2} \cdot [cos\alpha + (\alpha \cdot sin\alpha)] \tag{1.7.1}$$

$$y = \frac{db}{2} \cdot [sin\alpha + (\alpha \cdot cos\alpha)] \tag{1.7.2}$$

La envolvente de circulo o Involuta es la curva descrita por 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 Figura 1.6)

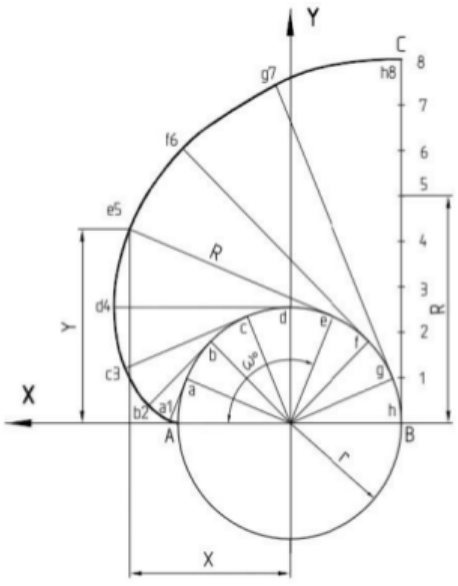


Figura 1.6. Trayectoria de la envolvente de círculo o Involuta.

Donde db es el diámetro de la circunferencia base y α es el ángulo del punto que genera la tangente en cada caso. El ángulo se va incrementando hasta llegar al valor deseado, cuando α >90 grados no es necesario seguir incrementando el ángulo. Luego de tener creados todos los puntos, se unen mediante segmentos para formar la curva Involuta y dicha curva se une a la circunferencia básica en caso de tener mayor diámetro que la circunferencia de fondo, de no ser así, la curva Involuta se une a la circunferencia de fondo como anteriormente se mencionó.

1.8. Construcción de Tredgold

Para estudiar en el plano la geometría del perfil de involuta en los engranajes cónicos, esta geometría es proyectada sobre la superficie esférica, se realiza la llamada construcción de Tredgold que consiste en proyectar sobre un cono complementario exterior, tangente a la esfera en la misma circunferencia de corte con el cono primitivo, el propio trazado de la circunferencia. Posteriormente, esa proyección sobre el cono exterior se desarrolla y representa de forma aproximada toda la geometría, se calcula el número de dientes equivalente. La cantidad de dientes en los engranaje cónicos de dientes rectos e helicoidales es imaginario; formulados en la ecuación:

$$N' = \frac{2\pi r}{p} \tag{1.8.1}$$

donde N' es el número virtual de dientes y p es el paso circular medido el pie del diente. Los engranajes cónicos estándar con dientes rectos se cortan usando una presión de 20°, addendum y dedendum desiguales, y dientes de profundidad completa. Esto aumenta el contacto relación, evita socavado, y aumenta la fuerza del piñón (SHIGLE, 1979).

1.9. Sistema de corte para engranajes cónicos

Los sistemas de cortes para engranajes cónicos es un aspecto importante para su modelado debido a que estos varían sus cálculos para determinadas máquinas y métodos con que se realicen, para los engranajes cónicos se realizan fundamentalmente con los métodos (STANDARD, 2005):

Face Milling

Los dientes de engranaje se pueden cortar con una fresa de forma que se adapte al espacio de los dientes. Con este método, teóricamente es necesario usar un cortador diferente para cada engranaje, porque un engranaje que tiene 25 dientes, por ejemplo, tendrá un espacio para dientes de forma diferente de uno que tiene, 24 dientes. En realidad, el cambio en el espacio no es demasiado grande, pero en ocasiones se ha encontrado que se pueden usar ocho cortadores para cortar con una precisión razonable cualquier marcha en el rango de 12 dientes. Por supuesto, se requiere un juego de cortadores por separado para cada corte (ibíd.).

Face Hobbing

Se denomina como el proceso de tallado, la encimera es simplemente una herramienta de corte que tiene la forma de un gusano. Los dientes tienen lados rectos, como en una cremallera básica, pero el eje de la placa

debe girar a través del ángulo de avance para cortar los dientes del engranaje cilíndrico. Por esta razón, los dientes generados por una placa tienen una forma ligeramente diferente de los generados por un cortador de *rack*. Tanto la placa como el blanco deben rotarse con la relación de velocidad angular adecuada. Luego se alimenta lentamente a través de la cara del blanco hasta que todos los dientes hayan sido cortados (STANDARD, 2005).

1.10. Módulos existentes en sistemas para el diseño asistido por computadora.

La producción de sistemas de diseño asistido por computadora a nivel mundial está muy diversificada. Independientemente de las tecnologías utilizadas, existen software especializados en la minería y al mismo tiempo se pueden encontrar soluciones orientadas a la arquitectura, la mecánica o incluso diseñados para la medicina o el audiovisual. A la hora de realizar una descripción de las tendencias de desarrollo de software de diseño asistido por computadora hasta la actualidad, hay que mencionar la controversia existente entre el software privativo y en sistemas de código abierto. El software privativo al ser desarrollados por empresas muy bien respaldadas económicamente, siempre se ha caracterizado por ser una solución muy potente o profesional, pero al mismo tiempo traen como principal inconveniente el problema de las licencias y la dificultad económica que esto representa. Por otro lado, existen otras alternativas que demuestran que se pueden realizar proyectos muy profesionales bajo la filosofía de sistemas de código abierto. En el proceso de investigación que se lleva a cabo a nivel de proyecto se logró identificar como una línea estratégica importante durante el desarrollo, la reutilización de todo el código disponible con funcionalidades probadas en otros sistemas de código abierto; de esta manera será posible reducir los tiempos hasta la obtención de las metas trazadas. Por ejemplo, se puede reutilizar código disponible en varios software como FreeCAD, LibreCAD y Salome Meca el cual es un sistema Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés) .

1.10.1. Sistemas para el diseño asistido por computadoras con licencia privada

Solid Edge es un portafolio de herramientas de software fácil de usar que abordan todos los aspectos del proceso de desarrollo de productos - diseño en 3D, simulación, fabricación, gestión del diseño y mucho más gracias a un creciente ecosistema de aplicaciones. Solid Edge combina la velocidad y simplicidad del modelado directo con la flexibilidad y el control de diseño paramétrico hecho posible con synchronous technology (Solid-Edge, 2018). (ver Figura 1.7)
Este programa comercial solo se puede usar en Windows y MacOS. En su versión ST4, a diferencia de Autodesk Inventor, solo posee dos módulos independientes en los que se pueden modelar y mostrar los cálculos de las propiedades. Estos módulos brindan pocos cálculos, provocando la posibilidad que el usuario no encuentre toda la información necesaria.

Solid Edge cuenta con un acelerador de diseño para el modelado de engranajes cónicos, el cual es cómodo e intuitivo para los diseñadores, pero este software al mostrar el resultado de modelado los engranajes cónicos no engranan, se superponen el piñón con la rueda y el ajuste se realiza manual, lo que genera un procedimiento engorroso y poco eficiente, además, no cuenta con el proceso de selección de tipo de engranaje y sus respectivas restricciones (PLM, 2018).

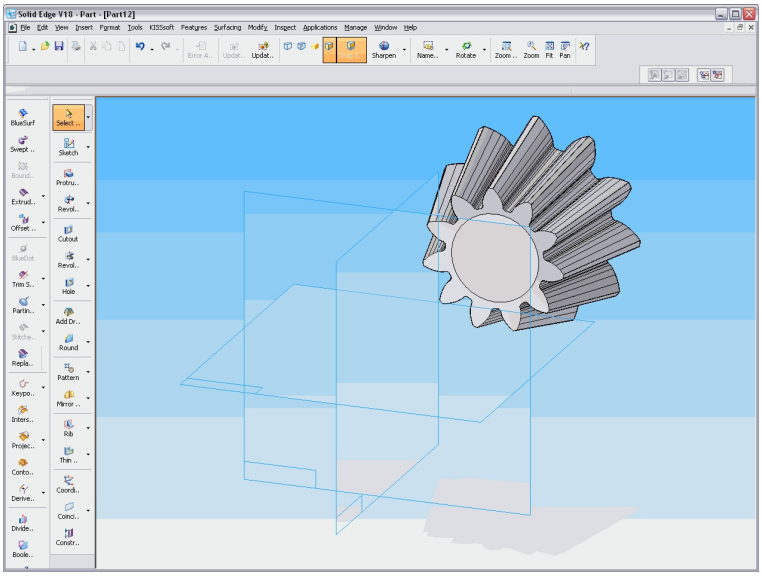


Figura 1.7. Engranaje cónico realizados en la herramienta Solid Edge.

Autodesk Inventor permite a los fabricantes ir más allá de un simple modelo 3D, permitiéndoles a los ingenieros y/o diseñadores crear prototipos digitales, gracias a un conjunto de herramientas para diseño mecánico 3D, simulación, análisis, visualización, y documentación. Con Autodesk Inventor los ingenieros pueden integrar los planos 2D de AutoCAD y los modelos 3D en un único modelo digital, creando representaciones digitales del producto final que les permite validar la forma, dimensiones, precisión del ensamble (tolerancias), así como el comportamiento del producto antes que sea construido (Inventor, 2018). (ver Figura 1.8)

Este programa comercial solo se puede usar en Windows y MacOS. Posee tres módulos independientes en los que se pueden modelar y mostrar los cálculos de las propiedades de engranajes cónicos de dientes rectos o helicoidales.

Autodesk Inventor posee un generador de componentes para el diseño de engranajes cónicos el cual es intuitivo para los diseñadores para el modelado y cálculo, pero no cuenta con el proceso de selección de tipo de engranaje y sus respectivas restricciones.

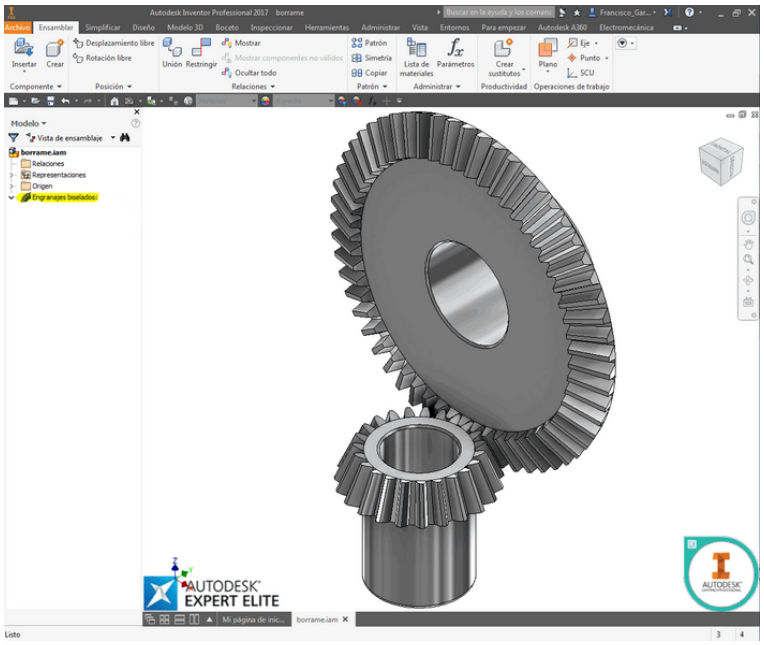


Figura 1.8. Engranaje cónico realizados en la herramienta Autodesk Inventor.

SolidWorks es un programa de diseño asistido por computadora para modelado mecánico desarrollado en la actualidad por SolidWorks Corp. para el sistema operativo Microsoft Windows. Es un modelador de sólidos paramétrico. Fue introducido en el mercado en 1995 para competir con otros programas de diseño asistido por computadora como Solid Edge, CATIA, y Autodesk Mechanical Desktop (SOLIDWORKS, 2018). (ver figura 1.9)

El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otras informaciones necesarias para la producción. Este posee un grupo de productos con distintas funcionalidades,

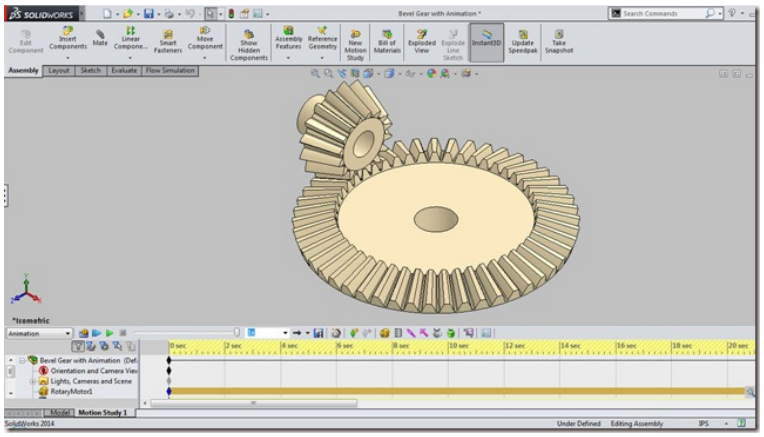


Figura 1.9. Par de engranajes cónicos realizados en la herramienta SolidWorks.

entre los que se encuentran:

- Soluciones de diseño asistido por computadora: 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.
- 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 creación fácil y rápida de contenido visual a diseñadores, ingenieros, expertos de marketing y otros creadores de contenidos.
- 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.
- 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.
- Diseño eléctrico: proporciona una serie de funcionalidades de diseño de sistemas eléctricos para satisfacer las necesidades de los profesionales.

SolidWorks cuenta con un acelerador de componentes para el modelado y el cálculo de engranajes cónicos, pero no cuenta con el proceso de selección de tipo de engranaje y sus respectivas restricciones.

CATIA: ofrece la posibilidad, no solo de modelar cualquier producto, sino de hacerlo en el contexto de su comportamiento en la vida real: diseño en la era de la experiencia. Los arquitectos de sistemas, los ingenieros, los diseñadores y todos sus colaboradores pueden definir el mundo que nos conecta, imaginarlo y darle forma.
(ver Figura 1.10) CATIA presenta las siguientes características:

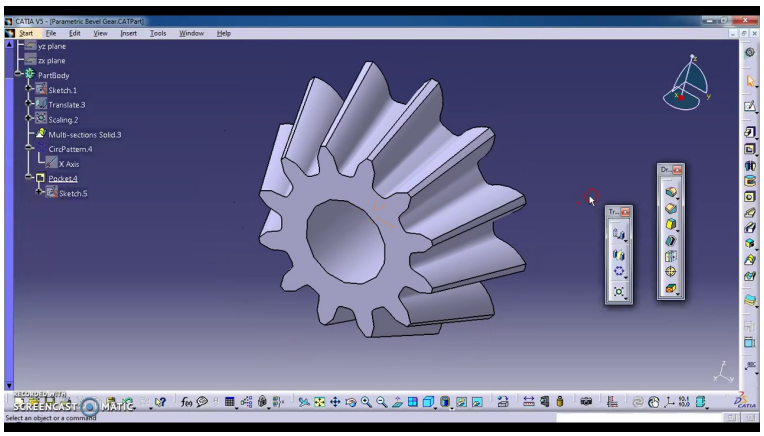


Figura 1.10. Engranaje realizado en la herramienta CATIA.

- Entorno de diseño social basado en una fuente única de autenticidad, al que se accede mediante potentes paneles en 3D que impulsan la inteligencia empresarial, el diseño simultáneo en tiempo real y la colaboración de todas las partes interesadas, incluidos los trabajadores móviles (CATIA, 2018).

- 3DEXPERIENCE ofrece una experiencia intuitiva con funcionalidades de modelado y simulación en 3D.
- Se trata de una plataforma inclusiva de desarrollo de productos, que resulta fácil de integrar con los procesos y herramientas existentes.
- Está disponible para Microsoft Windows, Solaris, IRIX y HP-UX.

CATIA no posee un acelerador de diseño para engranajes cónicos pero brinda la posibilidad de realizar de forma fácil con las primitivas correspondientes estos engranajes.

1.10.2. Sistemas para el diseño asistido por computadoras con licencia libre

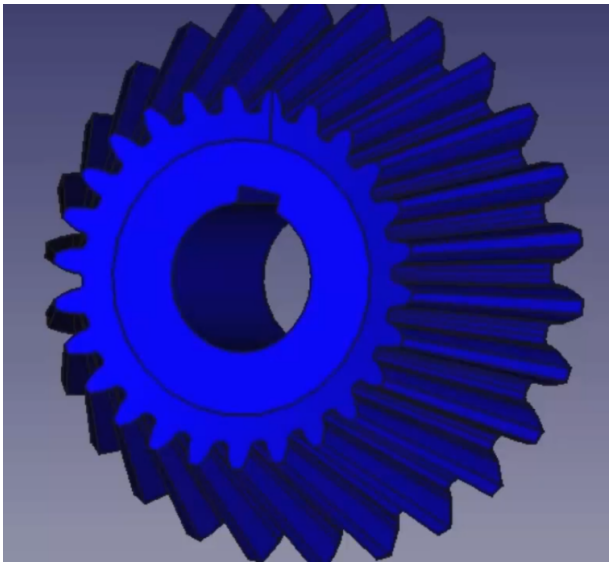


Figura 1.11. Par de engranajes helicoidales realizados en la herramienta FreeCAD.

FreeCAD es un modelador de diseño asistido por computadora en 3D. El desarrollo es completamente de código abierto (Licencia Pública General Reducida (LGPL, por sus siglas en inglés)). FreeCAD está dirigido directamente a la ingeniería mecánica y diseño de productos, pero también se emplea en una amplia gama de usos en la ingeniería, como la arquitectura (FreeCAD, 2017).
(ver Figura 1.11) FreeCAD cuenta con herramientas similares a CATIA, SolidWorks o Solid Edge, y por lo tanto también cae en la categoría de Diseño Asistido por computadora Orientado a la Mecánica (MCAD, por sus siglas en inglés) y CAE. 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 de diseño asistido por computadora en 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.
FreeCAD hace un uso intensivo de todas las grandes bibliotecas de código abierto que existen en el cam-

po de la Computación Científica. Entre ellas se encuentran Open CASCADE, un núcleo de gran alcance, Coin3D, una encarnación de Inventor abierto, Qt y Python. FreeCAD también es totalmente multiplataforma, y actualmente se ejecuta sin problemas en los sistemas Windows, Linux / Unix y Mac OSX, con el mismo aspecto y las funcionalidades exactas en todas las plataformas.

FreeCAD no cuenta con acelerador de diseño para engranajes cónicos, por lo que para su creación hay que realizar el engranaje por las primitivas correspondientes, lo cual es un proceso engorroso, puede existir errores de construcción y ocupa mucho tiempo.

1.10.3. Herramientas para el cálculo de engranajes cónicos de dientes rectos y helicoidales

MITCalc brinda un conjunto de cálculos de ingeniería, industriales, y técnicos para las rutinas del día a día de forma fiable y precisa (MITCalc, 2018).

Esta herramienta brinda, además de los cálculos, las verificaciones comunes de diseño de disímiles elementos mecánicos como: engranajes rectos, engranajes cónicos, engranaje helicoidal, engranaje planetario, correa de distribución, cojinetes, resortes, viga, pandeo, placas, conchas, y muchos otros. Permite la salida directa a disímiles sistemas de diseño comercial, como:

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

MITCalc posibilita a partir de varias normas, calcular los engranajes cónicos de forma eficiente y genera una tabla de resultados la cual puede ser exportar a Excel o a cualquiera de las aplicaciones anteriormente mencionadas.

eFunda es un sitio web que posee como objetivo: crear un destino en línea para la comunidad de ingeniería, donde pueda encontrarse de forma concisa y confiable la mayoría de los cálculos que necesiten (eFunda, 2018).
eFunda brinda los conceptos básicos, es decir, cubre el material 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. Dentro de estas se encuentran un conjunto de calculadoras independientes para distintas propiedades.

eFunda posibilita a partir de varias normas, calcular los engranajes cónicos de forma eficiente, ademas de realizar los cálculos de fuerza y resistencia.

1.11. Herramientas y lenguajes para el desarrollo

Visual Paradigm es una herramienta CASE 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 de los programas y la documentación (PRESSMAN, 2003).

Esta herramienta emplea el lenguaje de modelado UML. Se caracteriza por (ibíd.):

- Disponibilidad en múltiples plataformas (Windows, Linux).
- 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 Lenguaje Extensible de Enmarcado (XML, por sus siglas en inglés), 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 código en varios lenguajes, posee una licencia gratuita y emplea el lenguaje de modelado UML.

UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.

El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar, pero tiene como objetivo ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (JACOBSON, RUMBAUGH y BOOCH, 2000).

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos . El comportamiento dinámi-

co define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos (JACOBSON, RUMBAUGH y BOOCH, 2000).

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes (ibíd.).

Se decide emplear este lenguaje de modelado en el proceso de desarrollo de la propuesta de solución a causa de las características antes mencionadas y por su amplio uso en la Universidad de las Ciencias Informáticas.

Open CASCADE

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, y la visualización. La mayor parte de la funcionalidad OCCT se encuentra disponible en forma de bibliotecas de C ++. OCCT puede ser aplicado en el desarrollo de software cuyo objetivo sea el modelado en 3D, fabricación / medición (Fabricación Asistida por Computadora (CAM, por sus siglas en inglés)) o simulación numérica (CAE). Es una tecnología de software libre; se puede distribuir y/o modificar bajo los términos de la Licencia Pública General de GNU (LGPL) versión 2.1, con excepción adicional (Technology, 2018).

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) (ibíd.).

Open CASCADE Community Edition (OCE, en español Edición de la Comunidad de Open CASCADE) 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.

Lenguaje C++

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 (STROUSTRUP, 2013).

La librería estándar de C++ puede ser implementada en el propio C++ . 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, QT, OpenCV, entre otras.

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.

Framework Qt

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, 2018). Qt está disponible bajo varias licencias: licencia comercial y para software libre con varias versiones de la Licencia Pública General (GPL, por sus siglas en inglés) y la LGPL (ibíd.). Qt es mucho más que un conjunto de herramientas de Interfaces Gráficas de Usuario (Interface Gráfica de Usuario (GUI, por sus siglas en inglés), 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 Notación de Objetos JavaScript (JSON, por sus siglas en inglés), impresión, la generación de PDF, y mucho más (ibíd.). El Entorno Integrado de Desarrollo recomendado para el empleo de este framework es el Qt Creator.

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.

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 se aparte de casi todos los otros SCM es su modelo de ramificación. 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:

- **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 fíjelo.
- **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 este sistema de control de versiones durante el desarrollo de la propuesta de solución debido a las características antes mencionadas y por indicaciones del proyecto al que está integrado la presente investigación.

1.12. Acerca de la metodología para el desarrollo

Proceso Unificado Ágil (Agile Unified Process, AUP) es un enfoque de modelado híbrido creado por Scott Ambler cuando combinó el Rational Unified Process (RUP) con los métodos ágiles. Mediante esta combinación Ambler creó un marco sólido de procesos que se puede aplicar a todo tipo de proyectos de software, grandes o pequeños (Sánchez 2015).
Los principios que sustentan AUP son (Sanchez Rodriguez, 2015):

- La mayoría de la gente no va a leer documentación detallada. Sin embargo, se necesitará orientación y formación de vez en cuando.
- La descripción del proyecto debe ser en unas pocas páginas.
- Se ajusta a los valores y principios descritos en la Alianza Ágil.
- El proyecto debe centrarse en ofrecer valor esencial en lugar de características innecesarias.
- Los desarrolladores deben estar libres de utilizar las herramientas más adecuadas para la tarea en cuestión, en lugar de cumplir con un decreto.
- AUP se adapta fácilmente a través de herramientas de edición de HTML comunes.

En la Universidad de las Ciencias Informáticas (UCI) se realizó una variación a dicha metodología con el propósito de normalizar el proceso de desarrollo de software dentro de todos sus centros productivos (Sánchez 2015).

Tabla 1.3. Objetivos de las fases (Variación AUP-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 Construcción del 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 libera el producto. Durante esta se fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Para modelar el sistema en los proyectos productivos AUP-UCI define cuatro escenarios:

- Escenario No 1: Proyectos que modelen el negocio con Casos de Uso del Negocio solo pueden modelar el sistema con Casos de Uso del Sistema.
- Escenario No 2: Proyectos que modelen el negocio con Modelo Conceptual solo pueden modelar el sistema con Casos de Uso del Sistema.
- Escenario No 3: Proyectos que modelen el negocio con Diagrama de Procesos del Negocio solo pueden modelar el sistema con Diagrama de Requisitos por Proceso.
- Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de Usuario.

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 la necesidad de una metodología que responda con facilidad a los cambios continuos, por estar el cliente siempre acompañando el desarrollo y por estar bien definido el negocio.

1.13. Conclusiones del capítulo.

En el presente capítulo se presentó el diseño teórico de la investigación en cuyo marco se analizaron las dificultades que enfrenta la ingeniería nacional en relación con el uso de los sistemas comerciales para el diseño asistido por computadora. Se abordaron temas relacionados con los aceleradores de diseño, se mostró la metodología de cálculo de engranajes cónicos de dientes rectos y helicoidales, se expusieron las decisiones, que en el contexto del proyecto se tomaron, sobre las metodologías para el desarrollo de software en el desarrollo del componente Módulo para acelerar el diseño de engranajes cónicos de dientes rectos y helicoidales. Se presentó la selección del patrón arquitectónico adecuado junto con su justificación y se analizaron las tendencias y tecnologías de desarrollo de software actuales ofreciendo características, ventajas y desventajas de las mismas.

CAPÍTULO 2

Descripción de la solución propuesta.

En el presente capítulo se presenta la propuesta de solución, que incluye la descripción de las funcionalidades, los requisitos funcionales y no funcionales, las historias de usuario, así como los aspectos de diseño arquitectónico y los patrones orientados a objeto empleados.

2.1. Mapa conceptual

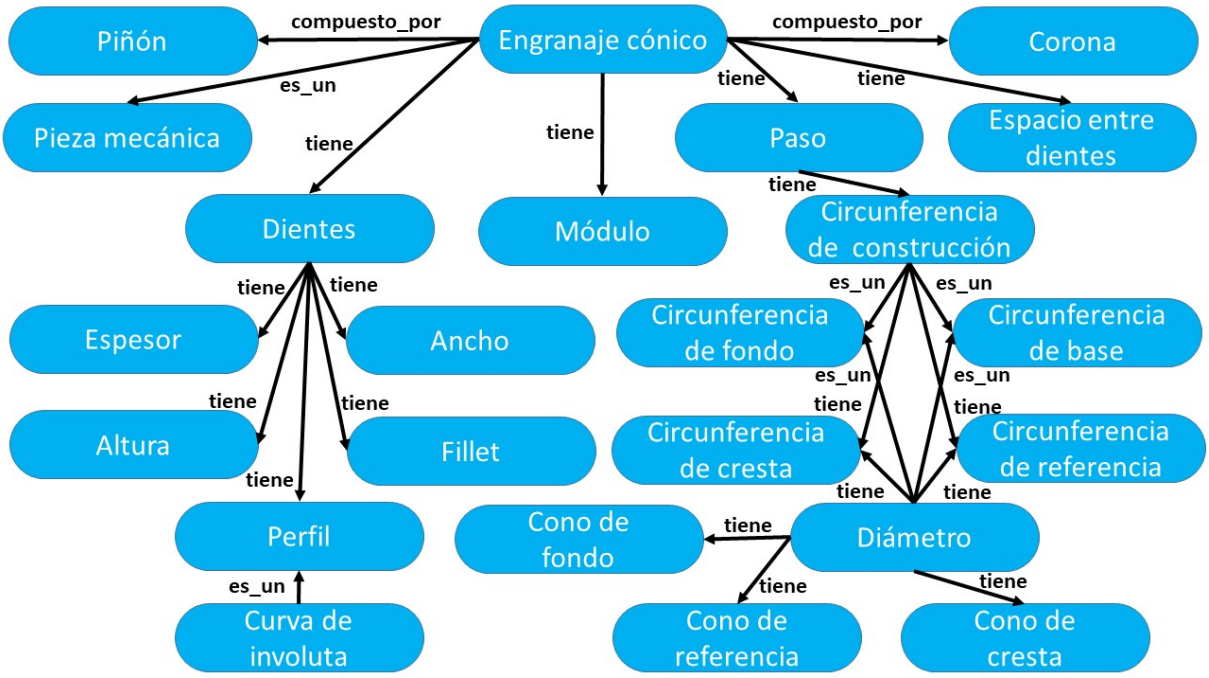


Figura 2.1. Mapa conceptual.

El mapa conceptual no se está utilizando desde el punto de vista ingenieril, sino para apoyar la descripción de la solución, con el objetivo de que los conceptos relacionados en el, sean entendidos por cualquier

persona (ver Figura 2.1).

En el mapa conceptual, muestra la relación que existe entre los parámetros asociados al cálculo de engranajes; el diámetro es un parámetro común en las circunferencias de referencia, básica, de cresta y de fondo debido a que es utilizado como base para el cálculo; estas cuatro circunferencias se denominan circunferencia de construcción que no son más que un componente del engranaje como el paso, el módulo, el espacio entre dientes y los dientes; estos últimos poseen espesor, ancho, altura, fileteo y un perfil que, en este caso, fue construido con la curva involuta, la cual se genera por el posicionamiento de puntos en el plano que se obtienen de dos ecuaciones paramétricas anteriormente descritas. Se incluye ademas los conos de referencia, fondo y cresta que influyen en el proceso de modelado.

Definición de los componentes que integran el mapa conceptual.

Engranaje cónico es el engranaje cónico resultante

Espacio entre dientes es la distancia en milímetro que existe entre los dientes del engranaje

Paso es la longitud de la circunferencia primitiva correspondiente a un diente y a un vano consecutivo.

Módulo es la característica de magnitud que se define como la relación entre las medidas del diámetro primitivo expresado en milímetros y el número de dientes.

Pieza mecánica es el mecanismo o conjunto de sólidos resistentes que reciben una energía de entrada y, a través de un sistema de transmisión y transformación de movimientos, realizan un trabajo.

Circunferencia de construcción es la unión circunferencias para trazar el perfil de involuta.

Circunferencia de fondo es una de las circunferencia para la construcción del engranaje.

Circunferencia de base es una de las circunferencia para la construcción del engranaje.

Circunferencia de referencia es una de las circunferencia para la construcción del engranaje.

Circunferencia de cresta es una de las circunferencia para la construcción del engranaje.

Diámetro es la distancia del centro del engranaje hasta la cabeza del diente.

Diente es el que realiza el esfuerzo de empuje y transmiten la potencia desde los ejes motrices a los ejes conductivos.

Espesor es el ancho de diente.

Altura es la suma de la cabeza del diente (adendum) más la altura del pie del diente(dedendum).

Fillet es el descanso del diente.

Perfil es la unión de los puntos trazados por las circunferencias de referencia para formar el perfil del diente

Curva de involuta o envolvente de circulo es una curva plana de desarrollo, cuyas normas son tangentes de la circunferencia.

Cono de fondo es una de las guías para la construcción del engranaje.

Cono de referencia es una de las guías para la construcción del engranaje.

Cono de cresta es una de las guías para la construcción del engranaje.

2.2. Descripción de las funcionalidades

El componente elaborado permite el modelado de engranajes cónicos de dientes rectos y helicoidales con perfil de involuta; el cual cuenta con varias características que serán explicadas en los próximos párrafos.
El componente muestra una interfaz que posee una pestaña en la parte superior izquierda llamada Design. En Design se encuentran los datos y parámetros relativos al diseño de engranajes cónicos; el usuario puede seleccionar los valores de los datos, de listas desplegables en el caso del ángulo de presión (con valores entre 14,5°y 30°) el módulo (con un rango de datos entre 0,050 mm hasta 100 mm) la relación de transmisión (con un rango de datos entre 1.000 ul y 40.000 ul (donde ul son unidades lineales)) y la guía de diseño que es la opción que se encarga de habilitar los campos para rectificar ciertos parámetros según las normas establecidas. También posee campos para introducir los valores deseados como el número de dientes del piñón y de la rueda secundaria, el ángulo de la hélice, el ángulo de inclinación, el ancho de la cara, o las correcciones de unidades de cada rueda y las totales.
La interfaz posee varios botones de acción, el de cancelar con la designación Cancel permite salir del componente de construcción de engranajes al instante, el de calcular designado como Calculate activa los cálculos correspondientes, si el engranaje no está correctamente calculado, en una pequeña área blanca que esta insertada en la interfaz se lanza un mensaje de error en letras rojas que deberá acercar al usuario al error de construcción; cuando el cálculo del engranaje está correcto, se muestra un mensaje en letras azules en la misma área blanca, y a su vez se activa el botón Aceptar, este cierra la interfaz de construcción de engranajes cónicos y muestra, en tres dimensiones, el diseño del engranaje cónico en un grid, el cual es la interfaz principal del proyecto.

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.

Los requisitos funcionales son las declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a las entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema debe hacer.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.

2.3.1. Requisitos funcionales

A continuación se describen los requisitos funcionales del componente.

RF 1 Insertar parámetros comunes para el piñón y la corona.

RF 2 Insertar parámetros del piñón.

RF 3 Insertar parámetros de la corona.

RF 4 Seleccionar tipo de engranaje cónico.

RF 5 Seleccionar piñón o piñón-corona para modelar.

RF 6 Mostrar resultado de los cálculos.

RF 7 Realizar modelado del piñón.

RF 8 Realizar modelado del par piñón-corona.

2.3.2. Requisitos no funcionales

A continuación, se exponen los requisitos no funcionales del componente:

Portabilidad:

RNF 1. Tener instalado cualquier distribución de GNU-Linux de 64 bits.

2.4. Historias de usuarios

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

Tabla 2.1. Insertar parámetros comunes para el piñón y la corona

Historia de usuario	
Número: 1	**Nombre:** Insertar parámetros comunes para el piñón y la corona
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 175 horas

Continuación de la página anterior

Tabla 2.1. Continuación de la página anterior

Descripción:

1.Objetivo Permitir la inserción de los parámetros de entrada comunes para los engranajes cónicos de tipo corona y de tipo piñón y a partir de ellos calcular sus otras propiedades.

2.Comportamiento válido y no válido

- Los valores deben de ser numéricos reales positivos y son campos obligatorios.
- El ángulo de la hélice debe corresponderse con el tipo de engranaje cónico seleccionado.

3.Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Calculate se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
- Si se selecciona el botón Cancel se cierra la ventana.
- Si los datos están incorrectos se muestra un mensaje de error dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Interfaz:

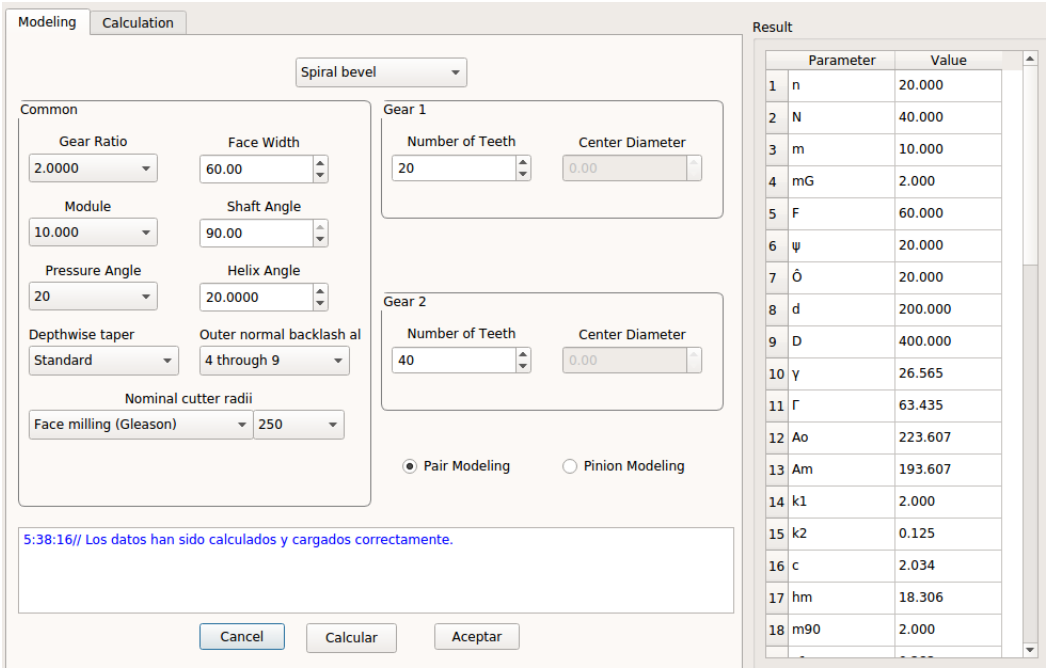


Tabla 2.2. Insertar parámetros del piñón

Historia de usuario	
Número: 2	**Nombre:** Insertar parámetros del piñón

Continuación de la página anterior

Tabla 2.2. Continuación de la página anterior

Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 90 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 110 horas

Descripción:

1.Objetivo Permitir la inserción de los parámetros de entrada tipo piñón y a partir de ellos calcular sus otras propiedades.

2.Comportamiento válido y no válido

- Los valores deben de ser numéricos reales positivos y son campos obligatorios.

3.Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Calculate se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
- Si se selecciona el botón Cancel se cierra la ventana.
- Si los datos están incorrectos se muestra un mensaje de error dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Interfaz:

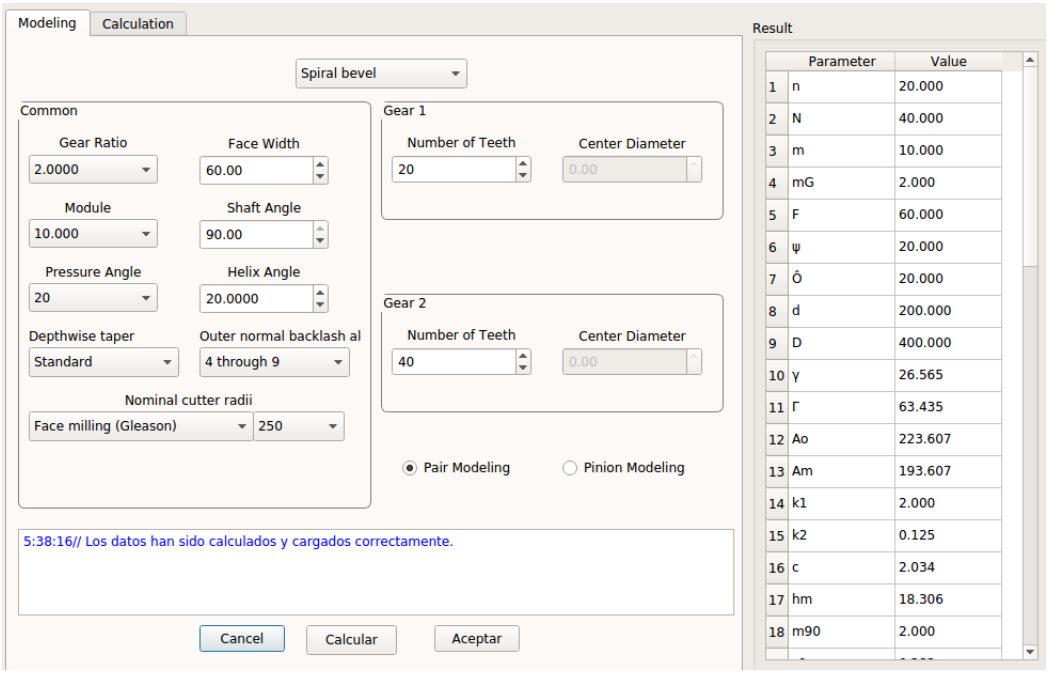


Tabla 2.3. Insertar parámetros de la corona.

Historia de usuario	
Número: 3	**Nombre:** Insertar parámetros de la corona.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 90 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 110 horas

Descripción:

1.Objetivo Permitir la inserción de los parámetros de entrada de tipo corona y a partir de ellos calcular sus otras propiedades.

2.Comportamiento válido y no válido

- Los valores deben de ser numéricos reales positivos y son campos obligatorios.

3.Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Calculate se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
- Si se selecciona el botón Cancel se cierra la ventana.
- Si los datos están incorrectos se muestra un mensaje de error dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Interfaz:

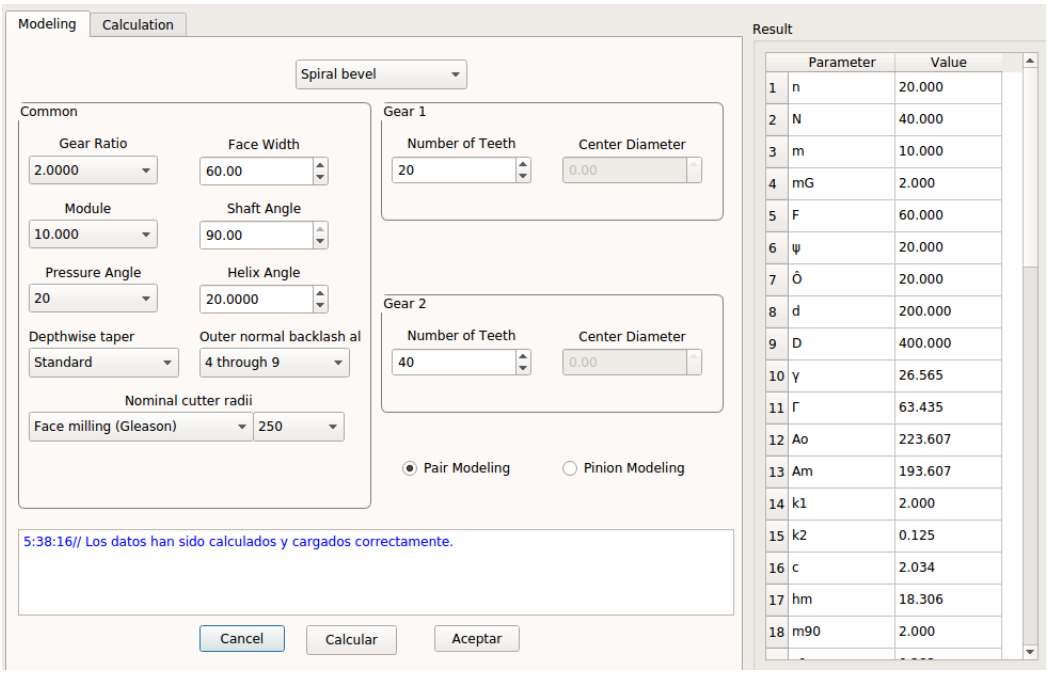


Tabla 2.4. Seleccionar tipo de engranaje cónico.

Historia de usuario	
Número: 4	**Nombre:** Seleccionar tipo de engranaje cónico.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 70 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 100 horas

Descripción:

1.Objetivo Permitir la selección de tipo de engranaje y a partir de ellos calcular con sus restricciones pertinentes.

2.Comportamiento válido y no válido

- Los campos son obligatorios.

3.Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Calculate se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
- Si se selecciona el botón Cancel se cierra la ventana.
- Si los datos están incorrectos se muestra un mensaje de error dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Interfaz:

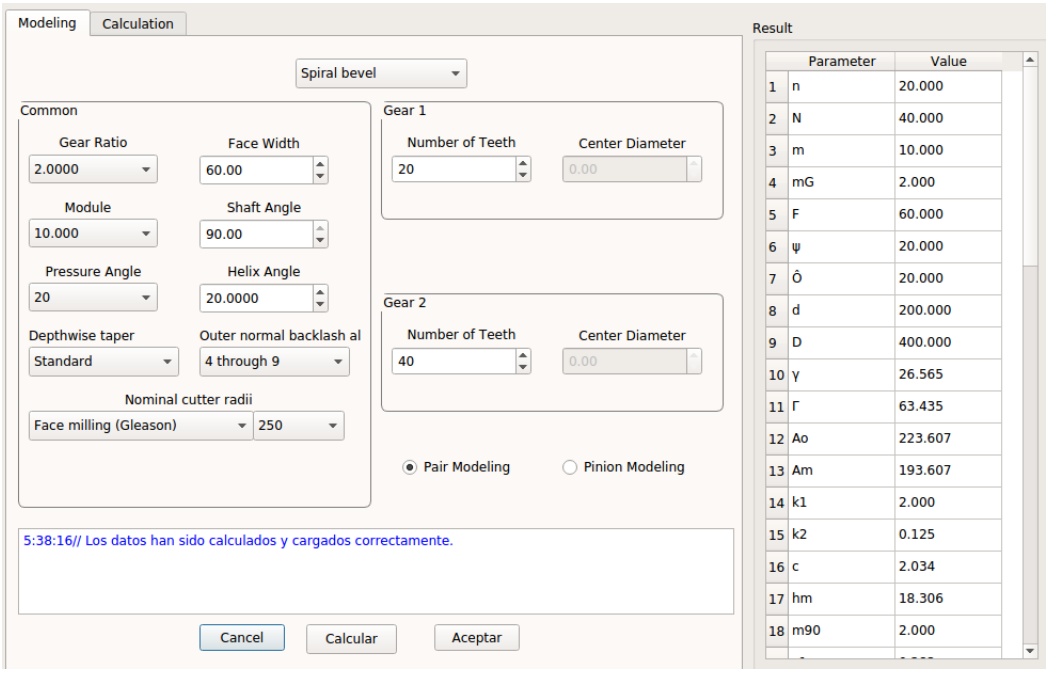


Tabla 2.5. Seleccionar piñón o piñón-corona para modelar.

Historia de usuario	
Número: 5	**Nombre:** Seleccionar piñón o piñón-corona para modelar.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 110 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 115 horas

Descripción:

1.Objetivo Permitir la selección de tipo de engranaje y a partir de ellos muestra su visualización.

2.Comportamiento válido y no válido

- Los campos son obligatorios.

3.Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción Calculate se calculan las restantes propiedades y se muestra la vista de los resultados de los cálculos.
- Si se selecciona el botón Cancel se cierra la ventana.
- Si los datos están incorrectos se muestra un mensaje de error dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.

Observaciones:

Interfaz:

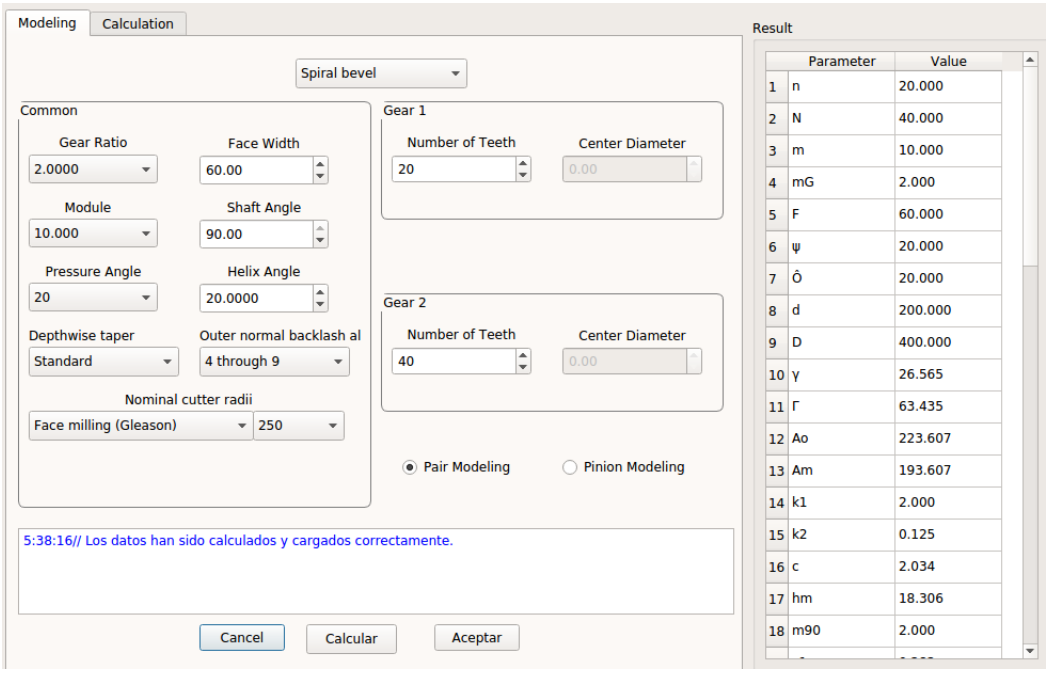


Tabla 2.6. Mostrar resultado de los cálculos.

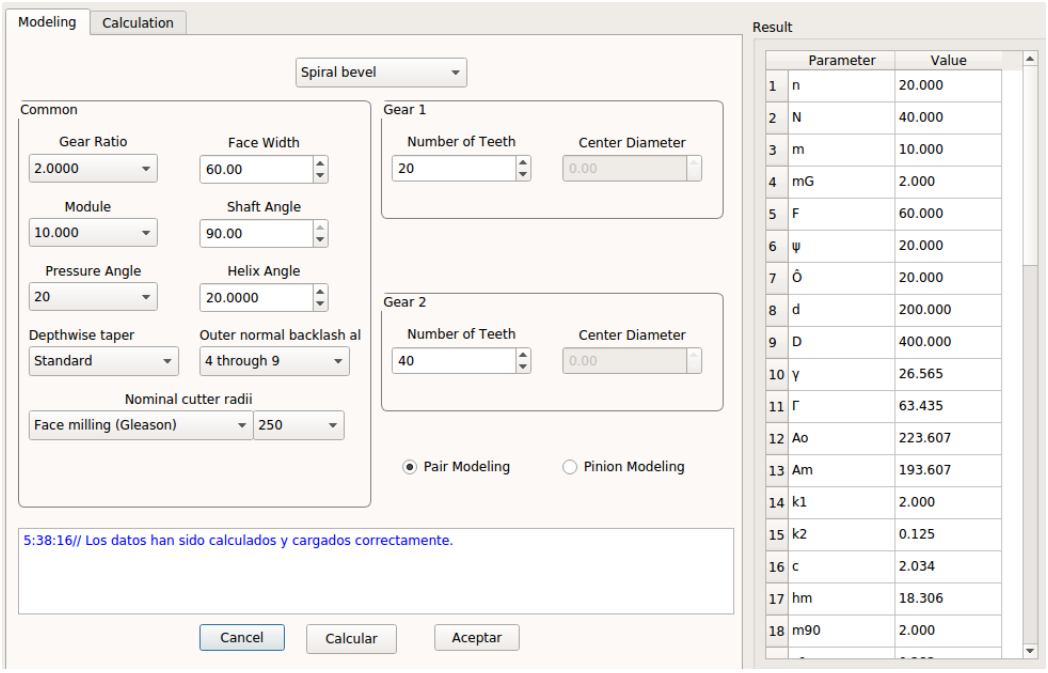
Historia de usuario	
Número: 6	**Nombre:** Mostrar resultado de los cálculos.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 85 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 75 horas
Descripción: **1.Objetivo** Muestra el resultado de los cálculos con la correcta entrada de parámetros. **2.Comportamiento válido y no válido** **3.Flujo de acción**	
Observaciones:	
Interfaz: 	

Tabla 2.7. Realizar modelado del piñón.

Historia de usuario	
Número: 7	**Nombre:** Realizar modelado del piñón.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 280 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 315 horas

Continuación de la página anterior

Tabla 2.7. Continuación de la página anterior

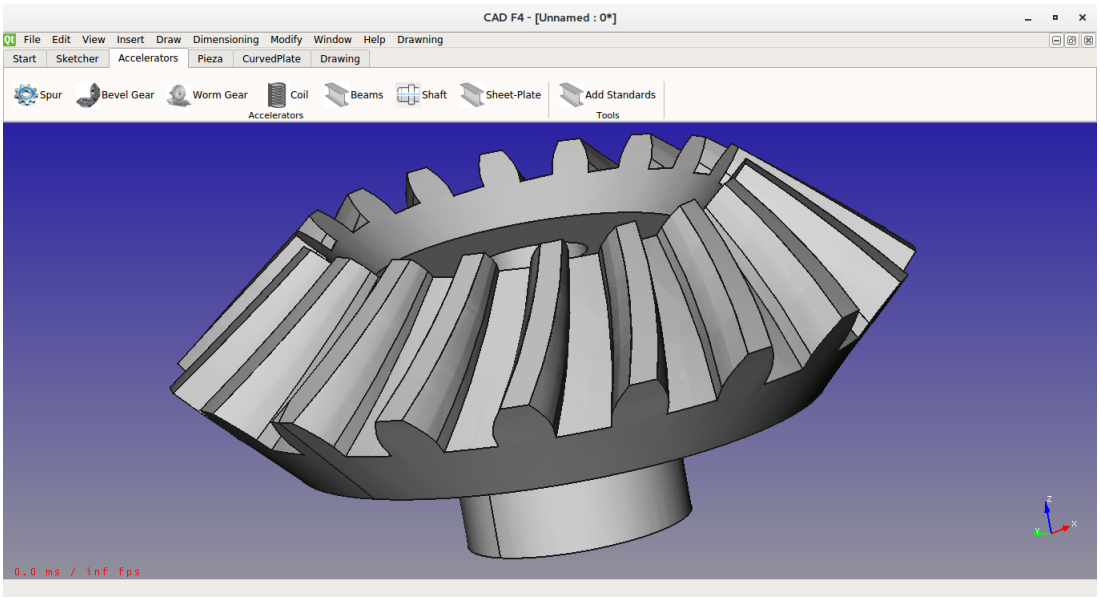
Descripción: **1.Objetivo** Muestra el resultado del diseño con el correcto cálculo dado por la entrada de parámetros. **2.Comportamiento válido y no válido** **3.Flujo de acción**
Observaciones:
Interfaz: 

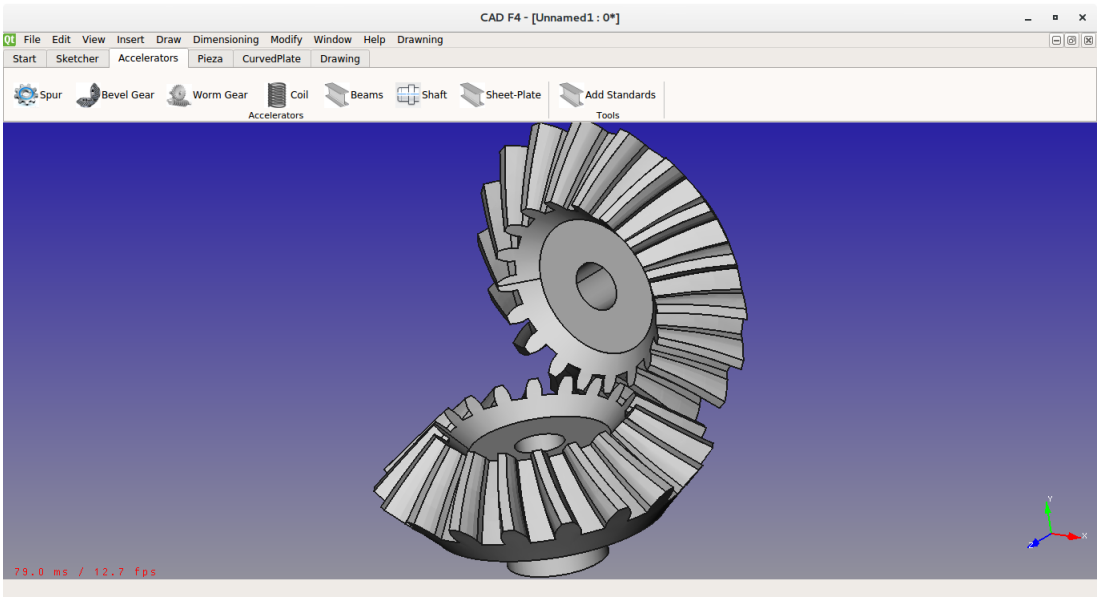
Tabla 2.8. Realizar modelado del par piñón-corona.

Historia de usuario	
Número: 8	**Nombre:** Realizar modelado del par piñón-corona.
Programador: Carlos Alberto Gómez García	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 280 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 315 horas
Descripción: **1.Objetivo** Muestra el resultado del diseño con el correcto cálculo dado por la entrada de parámetros. **2.Comportamiento válido y no válido** **3.Flujo de acción**	
Observaciones:	

Continuación de la página anterior

Tabla 2.8. Continuación de la página anterior

Interfaz:



2.5. Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida. En el diseño se modeló el sistema y se encontró su forma para que soporte todos los requerimientos o aspectos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado de la planificación, que proporciona una comprensión detallada de los requisitos o aspectos. Además, impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se le da forma al componente . El diseño crea una entrada apropiada y un punto de partida para actividades de implementación, y se es capaz de descomponer dichas actividades en partes más manejables, que pueden ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.

2.5.1. Estilo y patrón arquitectónico del software

``Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema. En caso de que una arquitectura existente se vaya a someter a reingeniería, la imposición de un estilo arquitectónico desembocará en cambios fundamentales en la estructura del software, incluida una reasignación de la funcionalidad de los componentes´´(Pressman, 2010).

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 cambiar de tamaño´´Pressman2010.

Este estilo posee como patrones arquitectónicos a las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

¨Un patrón arquitectónico define un enfoque específico para el manejo de alguna característica de comportamiento del sistema´´Pressman2010.

¨Los patrones se usan junto con un estilo arquitectónico para determinar la forma de la estructura general de un sistema ´´Pressman2010.

2.5.2. Arquitectura en Dos Capas

El modelo en dos capas organiza el sistema, la capa inferior proporciona un conjunto de servicios a la capa superior. Este modelo soporta el desarrollo incremental del sistema. También, soporta bien los cambios y es portable. Además, cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades a una capa, solamente se ve afectada la capas adyacente.

Se decide el empleo de este modelo en la confección del componente para el modelado de engranaje cónico de dientes rectos y helicoidales, debido a que esta arquitectura en dos capas permite la comunicación con el núcleo de la aplicación y con otros módulos de la misma.

2.5.3. Patrones de diseño

Los patrones de diseño brindan solución a problemas comunes que pueden ser encontrados durante el diseño, perfeccionando los componentes de un sistema de software y sus relaciones. El componente, para estructurar el diseño del sistema incluye un conjunto de patrones que formarán parte de la solución, como son:

Experto: mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidencia en las clases: “BevelGear”.

Creador: permite crear objetos de una clase determinada. Es utilizado en algunas de las clases controladoras para crear instancias de formularios y entidades. Este patrón esta presente en “DlgCreateBevel”y cuenta con la información necesaria para la creación de objetos tipo “BevelGear”

Controlador: se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. Este patrón se evidencia en las clases “qoccharnesswindow.cpp”, “qoccapplication.cpp”, “qoccinputoutput.cpp”.

Patrón GOF.

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. Este patrón se

evidencia con la función "*mustExecute*" la que permite que las propiedades de las piezas modeladas, puedan ser editadas.

2.5.4. Prototipos de interfaces de usuario

El diseño de la interfaz de usuario se puede definir como: el conjunto de pasos que seguirá el usuario, durante todo el tiempo que interactúe con el componente, detallando lo que verá en cada momento, y las acciones que realizará, así como las respuestas que el sistema dará.
A continuación se muestran los prototipos de interfaz del componente:
En la ventana *Modeling* Figura 2.2 se muestran los parámetros necesarios para el cálculo del diseño del

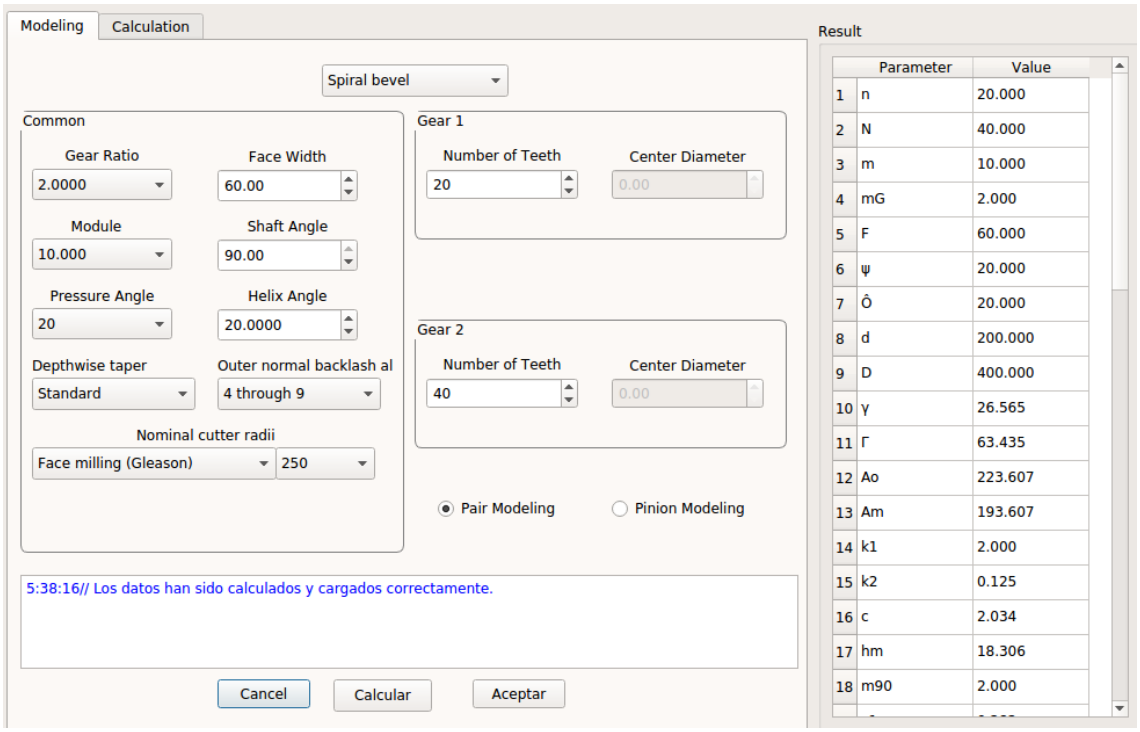


Figura 2.2. Prototipo de interfaces pestaña *Modeling*.

engranaje, así como la *DesignGuide* o guía de diseño la cual se encarga de determinar que parámetros están activos en cada caso y cuales no; una vez pulsado el botón *Calcular*, se lanza un mensaje en el área blanca en la parte inferior de la ventana, que de acuerdo al tipo de mensaje, está disponible el acceso a la ventana de *Preview* y a la visualización mediante el botón *Aceptar*.

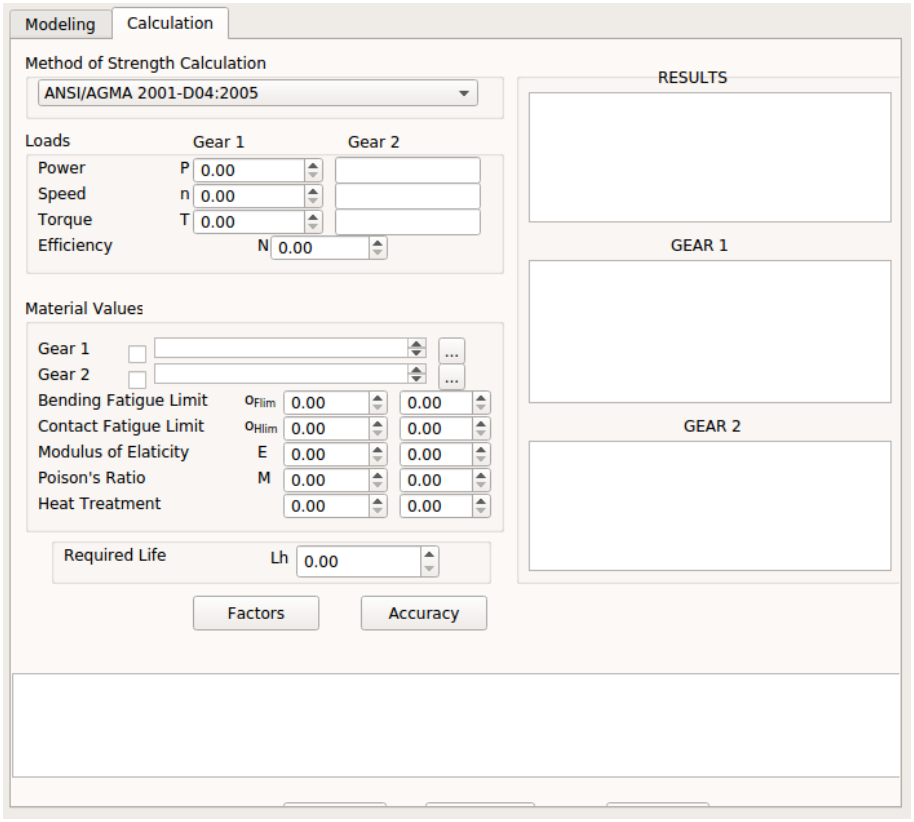


Figura 2.3. Prototipo de interfaces pestaña *Calculation* .

En la ventana *Calculation* Figura 2.3 se muestran los parámetros necesarios para el cálculo de propiedades físicas del engranaje, así como la norma por la que se realizarán dichos cálculos; una vez pulsado el botón *Calcular*, se mostrarán los valores correspondientes a cada rueda en el área blanca de la derecha.

2.5.5. Diagrama de clase del diseño.

Un diagrama o modelo de clases en UML es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos (herencia, agregación, asociación, entre otras) (ver Figura 2.4).

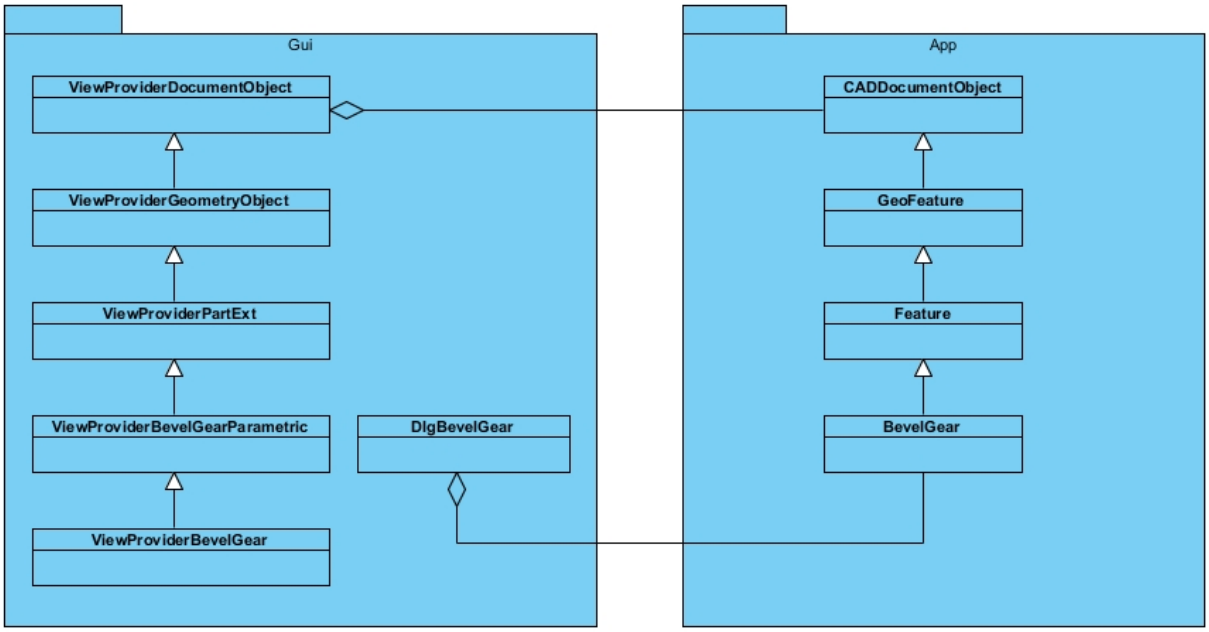


Figura 2.4. Diagrama de clase del diseño.

2.6. Conclusiones del capítulo.

A partir del estudio del marco teórico se definió la propuesta de solución y haciendo uso de las herramientas y metodología definidos en el capítulo anterior fue posible realizar un levantamiento de requisitos descritos en historias de usuario definiendo las funcionalidades a implementar para dar cumplimiento a los objetivos planteados al inicio de la investigación. Se definió el modelo de diseño de clases donde se aplicaron los patrones de diseño que permitirán el correcto desarrollo de las funcionalidades. Se definió el modelo arquitectónico y las clases que conformarán las diferentes capas y las relaciones entre ellas. Los artefactos obtenidos en este capítulo sientan las bases para la implementación de la solución propuesta. Propuesta de Solución

CAPÍTULO 3

Implementación y pruebas

En el presente capítulo se procede a realizar la implementación y las pruebas de software para garantizar la obtención de un producto de calidad. Se crea el diagrama de componentes del sistema y se definen los estándares de codificación empleados para el desarrollo de la solución. Son definidos los niveles y técnicas de pruebas que le serán realizadas a la aplicación. Se diseñan también casos de pruebas que servirán para llevar a cabo estas comprobaciones de calidad.

3.1. Implementación

Dentro del ciclo de vida de un software, se encuentra la fase de implementación; en esta fase se involucran personas, herramientas y recursos con el fin de completar todo el trabajo realizado previamente durante el ciclo de vida. En esta fase se establece el estándar de codificación que se utilizara según las normas establecidas.

El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. El objetivo del diseño es producir un modelo o representación de una entidad que se va a construir posteriormente (Pressman, 2010).

3.1.1. Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe quedar como si un único programador hubiera escrito todo el código de una sola vez. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Para un programador comprender bien un sistema de software, influye directamente la legibilidad del código fuente. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento (ver figura 3.1).

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()
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 3.1. Estándar de codificación.

3.1.2. Diagrama de componente.

Un diagrama de componentes 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. A continuación se muestra el diagrama de componentes de la solución(JACOBSON, RUMBAUGH y BOOCH, 2000) (ver Figura 3.2).

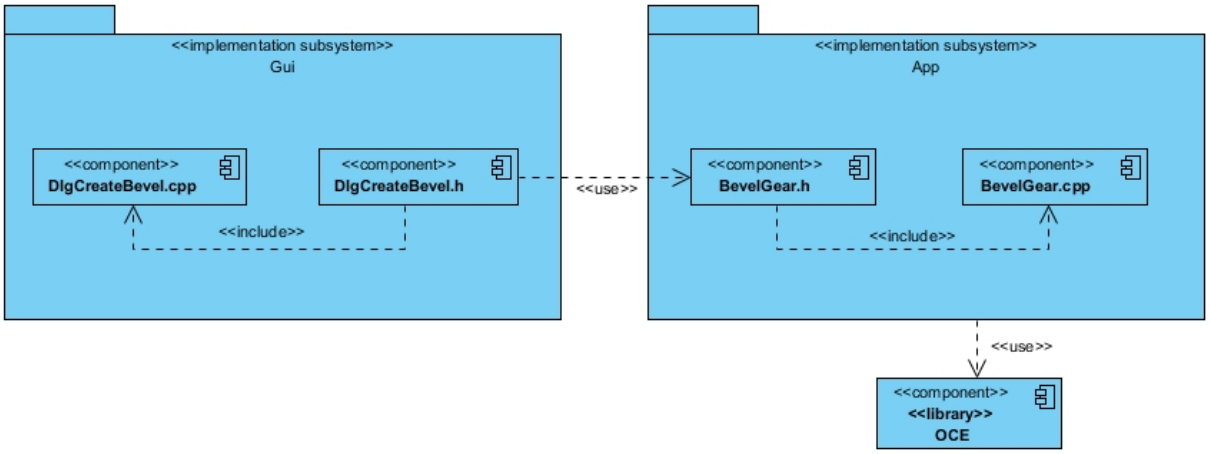


Figura 3.2. Diagrama de componente.

3.1.3. Descripción de la solución

Para el modelado geométrico de engranajes cónicos de dientes rectos y helicoidales se utilizaron una serie de bibliotecas las cuales permitieron la posterior visualización en la aplicación así como la manipulación de los datos, las cuales se presentan a continuación:

TopoDS_Shape.hxx: Esta librería es la principal utilizada para los formatos de retornos los cuales son mostrados por el visor propio de la aplicación ya que las entidades o secciones del árbol están constituidas por objetos de este tipo.

Standard_Real.hxx: Formato numérico incluido en el paquete de librerías de OpenCascade el cual es equivalente al dominio de números reales.

Standard_Integer.hxx: Formato numérico equivalente al dominio de enteros.

BRepPrimAPI_MakeCone.hxx: Posee las características para la modelación de secciones cónicas.

Geom_ConicalSurface.hxx: Describe la superficie cónica.

BRepAdaptor_Curve.hxx: Permite usar una topología Edge como una curva 3D.

BRep_Tool.hxx: Proporciona métodos de clase para acceder a la geometría de las formas.

BRepOffsetAPI_MakePipeShell.hxx: Esta clase proporciona un marco para construir un sólido a lo largo de un Wire.

gp_Dir.hxx: Conforma el par de coordenadas x, y, z necesarias para el posicionamiento de un punto en el espacio.

gp_Pnt.hxx: Crea un punto en el espacio utilizando coordenadas cartesianas XYZ.

gp_Circ.hxx: Describe un círculo en un espacio tridimensional.

gp_Vec.hxx: Define un vector no persistente en un espacio tridimensional.

gp_Trsf.hxx Define una transformación en el espacio 3D. Se implementan las siguientes transformaciones: Traducción, Rotación, Escala. Simetría con respecto a un punto, una línea o un plano.

AIS_InteractiveContext.hxx: Contiene el conjunto completo de funcionalidades para mostrar las secciones en el visor principal de la aplicación.

BRepAlgoAPI_Cut.hxx: Contiene los métodos necesarios para realizar la operación de diferencia booleana entre dos entidades geométricas en el espacio.

BRepAlgoAPI_Fuse.hxx: Posee los métodos de suma booleana entre dos entidades en el espacio.

BRepAlgoAPI_Common.hxx: Brinda las funciones de espacios comunes entre dos entidades geométricas y retorna las áreas comunes entre las dos.

También se crearon varios métodos, entre los cuales se encuentran:

makePinnon(...): Utilizando los valores que recibe por parámetros se construye la rueda primaria del engranaje.

makeSecondaryWheel(...): Utilizando los valores que recibe por parámetros se construye la rueda secundaria del engranaje.

loadBevelGear(...): Utilizando los valores que recibe por parámetros manda a visualizar el engranaje.

Un par de engranajes cónicos ha de calcularse conjuntamente, formando pareja ya que el ángulo de los conos primitivos han de complementarse, es decir sumar 90°; por esta razón no pueden intercambiarse con otros engranajes de distinto número de dientes que los calculados.

Una vez descritas las funciones principales por las cuales fueron creadas la solución, comenzaremos paso a paso como se construyó un engranaje cónico de dientes rectos o helicoidales. Los pasos que continuación se describen, se encuentran reflejados en las figuras 3.3 y 3.4 separadas por secciones respectivamente.

Construcción de engranajes cónico con dientes rectos o helicoidales.

- Paso 1: se crearon tres conos (base, referencia y cresta o corona) que se tomaron como guías para la conformación del módulo, partiendo de las fórmulas básicas para su geometría propuestas en el primer capítulo figura 3.3 sección 1.
- Paso 2: se realizó la hélice sobre una superficie cónica conformada por las ecuaciones paramétricas:

$$u = coneDir * cantidad\_de\_vueltas * 2 * \pi \tag{3.1.1}$$

$$v = ((pitch\_diameter\_pinion/2)/tan(pitch\_angle\_pinion))/cos(pitch\_angle\_pinion) \tag{3.1.2}$$

donde:

coneDir: es la dirección de la hélice ya sea izquierda o derecha.

cantidad_de_vueltas: es la cantidad de vueltas por las cual pasará la hélice.

pitch_diameter_pinion, pitch_angle_pinion, pitch_angle_pinion: se encuentran definidos en la tabla 1.1

- Paso 3: se dibujó un boceto con líneas de construcción acotado por la mitad, luego se realizó el proceso de cara, la cual representa, la proyección del núcleo cónico sobre un plano. Dicho núcleo se encuentra estructurado por las ecuaciones de la geometría básicas de los engranajes cónicos propuestas en la tabla 1.2.
- Paso 4: después de haber obtenido la cara resultante se revolucionó tomando como referencia el eje de simetría, para lograr un núcleo cónico sólido conformado en 3 dimensiones.
- Paso 5: con las ecuaciones paramétricas de la envolvente de círculo comentada en capítulos anteriores, se procedió a realizar el espacio entre dientes. Se tomó como referencia una curva de involuta, sobre las circunferencias de fondo, base, referencia y cresta comentadas también en capítulos anteriores. Una vez obtenida la curva acotada se realizó un espejo dado por una distancia radial y para cerrar el contorno se hallaron los puntos de intercesión con las circunferencias de construcción y se crean los arcos respectivos.

  Obtenida la hélice descrita sobre el cono de referencia y el espacio entre dientes, se realizó un desplazamiento entre el punto medio de referencia del espacio entre dientes y el punto final de la hélice, pesándole el vector tangente que es hallado por la primera derivada de la curva. Ya ubicado el diente en posición tangente a la curva se realizó un barrido que va desde el espacio entre dientes pasando por la hélice y terminando en el punto tope de dicha hélice. Con el proceso anteriormente descrito se conformó un sólido el cual cumplirá como cuchilla de corte al núcleo realizado en pasos anteriores. Fue posible realizar a dicha cuchilla el proceso de fileteo en 3 dimensiones el cual consiste en colocar un arco de circunferencia tangente entre el perfil de involuta y la base del diente redondeando las esquinas, con el objetivo de solucionar los problemas de socavado y fallas en los dientes de los engranajes cónicos.
- Paso 6: se ubicó la herramienta de corte de forma tal que se comportara tangente a el núcleo anteriormente realizado.
- Paso 7: se dividió 360 grados entre el número de dientes para obtener el ángulo por el cual va a estar replicada la cuchilla y va a estar tantos números de dientes tenga el engranaje. Una vez hecha la réplica se procede a realizar un sólido común entre todas las cuchillas para lograr una sola figura de corte.
- Paso 8: por último, se procedió cortar el núcleo antes creado con la figura de corte para lograr el engranaje cónico.

3.1.4. Principales problemas planteados

- Ubicar el espacio entre dientes tangente a la hélice.

  Esto resultó un proceso engorroso debido a que se probó con varios métodos que proporciona las librerías de Open CASCADE y no se encontraba un resultado acertado. Para lograr el vector tangente en la solución se calculó el primer y último punto de la hélice y se calculó además la primera derivada partiendo de estos puntos obtenidos logrando así el vector tangente a dicha curva. Luego se realizó un desplazamiento entre el punto medio de referencia del espacio entre dientes y el punto final de la

hélice, pasándole el vector tangente anteriormente calculado, logrando así que la hélice y el espacio entre dientes fueran tangentes entre sí.

- Figura de corte.
  La figura de corte fue necesaria agruparla en un solo sólido debido a que al aplicarle el método de corte por separado esta no cortaba, solo marcaba la superficie con líneas por donde debía estar el corte.
- Tiempo de demora de la solución.
  Los métodos booleanos de corte, unión y común poseen un alto costo computacional por lo cual al aplicar el corte entre el núcleo cónico y la herramienta de corte mientras mayor sean las dimensiones del engranaje, mayor será el tiempo de respuesta de la pieza. Para darle solución a este problema se investigó y Open CASCADE en la versión OCCE 0.18 existen soluciones para esto, partiendo de la función *Fuzzy* la cual es capaz de realizar las operaciones booleanas en menor tiempo. Para ello se emigró a la versión siguiente y se aplicaron los métodos pertinentes.
  Se realizó una prueba con el método booleano y luego con el método de *Fuzzy* en un ordenador de propiedades I3-2120 CPU @ 3.30 GHz y 4GB de RAM, compartiendo el mismo juego de datos (20 número de dientes del piñón, 40 número de dientes de la corona, módulo 10mm, ángulo de la hélice 20°, 20mm de cara del engranaje y ángulo de inclinación 90°) se arrojaron los siguientes resultados: con el método booleano 50s y luego de aplicar el método de *Fuzzy* 17s donde se aprecia una reducción en cuanto al tiempo de modelado de 34%.

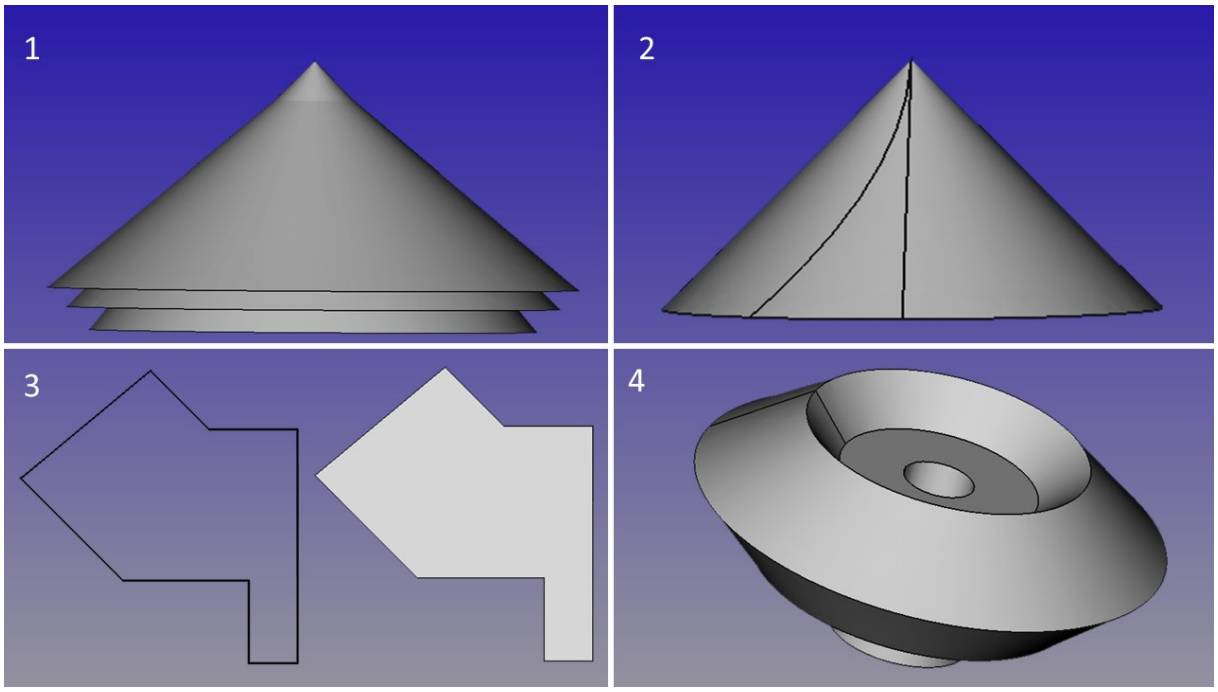


Figura 3.3. Pasos para conformar el engranaje (Simplificado).

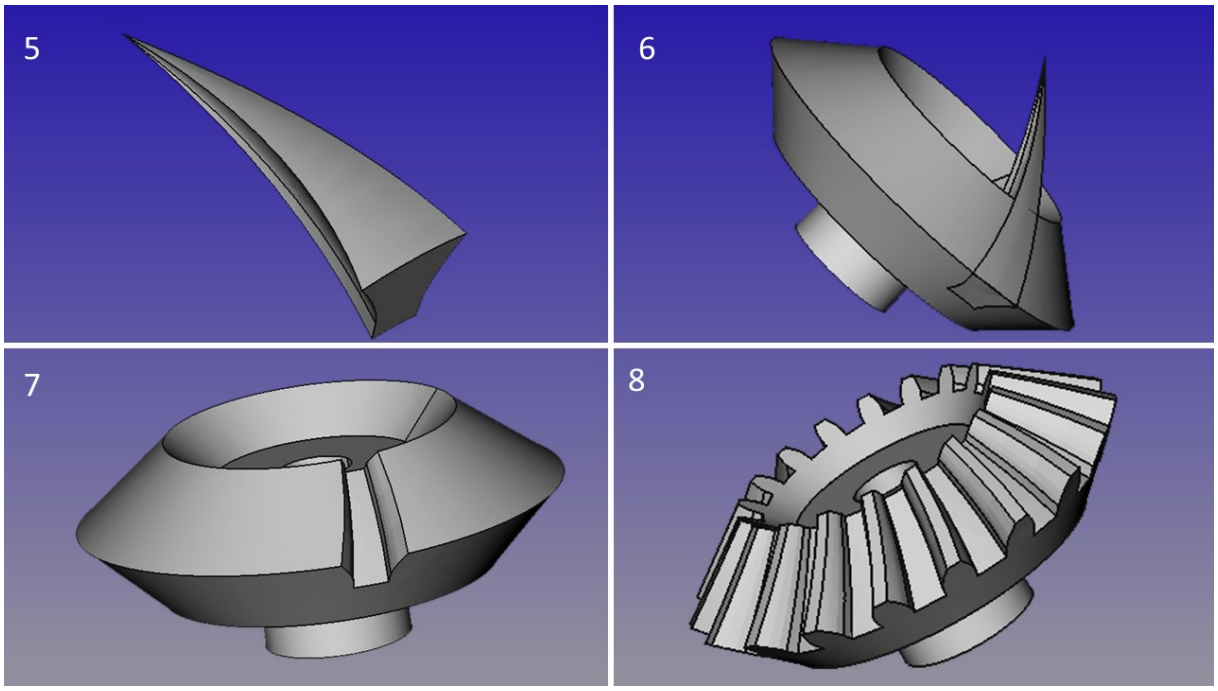


Figura 3.4. Pasos para conformar el engranaje (Simplificado).

3.2. Diagrama de componentes

Un diagrama de componentes 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(JACOBSON, RUMBAUGH y BOOCH, 2000) (ver Figura 3.5).

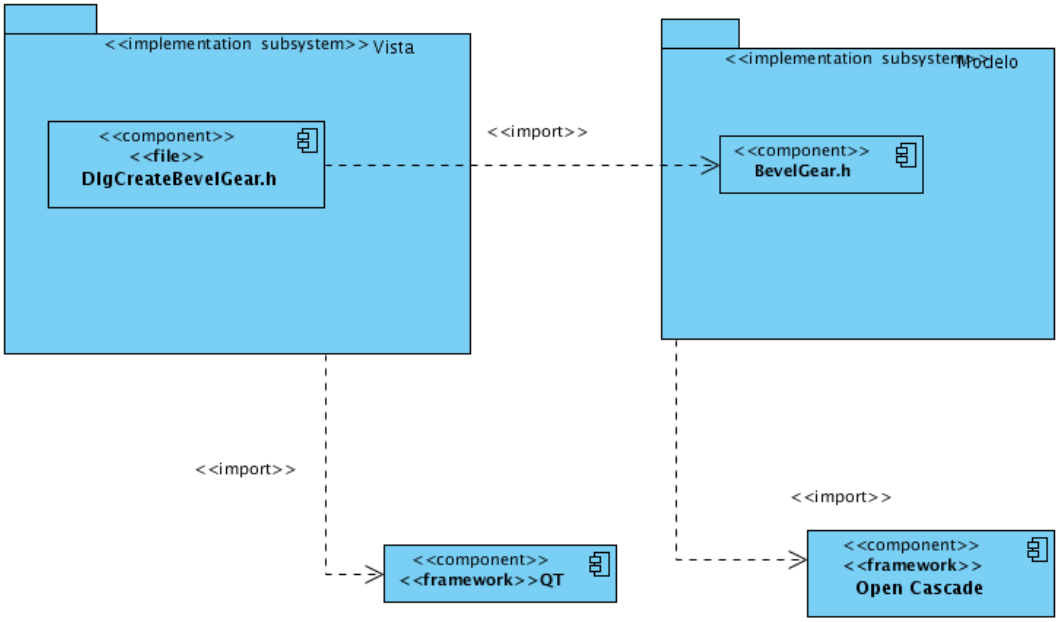


Figura 3.5. Diagrama de componentes.

3.3. Pruebas

El proceso de pruebas de software tiene dos objetivos distintivos:

1. 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.

2. 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.

3.3.1. Niveles de prueba

Para aplicarle pruebas a un sistema se deben tener en cuenta una serie de objetivos en diferentes escenarios y niveles de trabajo, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo. Los niveles de prueba son (RUIZ TENORIO, 2010):

Prueba Unitaria o de Unidad: se centra en el proceso de verificación de la menor unidad del diseño del software: el componente de software o módulo. Se emplea para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control inapropiados. Las pruebas del camino básico y de bucles son técnicas muy efectivas para descubrir una gran cantidad de errores en los caminos.

Prueba de Integración: es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba unitaria y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Pruebas de Sistema: está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. Algunas de estas son: pruebas de recuperación, seguridad, resistencia, entre otras.

Pruebas de Aceptación: es la realización de una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos. Para validar el correcto funcionamiento de la herramienta se le realizarán pruebas Unitarias, de Sistema y de Aceptación. Para validar el correcto funcionamiento de la herramienta se le realizarán pruebas Unitarias, de Sistema y de Aceptación.

subsectionMétodos de prueba El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Para llevar a cabo este objetivo, se emplearán los dos métodos de prueba:

- **Prueba de Caja Blanca:** se centran en la estructura de control del programa. Se obtienen casos de prueba que aseguren que durante la prueba se han ejecutado, por lo menos una vez, todas las sentencias del programa y que se ejercitan todas las condiciones lógicas.
- **Prueba de Caja Negra:** son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa, solo se fijan en las funciones que realiza el software. Las técnicas de prueba de caja negra se centran en el ámbito de información de un programa, de forma que se proporcione una cobertura completa de prueba.

3.3.2. Pruebas de aceptación

Para la prueba de aceptación se realizó la simulación mediante el método de elementos finitos, con la herramienta Salome-Meca, de dos modelos generados con el módulo, uno recto y el otro helicoidal. El mallado y posterior simulación con un estado de carga estática permitió verificar la integridad de los modelos geométricos (ver Figura 3.6).

3.3.3. Diseño de Casos de Prueba

Los Casos de Pruebas han sido realizados sobre la base de las Historias de Usuarios y tienen como objetivo fundamental encontrar la mayor cantidad posible de deficiencias existentes en las funcionalidades implementadas (ver Tabla 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 y 3.8).

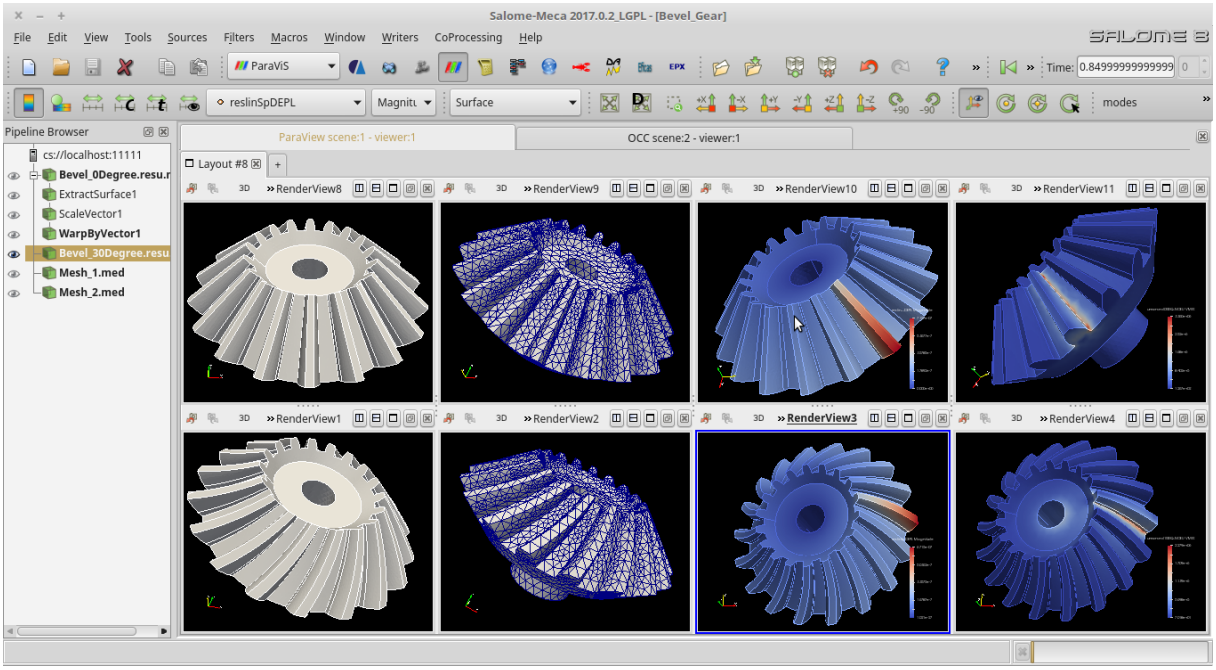


Figura 3.6. Pruebas de aceptación con la herramienta Salome Meca

Tabla 3.1. Diseño de Casos de Prueba RF 1

Descripción general			
Insertar parámetros			
SC 1 Insertar parámetros comunes para el piñón y la corona.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de insertar relación de transmisión.	Se introduce el valor de la relación de transmisión que va desde 1 hasta 40.	Proporciona el número de dientes de la corona en caso de que la relación de transmisión sea mayor que 1.	Accelerators/BevelGear /DlgCreateBevel
EC 1.2 Opción de insertar el ancho de la cara.	Seleccionar un valor comprendido entre la tercera parte del cono exterior	Muestra el ancho de la cara del engranaje.	Accelerators/BevelGear /BevelGear
EC 1.3 Opción de insertar módulo.	Se introduce el valor del módulo del engranaje que va desde 0.050 hasta 100.	Proporciona las dimensiones del engranaje	Accelerators/BevelGear /BevelGear
EC 1.4 Opción de insertar el ángulo de inclinación.	Se introduce el valor del ángulo de inclinación del engranaje.	Proporciona las dimensiones del engranaje	Accelerators/BevelGear /BevelGear

EC 1.5 Opción de insertar ángulo de presión.	Se introduce el valor del ángulo de presión del diente del engranaje que va desde 14.5 hasta 30 grados.	Proporciona las dimensiones del diente del engranaje	Accelerators/BevelGear /BevelGear
EC 1.6 Opción de insertar ángulo de la hélice.	Se introduce el valor del ángulo de la hélice del engranaje que va desde 0 hasta 45 grados.	Proporciona las dimensiones de curvatura del diente del engranaje	Accelerators/BevelGear /BevelGear
EC 1.7 Opción de seleccionar el tipo de diente.	Se selecciona tipo de diente del engranaje.	Proporciona las dimensiones del diente del engranaje	Accelerators/BevelGear /BevelGear
EC 1.8 Opción de seleccionar la separación de corte.	Se selecciona la separación de corte diente del del engranaje.	Proporciona las dimensiones del diente del engranaje	Accelerators/BevelGear /BevelGear
EC 1.9 Opción de seleccionar el radio de corte.	Se selecciona el radio de corte diente del del engranaje por las diferentes normas.	Proporciona las dimensiones del diente del engranaje	Accelerators/BevelGear /BevelGear

Tabla 3.2. Diseño de Casos de Prueba RF 2

Descripción general			
Insertar parámetros			
SC 1 Insertar parámetros del piñón.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de insertar el número de dientes del piñón.	Se introduce el valor del número de dientes del piñón	Proporciona el número de dientes del piñón.	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.3. Diseño de Casos de Prueba RF 3

Descripción general			
Insertar parámetros			
SC 1 Insertar parámetros de la corona.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de insertar el número de dientes de la corona.	Se introduce el valor del número de dientes de la corona.	Proporciona el número de dientes de la corona.	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.4. Diseño de Casos de Prueba RF 4

Descripción general			
Insertar parámetros			
SC 1 Seleccionar tipo de engranaje cónico.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de seleccionar el tipo de engranaje cónico.	Se selecciona el tipo del el tipo de engranaje cónico.	Proporciona el tipo de de engranaje cónico	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.5. Diseño de Casos de Prueba RF 5

Descripción general			
Insertar parámetros			
SC 1 Seleccionar piñón o piñón-corona para modelar.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de seleccionar el piñón o piñón-corona para modelar.	Se selecciona el tipo de piñón o piñón-corona para modelar.	Proporciona el tipo de componente para modelar.	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.6. Diseño de Casos de Prueba RF 6

Descripción general			
Cálculo			
SC 1 Mostrar resultado de los cálculos.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar el resultado de los cálculos.	Se muestran los resultado de los cálculos.	Proporciona el cálculo de engranajes cónicos por los cuales se va a realizar el modelado.	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.7. Diseño de Casos de Prueba RF 7

Descripción general			
Modelado			
SC 1 Realizar modelado del piñón.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Realiza el modelado del piñón.	Realiza el modelado del piñón.	Proporciona una vista en tercera dimensión de el piñón.	Accelerators/BevelGear /DlgCreateBeve

Tabla 3.8. Diseño de Casos de Prueba RF 8

Descripción general			
Modelado			
SC 1 Realizar modelado del par piñón-corona.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Realiza el modelado del par piñón-corona.	Realiza el modelado del par piñón-corona.	Proporciona una vista en tercera dimensión del par piñón-corona.	Accelerators/BevelGear /DlgCreateBeve

3.3.4. Pruebas Unitarias

Para la realización de las pruebas unitarias se empleó Qt Test, el cual es un framework para pruebas de este tipo a aplicaciones y librerías. Qt Test provee todas las funcionalidades comunes de los framework de pruebas unitarias, así como extensiones para probar interfaces gráficas de usuario. A continuación se muestra una imagen de las pruebas unitarias realizadas durante el proceso de implementación, en la que se evidencia la aceptación de todas las verificaciones (ver Figura 3.7).

```
********* 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::calaculationpitch_diameter_pinion()
PASS   : TestUnitTest::calculationpitch_diameter_gear()
PASS   : TestUnitTest::calculationpitch_angle_pinion()
PASS   : TestUnitTest::calculationpitch_angle_gear()
PASS   : TestUnitTest::calaculationouter_cone_distance()
PASS   : TestUnitTest::calculationmean_cone_distance()
PASS   : TestUnitTest::calculationmean_working_depth()
PASS   : TestUnitTest::calculationclearance()
PASS   : TestUnitTest::calculationmean_whole_depth()
PASS   : TestUnitTest::calculationEquivalent_90_ratio()
PASS   : TestUnitTest::calculationmean_circular_pitch()
PASS   : TestUnitTest::calculationouter_working_depth()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 14 passed, 0 failed, 0 skipped, 0 blacklisted
```

Figura 3.7. Prueba unitaria realizada con Qt Test.

3.3.5. Resultados de las pruebas

La realización de pruebas funcionales permitió identificar una serie de no conformidades (ver Tabla 3.9).

Tabla 3.9. No conformidades detectadas en las pruebas

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	La cara del diente no posee la restricción de ser la tercera parte del diámetro externo.	Media	Resuelta
2	RF 1	El ángulo de la hélice y el ángulo de inclinación tienen que ser llevados a radianes.	Alta	Resuelta

3	RF 4	El tipo de engranaje no se selecciona correctamente.	Alta	Resuelta
4	RF 5	Puede ser seleccionado los dos “*radioButtom*”.	Media	Resuelta
5	RF 6	La tabla donde se muestran los cálculos se desproporciona.	Baja	Resuelta

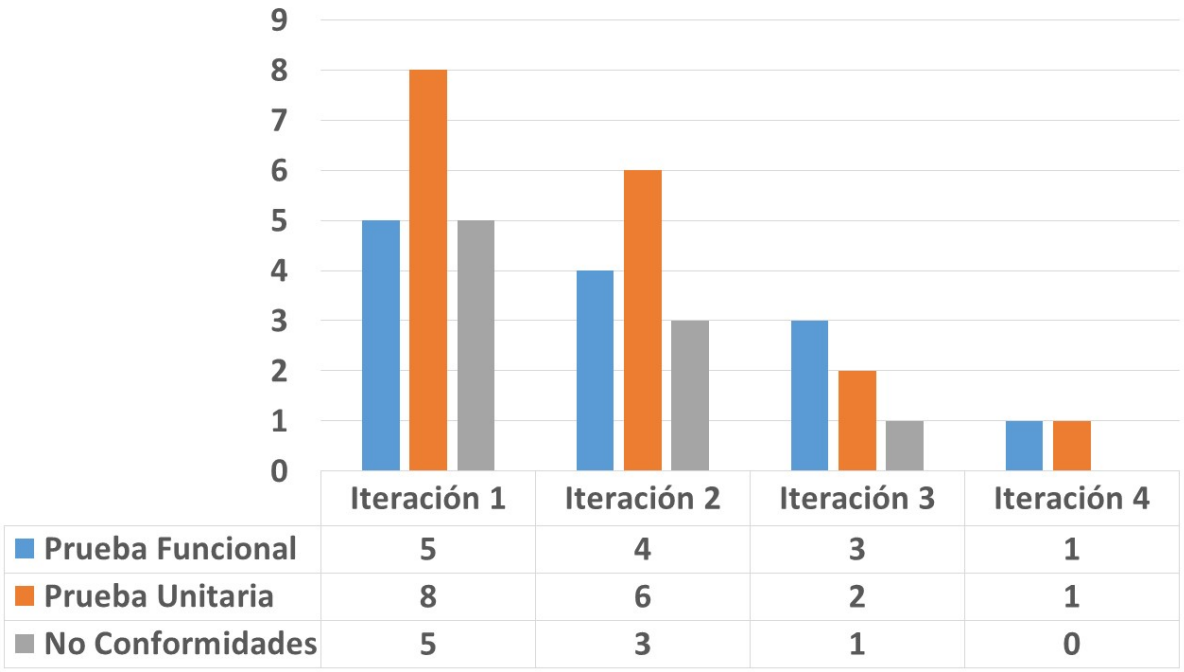


Figura 3.8. Resultados de las pruebas aplicadas al software en las distintas iteraciones.

En la primera iteración se realizó cinco pruebas funcionales, ocho unitarias y se detectaron cinco no conformidades; en la segunda iteración se realizó cuatro pruebas funcionales, seis unitarias y se detectaron tres no conformidades ; en la tercera iteración se realizó tres pruebas funcionales, dos pruebas unitarias y se detectaron una no conformidad; en la cuarta iteración se realizó una prueba funcional y una prueba unitaria, cumpliéndose todas las pruebas de manera satisfactoria sin ninguna no conformidad, dando por terminado dicho proceso de pruebas.

3.4. Conclusiones

En este capítulo se hizo referencia a los aspectos fundamentales en la implementación y la validación del módulo para el diseño de engranajes cónicos con dientes rectos o helicoidales.

La selección del estándar de codificación permitió lograr uniformidad en el código implementado.

Con la definición de una estrategia de pruebas se verificó el correcto funcionamiento del software. Las pruebas realizadas se enfocaron en el desarrollo y la evaluación de cada funcionalidad para lograr un mayor impacto en el usuario.

Se encontraron y solucionaron problemas como ubicar el espacio entre dientes tangente a la hélice, problemas en la figura de corte y el tiempo de demora de la solución.

3.5. Conclusiones generales

De la investigación realizada se concluye que:

- Con el módulo desarrollado se logra acelerar el proceso de diseño de engranajes cónicos con dientes rectos o helicoidales, reduciendo de esta manera el tiempo de modelado con respecto a procedimientos tradicionales.
- Las funcionalidades del módulo permiten el modelado de rasgos complejos en la topología del engranaje, así como la posibilidad de modelar el piñón o piñón-corona de forma independiente.
- La caracterización de los sistemas de código abierto permitió identificar las potencialidades y limitaciones para el modelado de engranajes cónicos, lo que permitió proyectar la idea de un módulo para acelerar el diseño de dicho componente mecánico.

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:

- Incorporar otras normas para el diseño y cálculo de engranajes cónicos de dientes rectos o helicoidales.
- Incluir los cálculos de resistencia de los engranajes cónicos.
- Incorporar el modelado del engranaje cónico hipoidal.
- Desarrollar sistema de base de datos para complementar el modelado de componentes normalizados.

Acrónimos

CAE Ingeniería Asistida por Computadora. 20, 22
CAM Fabricación Asistida por Computadora. 22

GPL Licencia Pública General. 23
GUI Interface Gráfica de Usuario. 23

JSON Notación de Objetos JavaScript. 24

LGPL Licencia Pública General Reducida. 19, 22, 23

MCAD Diseño Asistido por computadora Orientado a la Mecánica. 20

OCCT Tecnología Open CASCADE. 22, 23

XML Lenguaje Extensible de Enmarcado. 21, 24

Referencias bibliográficas

BALDASANO (2016). *DISEÑO ASISTIDO POR ORDENADOR (CAD). EVOLUCION Y PERSPECTIVAS DE FUTURO EN LOS PROYECTOS DE INGENIERIA*. URL: http://www.unizar.es/aeipro/finder/INGENIERIA%20DE%20PRODUCTOS/BF04.htm (vid. págs. 2, 3).

CATIA (2018). *CATIA*. CATIA. URL: http://www.3ds.com/es/productos-y-servicios/catiaTabla (vid. pág. 19).

eFunda (2018). *eFunda: About Us.* eFunda. URL: http://www.efunda.com/about/about.cfm (vid. pág. 21).

FreeCAD (2017). *About FreeCAD - FreeCAD Documentation*. FreeCAD. URL: http://www.freecadweb.org/wiki/index.php?title=About_FreeCAD (vid. pág. 20).

Inventor, Autodesk (2018). *Autodesk Inventor*. SEMCO. URL: http://www.semco.com.pe/web/curso_autodesk_inventor.html (vid. pág. 17).

JACOBSON, IVAR, JAMES RUMBAUGH y GRADY BOOCH (2000). *El Lenguaje Unificado de Modelado.* 2000 (vid. págs. 22, 23, 45, 50).

JEFFRIES, R., A. ANDERSON y C. HENDRICKSON (2001). *Extreme Programming Installed*. Ed. por Addison-Wesley (vid. pág. 30).

MITCalc (2018). *MITCalc - Mechanical, Industrial and Technical Calculations.* MITCalc. URL: http://www.mitcalc.com/ (vid. pág. 21).

PLM, Solid Edge: Siemens (2018). *Solid Edge*. Siemens PLM Software. URL: http://www.plm.automation.siemens.com/es_sa/products/solid-edge.html (vid. pág. 17).

PRESSMAN, ROGER S. (2003). *ngeniería de Software. Un enfoque práctico. 5ta Edición.* (Vid. pág. 22).

Pressman, Roger S (2010). *Software Engineering. s.l. : Higher Education*. 2010 (vid. págs. 38, 43).

QT (2018). *About Qt*. Qt. URL: ttps://qt.io/About_Qt (vid. pág. 24).

R Marambedu, Karthikeyan (2009). *DEVELOPMENT OF A PROCEDURE FOR THE ANALYSIS OF LOAD DISTRIBUTION, STRESSES AND TRANSMISSION ERROR OF STRAIGHT BEVEL GEARS*. Ed. por The Ohio State University. The Ohio State University (vid. pág. 2).

RUIZ TENORIO, ROBERTO (2010). *Las Pruebas de Software y su Importancia en las Organizaciones.* 2010 (vid. pág. 51).

Sanchez Rodriguez, Tamara (2015). *Metodoligia de desarrollo para la actividad productiva en la UCI* (vid. pág. 25).

SHIGLE, J.E. (1979). *Diseño en Ingeniería mecánica. 2da. Ediciones*. Ed. por Mc Graw Hill (vid. págs. 5, 14, 15).

Solid-Edge (2018). *Solid Edge Synchronous Technology: Siemens PLM Software.* Solid Edge. URL: https://www.plm.automation.siemens.com/en_us/products/solid-edge/design/synchronous-technologys.html (vid. pág. 16).

SOLIDWORKS (2018). *Productos de visualización de SOLIDWORKS*. SOLIDWORKS. URL: http://www.solidworks.es/sw/products/visualization/solidworks-visualization-overview.html (vid. pág. 18).

STANDARD, AMERICAN NATIONAL (2005). *Design Manual for Bevel Gears*. American Gear Manufacturers Association (vid. págs. 7, 15, 16).

STROUSTRUP, BJARNE (2013). *The C++ Programming Lenguage.* (Vid. pág. 23).

Technology, Open CASCADE (2018). *Open CASCADE Technology: Overview.* Open CASCADE Technology. URL: http://dev.opencascade.org/doc/overview/html/index.html#OCCT_OVW_SECTION_1 (vid. pág. 23).

Generado con LaTeX: 20 de junio de 2018: 12:42 Noon

Apéndices

APÉNDICE A

Imágenes de engranajes cónicos

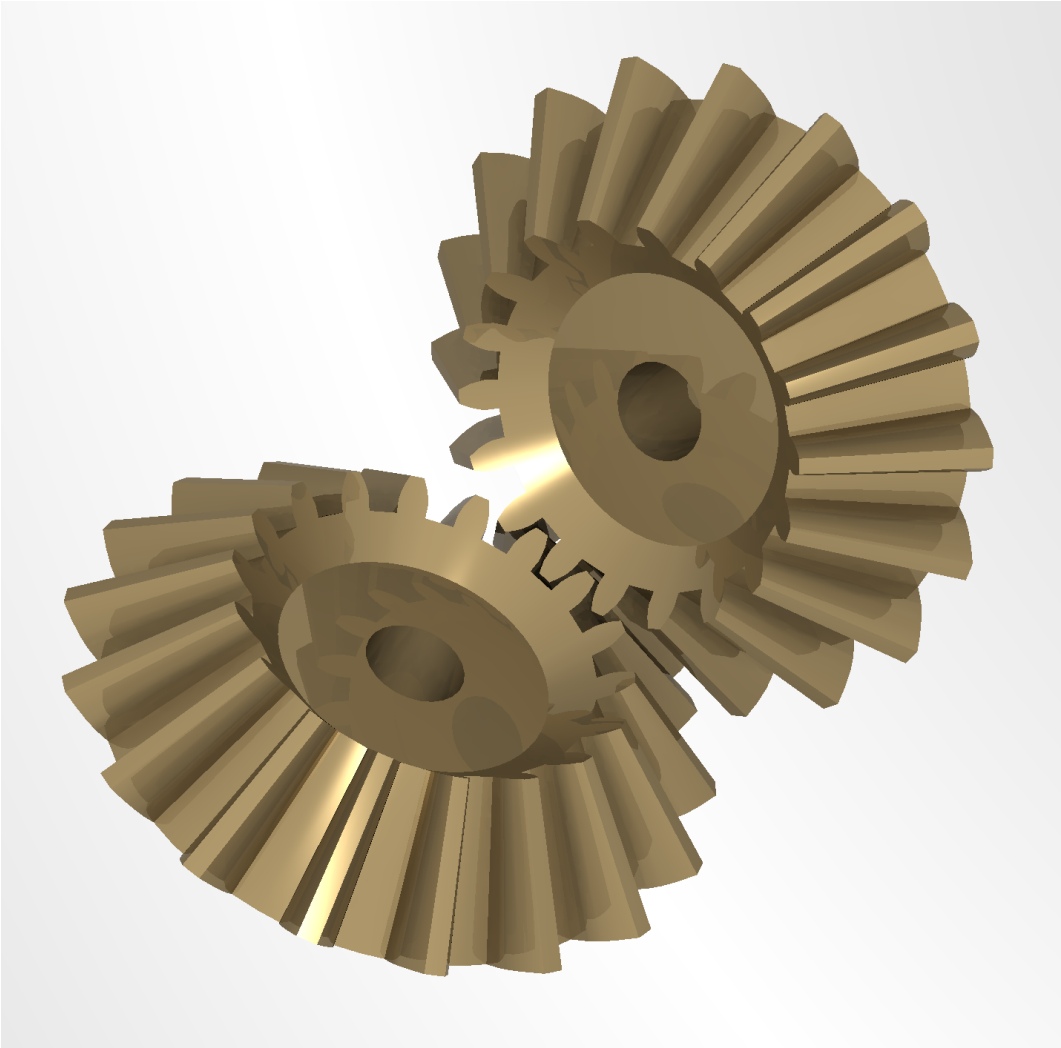


Figura A.1. Imagen realizada mediante el trazado de rayos.

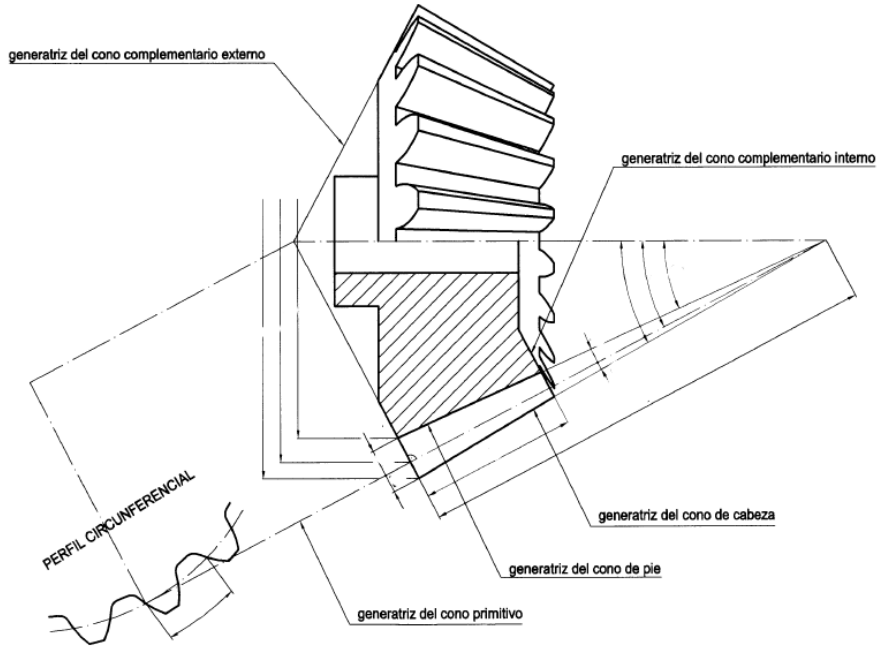


Figura A.2. Vista en perspectiva de engranajes cónicos 1.

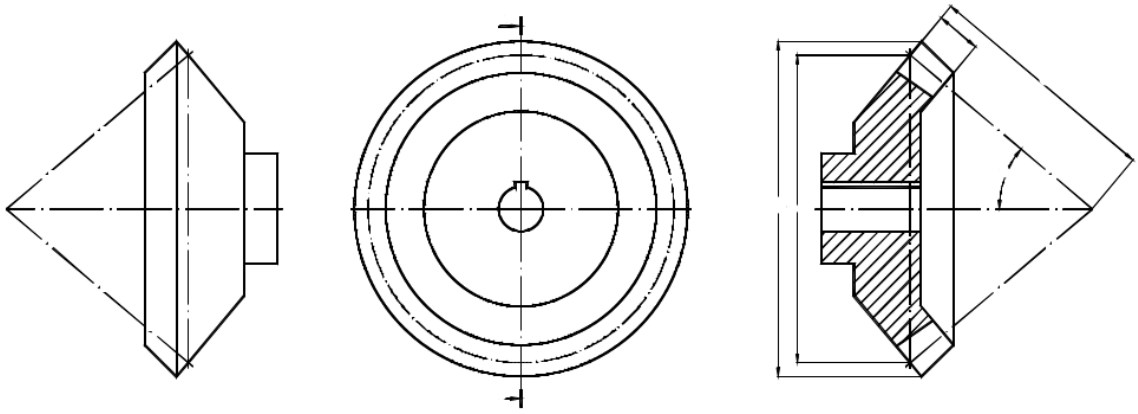


Figura A.3. Vista en perspectiva de engranajes cónicos 2.

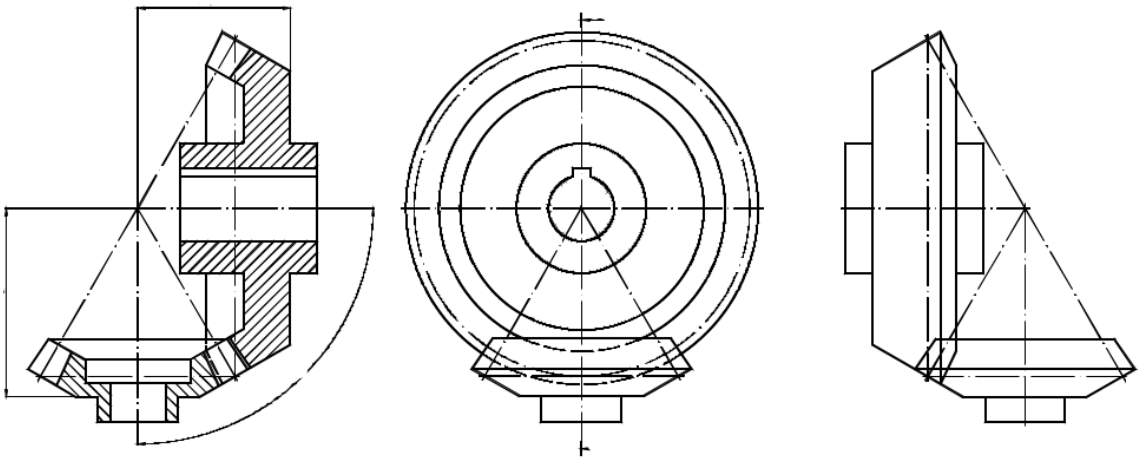


Figura A.4. Vista en perspectiva de engranajes cónicos 3.