Universidad de las Ciencias Informáticas
Facultad 4

Módulo para acelerar el diseño de engranajes piñón cremallera

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

Autor: Reimnel Drake Bueno.

Tutor: Dr.C Augusto César Rodríguez Medina.

La Habana, 2018

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Dedicatoria

A mi abuela Teresa, mi madre Mercedes, mi hermana Yanelis y a toda mi familia en general, por su apoyo e incondicionalidad, por la confianza depositada en mí y por guiarme durante mi recorrido estudiantil hasta el logro de mis metas.

Agradecimientos

Declaración de autoría

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

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

Reimnel Drake Bueno.	Dr.C Augusto César Rodríguez Medina.
Autor	Tutor

Resumen

En el presente documento se exponen los resultados del proceso de investigación cuyo propósito fundamental ha sido desarrollar un módulo informático capaz de acelerar el diseño de engranajes del tipo piñón cremallera. Las formulaciones empleadas en los cálculos se tomaron de las normas y fuentes especializadas, que fueron identificadas durante la investigación documental. El empleo del módulo obtenido reduce el tiempo de modelado de piñones y cremalleras, con dientes rectos o helicoidales y perfiles de involuta. Contiene una *interface* para la introducción de los datos iniciales, funciones para el modelado tomadas de la tecnología *Open CASCADE* y muestra el modelo generado en un visor de Coin3D. La importancia del resultado obtenido reside en que ofrece una alternativa basada en *software* libre para el modelado de engranajes piñón cremallera, contribuyendo a la soberanía tecnológica.

Palabras clave: Diseño Asistido por Computadora, engranaje piñón cremallera.

Índice general

Índice de figuras

Índice de tablas

Introducción

Entre las tendencias que se aprecian en los sistemas informáticos para el diseño en ingeniería, está la incorporación de módulos que aceleran el proceso de diseño de componentes mecánicos complejos. Para dar solución a carencias de la industria nacional se ha proyectado en el grupo de investigación Soluciones Informáticas Para la Ingeniería y la Industria (SIPII) desarrollar componentes de software con una filosofía similar a la de los sistemas comerciales. En el presente documento se exponen los resultados de un proceso de investigación y desarrollo, cuyo propósito fundamental ha sido generar un módulo para acelerar el diseño de un tipo especial de engranaje, compuesto por un piñón y un sector dentado lineal denominado cremallera, cuyos dientes pueden ser rectos o helicoidales en ambos casos. La complejidad de la idea requirió consultar diversas fuentes para evaluar su factibilidad a partir del análisis de la información recopilada, las tecnologías existentes y las metodologías de cálculo disponible.

El documento se encuentra estructurado en tres capítulos, en el primero se exponen los fundamentos de la investigación, especialmente la valoración de las exigencias y formulaciones para el diseño de engranajes tipo piñón cremallera, aunque también se incluyen consideraciones sobre soluciones similares de diseño asistido por computadora destinados al modelado de ese tipo de engranaje y las metodologías y herramientas para el desarrollo; en el segundo se expone la propuesta de solución, se realiza el levantamiento de los requisitos, el modelo de dominio, la descripción del componente, la arquitectura, los patrones de diseño, el modelo conceptual y las historias de usuarios; en el tercero se presentan elementos correspondientes a la implementación como: los estándares de codificación, las consideraciones sobre el módulo desarrollado y las pruebas realizadas al mismo.

CAPÍTULO 1

Aspectos generales del proceso de investigación

En el presente capítulo se exponen los aspectos fundamentales del proceso de investigación preliminar que motivaron la investigación, los fundamentos teóricos del diseño de engranajes tipo piñón cremallera y los principales aspectos de la metodología de desarrollo de *software*.

1.1. Aspectos de la situación problemática y el proceso de investigación

Un engranaje piñón cremallera está integrado por dos componentes: un engranaje circular denominado piñón que engrana y transfiere el movimiento hacia un tipo especial de dispositivo recto llamado cremallera, que consiste en una serie de dientes en línea recta sobre una superficie plana (Thirugnanam, Das y Rakesh, 2014). Los engranajes de este tipo permiten la conversión del movimiento de rotación de un piñón en uno de traslación por parte de la cremallera, o viceversa, el movimiento de traslación de la cremallera se puede convertir en un movimiento de rotación por parte del piñón (González, 2008). Un ejemplo del mencionado engranaje se ilustra en la figura 1.1.

La utilidad práctica de los engranajes piñón cremallera suele centrarse, principalmente, en la conversión del movimiento de rotación en traslación. Este mecanismo es muy apreciado para conseguir movimientos lineales de precisión como es el caso de microscopios u otros instrumentos ópticos como retroproyectores; también se pueden encontrar otras aplicaciones, algunos ejemplos se describen a continuación (ibíd.):

- **La dirección asistida**. El conjunto de mecanismos que componen el sistema de la dirección tienen la misión de orientar las ruedas delanteras para que el vehículo tome la trayectoria deseada por el conductor. Cuando se gira el volante de un automóvil, gira al mismo tiempo un piñón situado en el otro extremo del eje del volante, el cual engrana a una cremallera que, al desplazarse, posibilita el giro de las ruedas que le permiten cambiar la dirección del coche.
- **Las vías de los ferrocarriles en lugares en los que existe una gran pendiente en subida**. En este caso, se corre el riesgo de que el ferrocarril patine y es por eso que entre las vías se sitúa una cremallera que engrana con una rueda dentada motriz adosada al tren. Al girar, facilita la subida de la fuerte

Figura 1.1. Ejemplo de engranaje piñón cremallera.

pendiente sin riesgo de deslizamiento.

- **Las puertas correderas**, especialmente de aquellas con acceso a un aparcamiento que se activan con un mando a distancia. El mando a distancia activa un motor eléctrico cuyo eje lleva acoplado un piñón, mientras que la cremallera está adosada a la puerta. Al girar el piñón, la puerta se desplazarse gracias a la cremallera.
- **Los elevalunas manuales de un automóvil**. Cuando queremos subir la ventanilla de un coche, de forma manual, lo que hacemos es girar, además de la manivela, un piñón acoplado a una cremallera curva que tiene en un extremo una palanca articulada.

El proceso de producción de este u otro tipo de engranajes, se inicia en la etapa de concepción y diseño, la cual, con los resultados del avance científico técnico de los últimos cincuenta años ha sufrido transformaciones en la manera de realizarse. Este proceso ha transitado desde las tecnologías para la elaboración de planos basadas en la mesa de dibujo e implementos para el trazado, hasta las informáticas que logran elevados niveles de automatización, y se conocen como: Diseño Asistido por Computadora (CAD, por sus siglas en inglés).

Existen varios problemas en la industria nacional relacionados con esta área entre los que podemos encontrar: los elevados costos que tienen estas tecnologías y las restricciones del bloqueo impuesto por el gobierno de los Estados Unidos contra Cuba. En las entidades de ingeniería y diseño en nuestro país no es posible institucionalizar el empleo y desarrollo de estos sistemas; las soluciones implementadas en nuestras empresas por especialistas y profesionales son, en general, sistemas “sin licencia”, lo que trae consigo graves

riesgos en cuanto a seguridad informática se refiere. En tales condiciones no se puede planificar el desarrollo tecnológico de los procesos de diseño e ingeniería, exportar proyectos o prestar servicios científico técnicos en el exterior utilizando un sistema con las características antes mencionadas.

En caso de no existir el bloqueo, implementar tecnologías comerciales en la industria nacional sería insostenible, pues su principal forma de comercialización es mediante licencias de uso por un tiempo definido y para grupos de máquinas computadoras; esta situación requiere un financiamiento considerable para mantener estándares tecnológicos adecuados, lo que puede ser complicado incluso para empresas importantes. Algunos de los sistemas comerciales más difundidos son: AutoCAD e Inventor de Autodesk, Inc.; SolidWorks de SolidWorks Corp., una filial de Dassault Systèmes S.A; Solid Edge de Siemens PLM Software, entre otros que son dueños de ese mercado que lucra miles de millones de dólares al año. Un ejemplo de los precios de las licencias de estas herramientas se puede constatar en la tabla 1.1.

Tabla 1.1. Tabla de costo de las licencias de los *software* comerciales. (Asidek, 2018a), (Asidek, 2018b), (Araworks, 2018), (PLM, 2018)

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

Para un mejor entendimiento de la tabla 1.1 vamos a describir un ejemplo hipotético suponiendo el cese del bloqueo a nuestro país y el acceso de nuestras empresas a estas tecnologías. Una empresa decide utilizar el *software* Solid Edge Premium para 10 de sus diseñadores a los cuales destina el mismo número de máquinas. El costo de la licencia de este producto por un mes es de 329 USD para una estación de trabajo, por lo que para 10 estaciones ascendería a 3290 USD. Concluimos que por cada año que la empresa decida utilizar el *software* Solid Edge Premium en 10 de sus máquinas tendría que pagar 39480 USD por concepto de licencia.

Existen varios emprendimientos para investigar sobre las tecnologías para el diseño mecánico en la Facultad 4 de la SIPII, uno de ellos es la aplicación AXISMEC, perteneciente al Centro de Entornos Interactivos 3D, pero este no posee módulos que acelerador de diseño de determinado tipo de engranajes; otro de ellos es el grupo de investigación SIPII que pertenece al departamento de Ciencias Básicas; el objetivo del grupo es aportar una contribución modesta y de implementación paulatina para solucionar los problemas asociados a las tecnologías para el diseño mecánico con medios propios y ahorrarle recursos financieros al país mediante la realización de proyectos de investigación relacionados con el desarrollo de dichas tecnologías. Este grupo posee una aplicación para el diseño asistido por computadoras a la cual se le han

integrado varios módulos para acelerar el proceso de diseño de piezas complejas como resultado de dichas investigaciones. Actualmente se ha decidido añadir el engranaje piñón cremallera.

Teniendo en cuenta la imposibilidad de adquirir sistemas de alta tecnología como las herramientas para el diseño mecánico comerciales, la necesidad de una aplicación que garantice los disímiles usos en la industria de estos elementos mecánicos, se define como problema a resolver la inexistencia de un componente para acelerar el diseño de engranajes piñón cremallera en el contexto de la ingeniería y el diseño a nivel nacional, como parte de una plataforma informática legalmente adquirida.

El **objeto de estudio** de la investigación se centra en el gráfico de computadoras para aplicaciones industriales, teniendo como **campo de acción**, el desarrollo de componentes CAD.

Para la solución del problema antes mencionado se propone como **problema de investigación**: ¿Cómo automatizar el proceso de diseño de engranajes tipo piñón cremallera, con base en tecnologías de código abierto?

Y para darle solución al problema de investigación formulado, se propone como **objetivo general**: Desarrollar un componente de *software* para acelerar el diseño de engranajes tipo piñón cremallera, mediante el empleo de tecnologías de código abierto.

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

1. Caracterizar los sistemas de código abierto en relación con sus potencialidades y limitaciones para el modelado de engranajes piñón cremallera.
2. Identificar las características y funcionalidades necesarias que debe tener un módulo para acelerar el modelado de engranajes piñón cremallera.
3. Obtener un módulo informático para acelerar el modelado de engranajes piñón cremallera.

Para darle cumplimiento al objetivo, se proponen las siguientes **tareas de investigación** a cumplir:

- Diseño del perfil de la investigación.
- Estudio bibliográfico sobre el objeto de investigación.
- Estudio de la fundamentación teórico-matemática de los engranajes piñón cremallera.
- Consulta de las metodologías de diseño de engranajes piñón cremallera existentes con el propósito de establecer las variables y parámetros necesarios para el diseño de los mismos.
- Definición de los patrones de diseño con los que cumplirá el componente.
- Estudio de los aspectos de la metodología de desarrollo de *software* (AUP-UCI) que se aplicarán a 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 en los diseños de engranajes tipo piñón cremallera.
- Definición de la arquitectura del componente, lo que implica 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 del componente destinado a la aceleración del proceso de diseño de engranajes tipo piñón cremallera.
- Integración del componente en la aplicación del proyecto.
- Pruebas al componente (de las diferentes funcionalidades y de integración).

Con el fin de resolver y dar cumplimiento a los objetivos y las tareas propuestas se empleó como **métodos de investigación**:

Métodos teóricos:

- **Análisis y síntesis**: se empleó en la construcción del marco teórico de la investigación y la comprensión de los sistemas comerciales para, de esta manera, realizar una captura de requisitos con las principales funcionalidades de dichos sistemas.
- **Modelado**: se empleó en la generación de los artefactos para lograr una mejor comprensión entre el equipo de desarrollo y las personas relacionadas.

Métodos empíricos:

- **Observación**: se empleó para caracterizar las soluciones, teniendo en cuenta potencialidades y carencias de otras aplicaciones destinadas al diseño mecánico y así establecer los requisitos con las principales funcionalidades de estas.

1.2. Generalidades sobre los engranajes piñón cremallera

Los engranajes están presentes en muchas de las máquinas que se pueden encontrar, tanto en el mundo industrial como en el doméstico. Estos promueven el movimiento de las ruedas y hélices de los medios de transporte, ya sea por tierra, mar o aire. Un ejemplo de engranaje lo constituyen el denominado mecanismo de cremallera, el cual aplicado a los engranajes lo constituyen una barra con dientes, la cual es considerada como un engranaje de diámetro infinito y un engranaje de diente recto de menor diámetro, y sirve para transformar un movimiento de rotación del piñón en un movimiento lineal de la cremallera (Costas Rodríguez y Gómez García, 2011).

1.2.1. Perfil de involuta del diente

Los dientes del engranaje se pueden fabricar con una amplia variedad de formas y perfiles. El perfil de involuta (o evolvente de un círculo) es el sistema más utilizado para el engranaje actual (ibíd.).

Una involuta es una curva que se traza por un punto en un cordón tenso que se desenrolla de un círculo. La involuta es una forma de espiral, cuya curvatura se vuelve más recta a medida que se extrae de un círculo

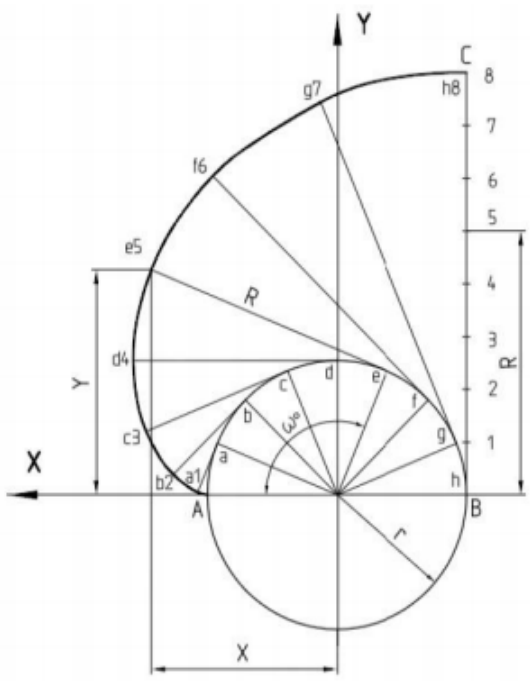


Figura 1.2. Trayectoria de la Involuta o envolvente de circulo.

base y eventualmente se convertiría en una línea recta si se dibuja lo suficiente. En la figura 1.2 se ilustra la trayectoria que describe la curva de la involuta (Costas Rodríguez y Gómez García, 2011).

El perfil del diente involuto de los engranajes más pequeños es considerablemente curvado, en engranajes más grandes es menos curvado (más recto), y es recto en una cremallera, que es esencialmente un engranaje infinitamente grande (ibíd.).

1.2.2. Geometría del diente

Para definir una transmisión hay que conocer los elementos que definen la geometría de los engranajes. Entre sus principales elementos se encuentran (Tulio Piovan, 2014):

- **Dientes de un engranaje:** Son los que realizan el esfuerzo de empuje y transmiten la potencia desde los ejes motrices a los ejes conducidos. El perfil del diente, o sea la forma de sus flancos, está constituido por dos curvas envolventes de círculo, simétricas respecto al eje que pasa por el centro del mismo.
- **Módulo:** Es la relación que existe entre el diámetro primitivo de un engranaje, expresado en milímetros, y el número de dientes. En los países anglosajones se emplea otra característica llamada Diametral Pitch[1], que es inversamente proporcional al módulo. Para que dos engranajes engranen tienen que tener el mismo módulo.

[1]El diametral pitch o paso diametral es el cociente entre el número de dientes y el diámetro primitivo, expresado en pulgadas.

- **Espacio entre dientes:** Espacio entre dientes medido sobre la circunferencia primitiva.
- **Circunferencia primitiva:** Es la circunferencia a lo largo de la cual engranan los dientes. Con relación a la circunferencia primitiva se determinan todas las características que definen los diferentes elementos de los dientes de los engranajes.
- **Paso circular:** Es la longitud de la circunferencia primitiva correspondiente a un diente y un espacio consecutivos.
- **Espesor del diente:** Es el grosor del diente en la zona de contacto, o sea, del diámetro primitivo.
- **Número de dientes:** Es el número de dientes que tiene el engranaje.
- **Diámetro exterior:** Es el diámetro de la circunferencia que limita la parte exterior del engranaje.
- **Diámetro interior:** Es el diámetro de la circunferencia que limita el pie del diente.
- **Pie del diente:** También se conoce con el nombre de dedendum. Es la parte del diente comprendida entre la circunferencia interior y la circunferencia primitiva.
- **Cabeza del diente:** También se conoce con el nombre de adendum. Es la parte del diente comprendida entre el diámetro exterior y el diámetro primitivo.
- **Cara del diente:** Superficie del diente entre la circunferencia primitiva y la de pie.
- **Flanco del diente:** Es la cara interior del diente, es su zona de rozamiento.
- **Altura del diente:** Es la longitud que tiene el diente del engranaje. Se representa por la suma de la altura de la cabeza (adendum) más la altura del pie (dedendum).
- **Angulo de presión:** Ángulo de la línea de acción (línea formal sobre la superficie del diente en el punto de contacto entre dos engranajes) con la tangente a la circunferencia de paso.
- **Distancia entre centro de dos engranajes:** Es la distancia que hay entre los centros de los engranajes.

La metodología de cálculo para engranajes de este tipo se conforma de varias fórmulas las cuales se ilustran en la tabla 1.2 para el caso de dientes rectos y en la tabla 1.3 para dientes helicoidales (Michael, 2015).

Tabla 1.2. Metodología de cálculo para el engranaje piñón cremallera de dientes rectos.

Término	Símbolo	Fórmula
Módulo	m	valor introducido por el usuario
Ángulo de presión	α	valor introducido por el usuario
Número de dientes	z	valor introducido por el usuario
Altura de la línea de paso	H	valor introducido por el usuario
Ángulo de presión de trabajo	α_w	valor introducido por el usuario
Distancia de montaje	a	$\frac{z * m}{2} + H + xm$

Diámetro de referencia	d	$z * m$
Diámetro base	d_b	$d * \cos(\alpha)$
Diámetro de paso de trabajo	d_w	$\frac{d}{\cos(\alpha_w)}$
Addendum	h_a	$1 * m$
Dedendum	h_d	$1.25 * m$
Profundidad del diente	h	$2.25 * m$
Diámetro de cresta	d_a	$d + 2h_a$
Diámetro de fondo	d_f	$d_a - 2h$

Tabla 1.3. Metodología de cálculo para el engranaje piñón cremallera de dientes helicoidales.

Término	Símbolo	Fórmula
Módulo normal	m_n	valor introducido por el usuario
Ángulo de presión normal	α_n	valor introducido por el usuario
Ángulo de la hélice del cilindro de referencia	β	valor introducido por el usuario
Número de dientes	z	valor introducido por el usuario
Coeficiente de perfil de cambio normal	x_n	valor introducido por el usuario
Altura de la línea de paso	H	valor introducido por el usuario

Ángulo de presión de transversal	α_t	$\arctan(\frac{\tan(\alpha_n)}{\cos(\beta)})$
Distancia de montaje	a	$\frac{z * m_n}{2\cos(\beta)} + H + x_n * m_n$
Diámetro de referencia	d	$\frac{z * m_n}{\cos(\beta)}$
Diámetro base	d_b	$d * \cos(\alpha_t)$
Addendum	h_a	$m_n * (1 + x_n)$
Profundidad del diente	h	$2.25 * m_n$
Diámetro de cresta	d_a	$d + 2h_a$
Diámetro de fondo	d_f	$d_a - 2h$

1.3. Componentes de *software* para el cálculo y modelado de engranajes piñón cremallera

En la actualidad existen herramientas comerciales destinadas al diseño mecánico que realizan el modelado y cálculo de los engranajes piñón cremallera. Estos brindan resultados precisos a dichos cálculos y aceleran el proceso de diseño llevado a cabo por ingenieros, diseñadores y arquitectos. En este apartado se estudiarán algunas de estas herramientas desarrolladas para diferentes plataformas y tecnologías.

1.3.1. Herramientas de *software* comerciales

Autodesk Inventor

Autodesk Inventor proporciona soluciones de ingeniería y diseño de grado profesional para diseño mecánico en 3 dimensiones (3D), simulación, herramientas, visualización y documentación. Con el *software* Inventor, los ingenieros pueden integrar dibujos en 2 dimensiones (2D) AutoCAD y los datos 3D en un solo modelo digital, la creación de una representación virtual del producto final que permita validar la forma, el ajuste y la función del producto antes de que se haya construido. El Inventor permite diseñar, visualizar y simular productos digitalmente, lo que ayuda a reducir los costes de desarrollo, llegar al mercado más rápido, y hacer grandes productos (Inventor, 2017).

Este *software* no posee un acelerador de diseño para el cálculo del engranaje piñón cremallera, en cambio, si lo posee para el piñón que puede ser de dientes rectos o helicoidales, el cual es posible construir mediante la inserción de parámetros como el ángulo de presión, módulo, número de dientes, entre otros. Para la construcción de la cremallera se dibuja un engranaje y con la opción que tiene el Inventor de exportar un diente, se copia e inserta en un prisma recto hecho con la forma de la cremallera y se replica, según el número de dientes que tenga, utilizando el paso de la misma. Estos dos componentes luego se engranan.

Solid Edge

Solid Edge es un portafolio de herramientas de *software* asequibles y fáciles de usar que abordan todos los aspectos del proceso de desarrollo de productos y 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 (PLM, 2018).

Esta herramienta posee un acelerador de diseño para el cálculo del engranaje piñón cremallera, la cual se encuentra ubicada en *Engineering Reference* en el modo *Assemble* seleccionando la opción *Rack and Pinion Gear Designer* lo que activa la ventana principal del módulo que visualiza el engranaje en varios pasos: primero se entran los datos correspondientes a cada engranaje, luego se seleccionan varias opciones como el tipo de engranaje que, en el caso de la cremallera ofrece la opción de visualizarse de varias maneras (rectangular, circular interno o circular externo), también se puede seleccionar el tipo de material de cada componente, el cálculo de las dimensiones del engranaje, entre otras opciones. Este engranaje no posee un módulo que acelere el diseño de los engranajes piñón cremallera de dientes helicoidales.

SolidWorks

SolidWorks es un programa que permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas destinados al diseño mecánico. El proceso consiste en trasvasar la idea mental del diseñador a la herramienta, construyendo virtualmente la pieza o conjunto.

Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada (SolidWorks, 2017).

Esta herramienta, al igual que Inventor, no posee un acelerador de diseño para el engranaje piñón cremallera, sin embargo, contiene una biblioteca de diseño que brinda la posibilidad de cargar una serie de piezas mecánicas ya construidas entre las que se encuentran el piñón y la cremallera; de estas piezas es posible editar sus propiedades y obtener un nuevo componente mecánico.

1.3.2. Herramientas de *software* libre

FreeCAD

FreeCAD es una aplicación libre de diseño asistido por computadora en tres dimensiones, ingeniería asistida por computadora, para la asistencia en ingeniería mecánica y el diseño de elementos mecánicos. Está basado en OpenCascade y programado en los lenguajes C++ y Python; presenta un entorno de trabajo similar a Catia, SolidWorks, SolidEdge, ArchiCAD o Autodesk Revit; utiliza técnicas de modelado paramétrico y está provisto de una arquitectura de *software* modular, pudiendo añadir de forma sencilla funcionalidades sin tener que cambiar el núcleo del sistema (FreeCAD, 2017).

A diferencia de los *software* para el diseño mecánico analíticos tradicionales, como pueden ser AutoCAD o Microstation, FreeCAD es un *software* paramétrico que utiliza parámetros para definir sus límites o acciones. En el diseño paramétrico cada elemento del dibujo (muros, puertas, ventanas, etc.) es tratado como un objeto, el cual no es definido únicamente por sus coordenadas espaciales (x, y, z), sino también por sus parámetros, ya sean estos gráficos o funcionales. Las bases de datos relacionadas con el objeto hacen que este *software*, y especialmente su banco de trabajo de arquitectura, esté muy relacionado con el enfoque BIM, en el que un modelo BIM contiene el ciclo de vida completo de la construcción, desde el concepto hasta la edificación (ibíd.).

Como muchos modernos modeladores mecánicos en tres dimensiones, tiene un componente para dos dimensiones para extraer un diseño detallado de un modelo en tres dimensiones y con ello producir dibujos en dos dimensiones, pero el diseño directo en dos dimensiones (como el de AutoCAD LT) no es la meta, ni tampoco la animación ni formas orgánicas (como las creadas por Maya, 3ds Max o Cinema 4D). FreeCAD posee elementos que se reutilizaran como la polilínea, el arco de circunferencia, el círculo, el punto y la recta, además de módulos de parametrización y de mayado de entidades (ibíd.).

FreeCad, al igual que Inventor y SolidWorks, no posee un acelerador de diseño para el cálculo del engranaje piñón cremallera, pero incluye una herramienta en el banco de trabajo *Part Design* para crear engranajes cilíndricos de ejes paralelos. Los dientes de estos engranajes pueden ser rectos, helicoidales o helicoidales dobles. En *Part Design* se selecciona la opción *Involute Gear* y aparece el perfil de un engranaje, con los parámetros por defecto, ubicados en un panel a la izquierda del perfil en donde se pueden editar sus dimensiones. Para obtener el engranaje se extruye el perfil, otorgándole el grosor deseado. Para construir la cremallera es necesario seguir los siguientes pasos: dibujar un diente haciendo un boceto o utilizando la figura de un trapecio, usar la extrusión para hacerlo un sólido, luego usar la repetición mediante una matriz

activando la opción de "fundir", y finalmente añadir un prisma debajo que haga de soporte de los dientes. Luego se engranan ambos componentes.

1.3.3. Herramientas para el cálculo de las propiedades de los engranajes

MITCalc

Es un conjunto de cálculos técnicos, ingenieriles e industriales para las rutinas diarias. Es preciso, confiable y guía a través del diseño de componentes, la solución a un problema técnico, o un cálculo de un punto de la ingeniería, sin necesidad de poseer conocimiento experto (MITCalc, 2017).

MITCalc contiene tanto el diseño como la verificación de los cálculos de muchas tareas comunes, tales como: engranajes rectos, engranaje helicoidal, engranaje piñón cremallera, resortes, viga, tolerancias, análisis de la tolerancia, fórmulas técnicas y muchos otros. Es un sistema abierto diseñado en Microsoft Excel a la que se le pueden realizar futuras modificaciones o extensiones de usuario sin tener habilidades de programación. Esta herramienta permite el intercambio de datos entre muchos sistemas comerciales 2D y 3D, por ejemplo, AutoCAD, AutoCAD LT, IntelliCAD, Autodesk Inventor, SolidWorks, SolidEdge, Pro/Engineer, entre otros (ibíd.).

eFunda

Es un sitio web que responde a una empresa con el mismo nombre y tiene como objetivo crear un destino en línea para la comunidad de ingeniería, donde pueda encontrarse ayuda en la solución de problemas complejos de diseño de forma concisa y confiable. Abarca materiales de nivel universitario de las escuelas de ingeniería, además dice exactamente bajo qué condiciones son utilizadas las fórmulas que posee (eFunda, 2017).

Esta herramienta realiza los cálculos de los parámetros físicos y geométricos que pueden tener los engranajes piñón cremallera de una manera rápida, pero no exporta el modelo calculado.

1.4. Metodología para el desarrollo de *software*

La metodología para el desarrollo de *software* es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de *software* desde que surge la necesidad del mismo hasta que se cumple el objetivo por el cual fue creado (Edeki, 2017).

El Proceso Unificado Ágil (AUP, por sus siglas en inglés) es un enfoque de modelado híbrido creado por Scott Ambler cuando combinó Proceso Unificado Racional (RUP, por sus siglas en inglés) con los métodos ágiles (AM, por sus siglas en inglés). Mediante la combinación de RUP con AM, Ambler creó un marco sólido de procesos que se puede aplicar a todo tipo de proyectos de *software*, grandes o pequeños (ibíd.).

Los principios que sustentan AUP son (ibíd.):

- 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.

Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decidió hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la Universidad de las Ciencias Informáticas. Con esta adaptación de AUP se logra estandarizar el proceso de desarrollo de *software* permitiendo que se adapte el ciclo de vida definido para la actividad productiva de la institución.

En la tabla 1.4 se explica de las 4 fases que define originalmente AUP, la variación de AUP-UCI (Rodríguez Sánchez, 2015):

Tabla 1.4. Fases AUP.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante la fase 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 la fase s ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obteniendo los requisitos y el diseño, se implementa y se libera el producto
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Para modelar el sistema en los proyectos productivos AUP-UCI define cuatro escenarios (ibíd.):

- **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.

Como guía para el desarrollo del trabajo de diploma será utilizada la metodología AUP en su variante UCI; para la presente investigación se seleccionó el escenario 4 debido a que posee un negocio muy bien definido, además el cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y se cuenta con poco tiempo para su realización.

1.5. Herramientas y lenguajes para el desarrollo

El desarrollo continuo de las TICs propicia el perfeccionamiento de las herramientas informáticas, de ahí que se hace necesario realizar una investigación acerca de las posibles herramientas a utilizar para el desarrollo del componente. A continuación, se describen las tecnologías y herramientas empleadas en la presente investigación, realizando un análisis de las principales características de cada una de ellas.

1.5.1. Open CASCADE Technology

Open CASCADE Technology (OCCT, por sus siglas en inglés) es una biblioteca de clases orientada a objetos desarrollada en C++ y diseñada para la producción de sofisticadas aplicaciones CAD/Fabricación Asistida por Computadora (CAM, por sus siglas en inglés)/Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés). Provee servicios para superficies 3D y modelado de sólidos, intercambio de datos CAD y visualización (OpenCascadeTechnology, 2017).

OCCT es una tecnología *software* libre, puede ser redistribuida y/o modificada bajo los términos de la Licencia Pública General Reducida (LGPL, por sus siglas en inglés) versión 2.1, con excepciones adicionales. Está diseñada para ser altamente portátil y es conocida por trabajar en una amplia gama de plataformas (UNIX, Linux, Windows, Mac OS X, Android) (ibíd.).

Open CASCADE cuenta con una comunidad que ha desarrollado Opencascade Community Edition (OCE, por sus siglas en inglés), la cual es reconocida y aceptada por OPEN CASCADE Company. OCE es una versión de OCCT desarrollada por las experiencias de la comunidad a través de las recomendaciones de optimización de las versiones liberadas mediante foros o en la propia página principal de desarrollo de esta tecnología (OpenCascadeCommunityEdition, 2017).

Se decide el empleo de OCE por ser una tecnología compatible con C++ y QT y además tener funcionalidades para el trabajo con OpenGL de forma nativa y brindar funcionalidades para la descomposición de objetos en sus componentes primarios (puntos, vértices y aristas).

1.5.2. Entorno de desarrollo integrado

Qt es un framework de desarrollo para aplicaciones multiplataforma que simplifica mucho la implementación de aplicaciones en C++ de forma nativa, también puede ser utilizado en otros lenguajes y tiene un

amplio apoyo. Es una biblioteca para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de consola y servidores (Qt, 2015).

Provee módulos para el desarrollo en áreas como redes, bases de datos, OpenGL, tecnologías web, sensores, protocolos de comunicación, procesamiento de XML y JSON, impresión, generación de PDF, entre otros (ibíd.).

Algunas características que posee el framework Qt son (ibíd.):

- Constituye además una biblioteca para la creación de interfaces gráficas. Se distribuye bajo una licencia libre: Licencia Pública General (GPL, por sus siglas en inglés), además de la LGPL, que permite su utilización gratuita con fines comerciales.
- Se encuentra disponible para un gran número de plataformas: Linux, MacOs X, Solaris, HP-UX, UNIX con X11, Windows.
- Possee Compatibilidad multiplataforma con un sólo código fuente.
- Fácil de internacionalizar.
- Arquitectura lista para plugins.

Se escoge como framework de desarrollo debido a todas las ventajas antes mencionadas, además es el utilizado en el FreeCAD que es la herramienta de diseño mecánico de mayor desarrollo dentro del *software* libre y que constituye la guía para el desarrollo del *software* desarrollado por el grupo SIPII.

1.5.3. Lenguaje de programación

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C, abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos por lo que se considera un lenguaje híbrido multiparadigma (Engineering, 2013).

C++ es un lenguaje de programación maduro y de gran velocidad de compilación y presenta las siguientes características (Stroustrup, 1997):

- Lenguaje versátil, potente y general.
- Lenguaje rico en operadores y expresiones.
- Flexible, conciso y eficiente.
- Presenta programación modular.
- Introduce nuevas palabras claves y operadores para manejo de clases.

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Además, se trata de un lenguaje de programación estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar C++ que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo (Engineering, 2013).

Posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel (Stroustrup, 1997):

- Posibilidad de redefinir los operadores (sobrecarga de operadores).
- Identificación de tipos en tiempo de ejecución (RTTI).

Se escoge C++ como lenguaje de programación debido a las potencialidades expuestas anteriormente, además de que la biblioteca de visualización OCE está desarrollada sobre este lenguaje y el framework Qt define C++ como su lenguaje nativo.

1.5.4. Lenguaje de modelado

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) ofrece un estándar para describir un "plano"del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (VisualParadim, 2017).

1.5.5. Herramienta de modelado

Visual Paradigm para UML 8.0 es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos (ibíd.).

Visual Paradigm también ofrece (ibíd.):

- intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automáticamente de layout.
- Sincronización de código fuente en tiempo real.

1.5.6. Sistema de control de versiones

Un sistema de control de versiones es una herramienta capaz de registrar todos los cambios hechos en el contenido de un proyecto, guardando versiones del producto en todas sus fases del desarrollo.

GitLab es una aplicación individual construida desde cero para equipos de producto, desarrollo, seguridad y operaciones y permite trabajar simultáneamente en el mismo proyecto. Ofrece la posibilidad a los equipos de colaborar y trabajar desde una única conversación, en lugar de administrar múltiples hilos entre herramientas dispares. GitLab proporciona a los equipos un único almacén de datos, una interfaz de usuario y un modelo de permiso en todo el ciclo de vida de DevOps, lo que permite a los equipos colaborar, reduciendo significativamente el tiempo de ciclo y centrándose exclusivamente en crear *software* de gran calidad rápidamente (GIitLab, 2018).

1.6. Conclusiones del capítulo

Al finalizar este capítulo se pudo arribar a la conclusión que el estudio de las herramientas similares permitió identificar que algunas herramientas comerciales, como Solid Edge, poseen módulos independientes dedicados a acelerar el proceso de diseño del engranaje piñón cremallera, sin embargo, las herramientas libres ni los contienen ni brindan los cálculos de sus propiedades. Se identificaron las funcionalidades requeridas para algoritmizar el proceso de modelado del engranaje piñón cremallera en correspondencia de las herramientas comerciales. Finalmente, las herramientas, lenguaje, metodología y tecnologías definidas en el ambiente de desarrollo permitieron definir el perfil tecnológico de la investigación.

CAPÍTULO 2

Propuesta de solución

En el presente capítulo se aborda la propuesta de solución e identifican y describen los requisitos funcionales y no funcionales. Se presenta el estilo y patrón arquitectónico, los patrones del diseño empleados, el diagrama de clases y las historias de usuario; además, se aborda la metodología de cálculo a emplear.

2.1. Modelo conceptual

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de *software*, constituye el artefacto más importante que se crea durante el análisis orientado a objetos. Mediante la notación del UML, se representa con un conjunto de diagramas de clases en los que no se define ninguna operación, en el mismo se muestran objetos del dominio o clases conceptuales, asociaciones entre las clases conceptuales y atributos de las clases conceptuales (Barzaga Pérez, 2011).

Por exigencias del proyecto se estimó necesario generar este artefacto para un mejor entendimiento de la propuesta de solución a desarrollar. El mismo se puede ilustrar en la figura 2.1.

En el diagrama 2.1 se 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.

2.2. Descripción del componente

El componente elaborado permite el modelado del engranaje piñón cremallera de dientes rectos o helicoidales con perfil de involuta para ser insertado en la aplicación desarrollada por el grupo de investigación SIPII; el mismo cuenta con varias características que serán explicadas en los párrafos que siguen.

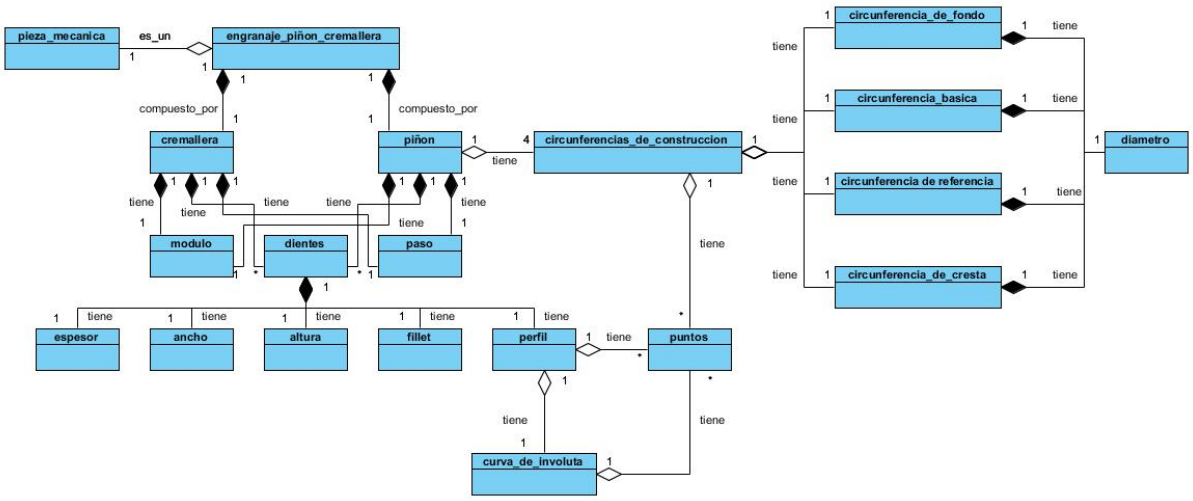


Figura 2.1. Diagrama de modelo del negocio.

El componente muestra una interfaz que posee una pestaña en la parte superior llamada ”Modeling”. En dicha pestaña se encuentran los datos y parámetros relativos al diseño del engranaje; 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 grados), el módulo (con un rango de datos entre 0,050 mm hasta 100 mm) y le brinda la opción al usuario para escoger entre el caso que desee diseñar un engranaje piñón cremallera o solo una cremallera y el sentido que tendrán los dientes de la cremallera (izquierdo o derecho), siendo los dientes del piñón orientados en sentido opuesto a los de la cremallera. Posee campos para introducir los valores deseados como el ángulo de la hélice del engranaje, el número de dientes del piñón y de la cremallera y las correcciones de unidades del piñón. Por otra parte, a la derecha de la interfaz se muestran, en una tabla, los datos del cálculo de las dimensiones del engranaje.

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 tabla ubicada en un 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 de acción *Accept*, el cual vuelve a revisar los datos introducidos y en caso de existir un error se ocultará y mostrará un mensaje, en el área de notificaciones, alertando al usuario que hay datos erróneos y debe volver a presionar el botón *Calculate* para encontrarlos; en cambio, si no hay ningún error cierra la interfaz de construcción de engranajes y muestra, en tres dimensiones, el diseño del mismo;

2.3. Requisitos

Los requisitos para un sistema son la descripción de los servicios proporcionados por el mismo 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 (Pressman, 2005).

Los requisitos funcionales permiten conocer detalladamente las características y funciones del sistema. Los requisitos no funcionales dan a conocer las cualidades que deberá cumplir el sistema.

2.3.1. Requisitos funcionales

A continuación, se describen los aspectos funcionales que deberá cumplir el componente.

RF 1. Especificar los parámetros comunes del engranaje.
RF 2. Introducir los parámetros de la cremallera.
RF 3. Introducir los parámetros del piñón.
RF 4. Visualizar la cremallera de dientes rectos en tres dimensiones.
RF 5. Visualizar la cremallera de dientes helicoidales en tres dimensiones.
RF 6. Visualizar el engranaje piñón cremallera de dientes rectos.
RF 7. Visualizar el engranaje piñón cremallera de dientes helicoidales.
RF 8. Visualizar los resultados del calculo de los engranajes.

2.3.2. Requisitos no funcionales

A continuación, se describen los aspectos no funcionales que deberá cumplir el componente:

- **Software:**

RNF 1: Se debe poder ejecutar en un sistema operativo basado en GNU-Linux de 64bits.

- **Restricciones del diseño:**

RNF 2: Lenguaje C++, framework de desarrollo QT, bibliotecas de OCE comunitaria y arquitectura en capas.

2.4. Historias de usuarios

Las historias de usuario son tarjetas de papel en las cuales el cliente describe brevemente las características funcionales o no funcionales que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento pueden eliminarse o reemplazarse por otras más específicas,

añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en poco tiempo (Jeffries, Anderson y C., 2001).

En la tabla 2.1 se muestra una historia de usuario realizada a la propuesta de solución. Para el estudio de las demás historias de usuario remitirse al Anexo A.

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Introducir los parámetros comunes del engranaje piñón cremallera.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 0.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 0.5 semanas

Descripción:1. Objetivo

- Permitir la inserción de los parámetros comunes del piñón y la cremallera.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para insertar los parámetros comunes del engranaje piñón cremallera hay que tener en cuenta los siguientes datos: módulo, ángulo de presión, ángulo de la hélice, dirección de la hélice, tipo de engranaje que se desea modelar.

3. Comportamiento válido y no válido

- El módulo es una lista desplegable.
- El ángulo de presión es una lista desplegable.
- El ángulo de la hélice debe ser un valor numérico real y positivo.
- La dirección de la hélice se conforma de dos *radio button*, uno con el nombre *Helix Left* y otro *Helix Rigth*. Por defecto aparece marcado *Helix Left*.
- El tipo de engranaje se conforma de dos radio button, uno con el nombre *Pinion Rack Modeling* y el otro *Rack Modeling*. Por defecto aparece marcado *Pinion Rack Modeling*.

4. Flujo de acción

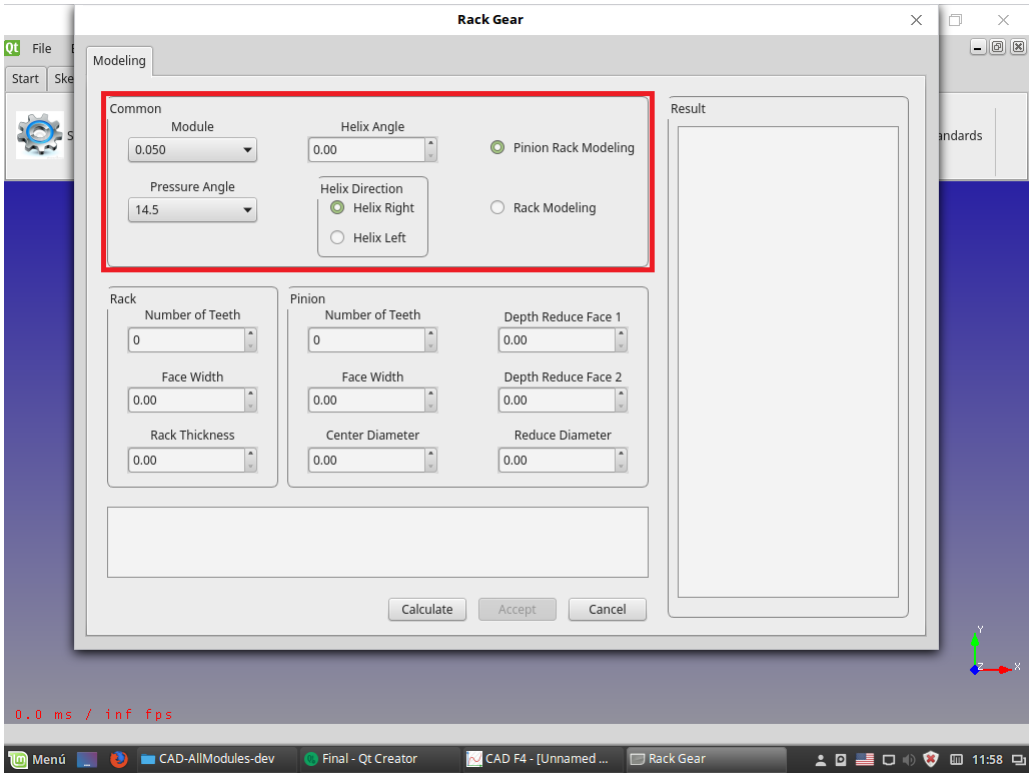
- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz.
- Si los datos están incorrectos se lanza un mensaje de error en letras rojas en la parte inferior de la interfaz que deberá acercar al usuario al error de construcción, dándole a la oportunidad nuevamente de realizar la acción en cuestión.
- Si se selecciona el botón *Cance* se cancela la acción y se cierra la ventana.

Observaciones:

Continuación de la página anterior

Tabla 2.1. Continuación de la página anterior

Interfaz:



2.5. Diseño

El diseño está definido, según la Institute of Electrical and Electronics Engineers (IEEE, por sus siglas en inglés), de dos formas: “el proceso para definir la arquitectura, los componentes, las interfaces y otras características del sistema” y “el resultado del proceso (mencionado)”. Teniendo en cuenta la segunda definición, el resultado del diseño debe describir la arquitectura utilizada, que representa como el *software* se descompone y organiza; además de las interfaces y la descripción detallada de los componentes que posibiliten su posterior desarrollo (Computer Society, 2004).

2.5.1. Estilo y patrón arquitectónico del *software*

Según Pressman “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, 2005).

En este sentido 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. Este estilo posee como patrones arquitectónicos a las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los que son orientados a objeto y los jerárquicos en capas (Pressman, 2005).

2.5.2. Arquitectura en dos capas:

El modelo en capas organiza el sistema en capas, cada una de las cuales proporciona un conjunto de servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Este modelo soporta el desarrollo incremental del sistema, los cambios y es portable. Cuando las interfaces de las capas cambian o se añaden nuevas funcionalidades a una capa, solo se ven afectadas las capas adyacentes.

La arquitectura del componente se organiza en dos capas: la capa "Vista" contendrá la clase interfaz con las que el usuario interactúa y la capa "Modelo" contendrá las clases que permiten la creación de los objetos. Para un mayor entendimiento de la distribución de las clases dentro de las capas ver figura 2.3 Diagrama de clases en el siguiente apartado.

En la figura 2.2 podemos observar la arquitectura del módulo.

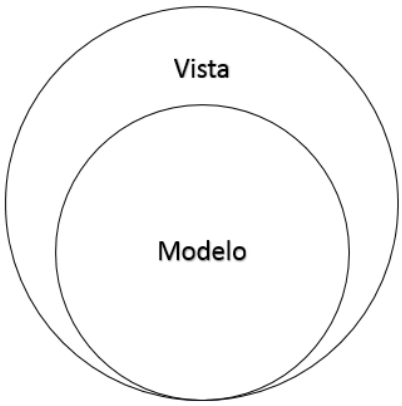


Figura 2.2. Arquitectura del módulo.

2.5.3. Diagrama de clases del componente

Un diagrama o modelo de clases en UML es un tipo de diagrama de estructura estática que describe gráficamente las especificaciones de las clases de *software*, sus atributos, operaciones (o métodos) y de las relaciones entre las interfaces en una aplicación (Larman, 1999).

El diagrama de clases del componente se puede ilustrar en la figura 2.3.

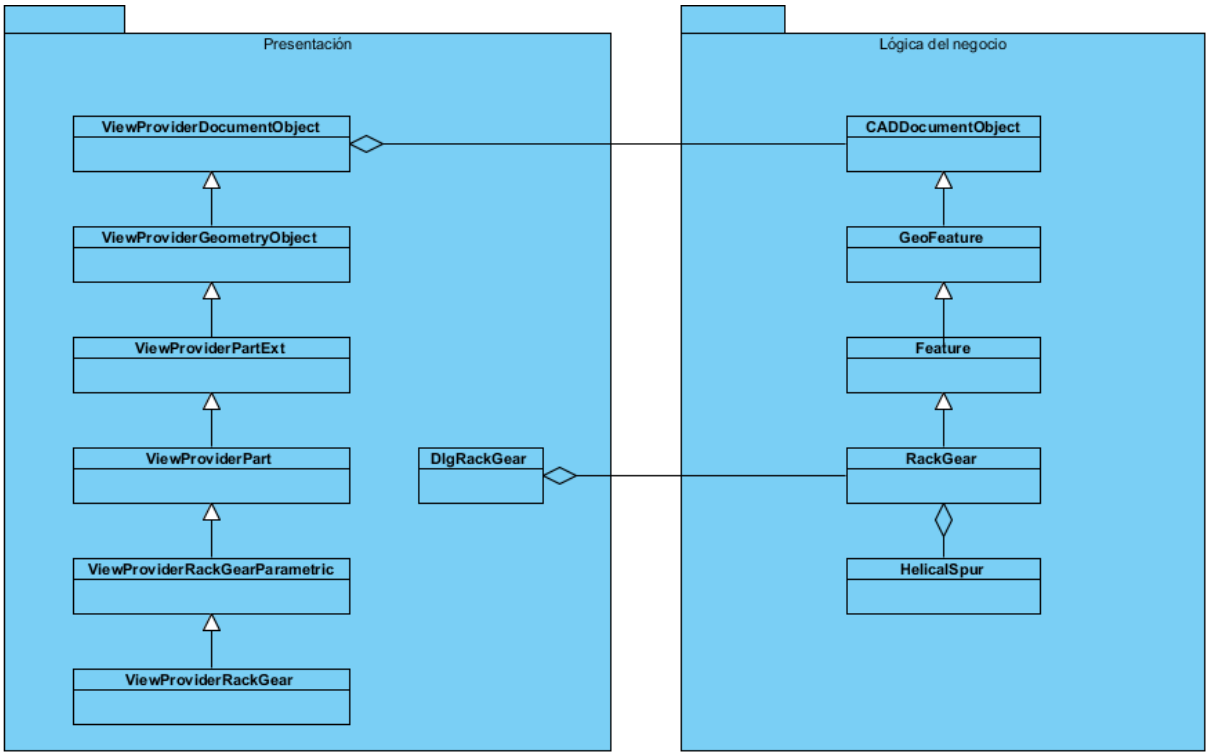


Figura 2.3. Diagrama de clases.

2.5.4. Patrones de diseño

Según Craig Larman "un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas"(Larman, 1999).

Los Patrones Generales de Software Para Asignar Responsabilidades (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (ibíd.).

Los patrones GRASP definidos en el modelo de clases del componente fueron:

- **Experto:** la responsabilidad de la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para hacerlo(ibíd.). Este patrón se evidencia en las clases *RackGear* y *HelicalSpur* que se encargan de la creación de la cremallera y el piñón respectivamente.
- **Creador:** ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos (ibíd.). Se evidencia en la clase *DlgRackGear* la cual tiene la información necesaria para crear el objeto de tipo piñón cremallera.
- **Alta cohesión:** es una medida de cuan relacionadas o enfocadas están las responsabilidades de una clase (ibíd.). Se evidencia en la relación entre las clases *RackGear* y *HelicalSpur*.
- **Bajo acoplamiento:** estimula asignar una responsabilidad de modo reduzca el impacto de los cambios y que no incremente el acoplamiento que es una medida de la fuerza en que las clases se relacionan

(Larman, 1999). Se evidencia en las clases *RackGear* y *HelicalSpur*.

The Gang of Four (GOF, por sus siglas en inglés) son aquellos patrones que describen soluciones simples y elegantes a problemas específicos en el diseño de *software* orientado a objetos (Guerrero, Suárez y Gutiérrez, 2009).

El patrón GOF definido en el modelo de clases del componente fue:

- **Observador:** Este patrón se evidencia con la función *mustExecute* el cual permite que las propiedades de las piezas modeladas, puedan ser editadas.

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 permiten el correcto desarrollo de las funcionalidades. Quedó definido el modelo arquitectónico y las clases que conforman las diferentes capas y las relaciones entre ellas. Finalmente, los artefactos obtenidos en este capítulo sientan las bases para la implementación de la solución propuesta.

CAPÍTULO 3

Implementación y pruebas

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

3.1. Implementación

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

3.1.1. Estándares de codificación

Los estándares de codificación se utilizan durante la fase de implementación de un *software* para darle una estructura al código facilitando su comprensión y mantenimiento (Fernández Ferrer, 2015).

El estándar de codificación utilizado para desarrollar el componente se describe en las figuras 3.1, 3.2 y 3.3 respectivamente.

Descripción	Ejemplo
Definición de Objetos, Clases, funciones y atributos	

Todos los nombres de las clases implementadas comenzarán con letra mayúscula. En caso de poseer un nombre compuesto se escribirán de acuerdo a la normativa CamelCase-UpperCamelCase.	class Foo{
	cuerpo de la clase
	}
	class FooFirst{
	cuerpo de la clase
	}

Siempre se declara para todas las clases implementadas su respectivo destructor de clase.	virtual ~Foo()
La declaración de funciones o métodos siempre comenzarán en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato retorno> funcion() <Tipo dato retorno> funcionCompuesta() <Tipo dato retorno>funcionDobleCompuesta()

Figura 3.1. Estándar de codificación parte 1.

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[];

Figura 3.2. Estándar de codificación parte 2.

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 3.3. Estándar de codificación parte 3.

3.1.2. Detalles técnicos de la implementación

Objetos y funciones empleadas de Open CASCADE:

- **gp_Ax2d:** Crea un eje representando en el eje X un objeto de referencia en el sistema de coordenadas.
- **gp_Circ2d:** Crea un circulo en dos dimensiones en el plano. Este circulo esta definido por un radio y una posición en el plano.
- **gp_Trsf:** Función que brinda la posibilidad de realizar transformaciones en el espacio tridimensional como traslación o rotación.
- **gp_Pnt:** Crea un punto en tres dimensiones.
- **gp_Pnt2d:** Crea un punto en dos dimensiones.
- **GCE2d_MakeSegment:** Crea un segmento entre dos puntos en dos dimensiones.
- **GCE2d_MakeCircle:** Crea un círculo en el espacio a partir de uno en el plano.
- **TopoDS_Wire:** Topología presente en las biblioteca Open CASCADE.
- **TopoDS_Face:** Topología presente en las biblioteca de Open CASCADE que tiene como objetivo crear un face de un *wire* cerrado.
- **TopoDS_Shape:** Topología presente en las biblioteca de Open CASCADE, la cual tiene como objetivo hacer un sólido.
- **TopoDS_Compound:** Función que permitir unir varios *shape* para poder trabajar con ellos.
- **SetDisplacement:** Función que modifica una transformación y permite crear un nuevo sistema de

coordenadas.

- **SetRotation:** Función que se utiliza dentro de una transformación para rotar
- **BrepLib:** Función que convierte una curva en 2D en una 3D.
- **BRepAlgoAPI-Cut:** Función que realiza una operación *boolean* para realizar un corte sobre dos sólidos.
- **BRepBuilderAPI_MakeWire:** Topologías de la biblioteca Open CASCADE.
- **BRepBuilderAPI_MakeEdge:** Topologías de la biblioteca Open CASCADE, la cual tiene como objetivo crear un borde.
- **BRepBuilderAPI_Transform:** Topologías de la biblioteca Open CASCADE, la cual tiene como objetivo construir un esquema para aplicar la geometría.
- **BRepFill_PipeShell:** Topologías de la biblioteca Open CASCADE, la cual tiene como objetivo realizar el *Swep*.

Construcción del engranaje piñón cremallera

Creando el piñón

Para iniciar la construcción del piñón es necesario representar cuatro 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 cremallera; se crea posicionando un punto como centro, en este caso el punto (0,0,0) y su diámetro se calcula por la expresión 3.1.1:

$$d = z_p * m \tag{3.1.1}$$

  Donde:

  - d : diámetro de referencia.
  - z_p : número de dientes del piñón.
  - m : módulo normal.

- **Circunferencia de cresta:** La circunferencia de cresta es la encargada de señalar el valor de altura al diente cortando a su perfil y en algunos casos es tangente a la circunferencia básica de la cremallera; se crea posicionando un punto como centro, en este caso el punto (0,0,0) y su diámetro se calcula por la expresión 3.1.2:

$$d_a = d + 2 * h_a \tag{3.1.2}$$

  Donde:

  - d_a : diámetro de cresta.
  - h_a : *addendum*.

- **Circunferencia base:** La circunferencia básica sirve como base al trazado del perfil del diente y, en algunos casos, es tangente a la circunferencia de cresta de la cremallera; se crea posicionando un punto como centro, en este caso el punto (0,0,0) y su diámetro se calcula por la expresión 3.1.3:

$$d * \cos(\alpha) \tag{3.1.3}$$

  Donde:

  - d : diámetro de cresta.
  - α : ángulo de presión.

- **Circunferencia de Fondo:** La circunferencia de fondo es la encargada de representar el núcleo de la rueda; si el diámetro de la circunferencia de fondo es menor que el diámetro de la circunferencia básica, se traza una línea radial desde el inicio del perfil del diente en la circunferencia básica, hasta el centro de la circunferencia, para darle al diente el cuerpo bajo el perfil; por otro lado, si el diámetro de la circunferencia de fondo es mayor que el diámetro de la circunferencia básica, el perfil del diente se crea a partir de la circunferencia de fondo; se crea posicionando un punto como centro, en este caso el punto (0,0,0) y su diámetro se calcula por la expresión 3.1.4:

$$d_f = d_a - 2 * h \tag{3.1.4}$$

  Donde:

  - d_f : diámetro de fondo.
  - h : altura del diente.

Con las circunferencias de construcción obtenidas es posible crear el núcleo de la rueda, para el cual se utiliza la circunferencia de fondo; se obtiene de esta manera, un objeto de tipo *shape*. Con el núcleo finalizado se conforma el perfil de involuta en el plano XY. Para la creación de este se utiliza un perfil opuesto, que se representa visualmente como el espacio entre los dientes del piñón, el cual servirá para construir el perfil evolvente; para esto se utilizan las ecuaciones 3.1.5, 3.1.6, 3.1.7 (Michael, 2015):

$$inv\ \alpha = tan(\alpha) - \alpha \tag{3.1.5}$$

$$x = r * cos(inv\ \alpha) \tag{3.1.6}$$

$$y = r * sin(inv\ \alpha) \tag{3.1.7}$$

Donde:

- x : Ordenada x del punto de involuta para las condiciones dadas.
- y : Ordenada y del punto de involuta para las condiciones dadas.
- r : Radio de referencia.

- $inv\alpha$: Función de involuta

En donde α representa el ángulo que genera la tangente en cada caso. El ángulo se va incrementando hasta llegar al valor deseado. Cuando α >90 no es necesario seguir incrementando el ángulo. Luego de tener formados todos los puntos se unen para crear una línea que se replica para obtener su imagen y se une con la original mediante una línea en la parte inferior y un arco en la parte superior, formando así el perfil de involuta. Una vez obtenido este creamos el objeto de tipo *face* y de ahí se extruye para conformar el sólido. Este proceso se ilustra en la figura 3.4.

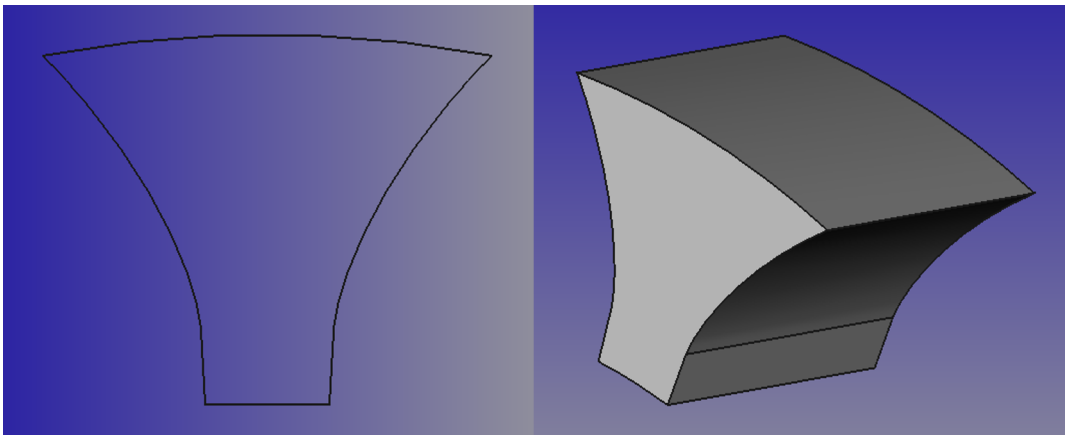


Figura 3.4. Secuencia de imágenes del perfil del diente.

Con el sólido del perfil creado se procede a unirlo con el núcleo incorporándolo en la circunferencia básica en caso de tener mayor diámetro que la circunferencia de fondo, de no ser así, el perfil de involuta se une a la circunferencia de fondo. Luego se procede a replicarlo según la cantidad de dientes del engranaje, rotándolo por el núcleo, para al final cortarlo con todos los perfiles, obteniendo de esta manera el piñón. Este proceso se ilustra el la figura 3.5.

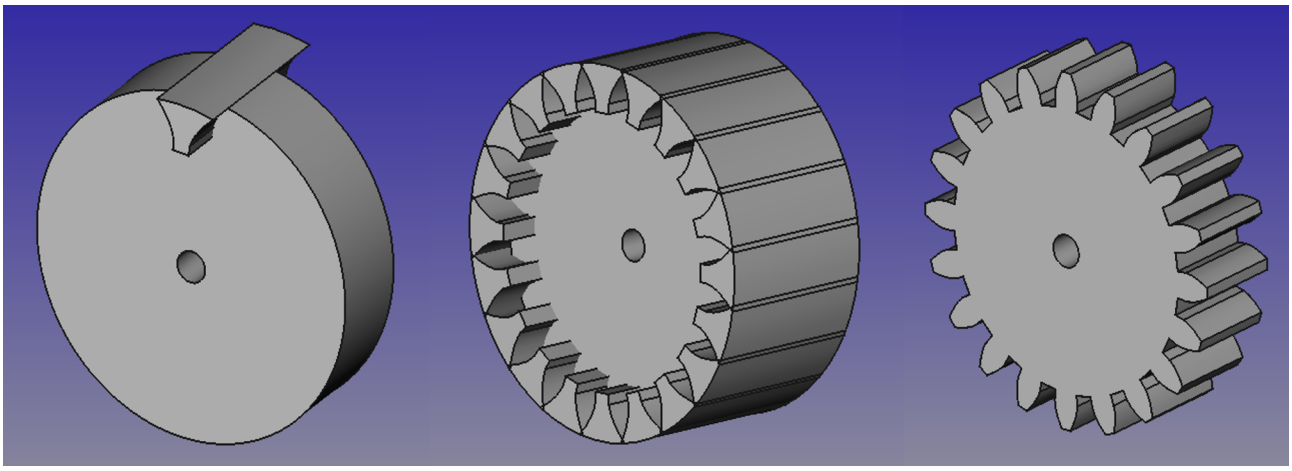


Figura 3.5. Secuencia de imágenes de conformación del piñón de dientes rectos.

Para crear los dientes helicoidales se crea una hélice cilíndrica en correspondencia con las ecuaciones del diámetro de fondo que atraviese el núcleo previamente creado. Posteriormente se calculan los puntos inicial y final de la misma y mediante la primera derivada se obtiene el vector tangente en el primer punto. Luego se realiza un desplazamiento del punto de referencia del perfil creado pasándole el vector tangente para tener entre la hélice tangente a dicho perfil. Se realiza un barrido que va desde el inicio hasta el final de la hélice y se replica el sólido del diente por todo el círculo y se aplica la función *cut*. Este proceso se ilustra el la figura 3.6.

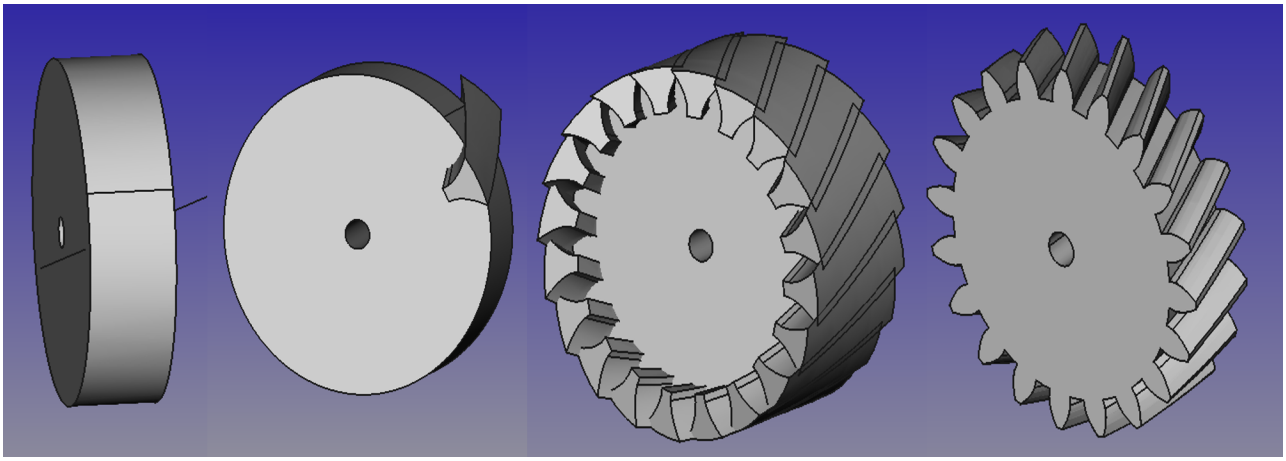


Figura 3.6. Secuencia de imágenes de conformación del piñón de dientes helicoidales.

Para crear la cremallera es necesario determinar la longitud y altura del diente mediante la suma del *addendum* más el *dedendum*, altura de la cremallera, el ancho del fondo del diente y el paso mediante las ecuaciones 3.1.8 a la 3.1.13 (Michael, 2015):

$$l = z_r \frac{*}{p} cos(\beta) \tag{3.1.8}$$

$$h = h_a + h_d = r * sin(inv\ \alpha) \tag{3.1.9}$$

$$h_a = m \tag{3.1.10}$$

$$H_r = r_t + h \tag{3.1.11}$$

$$h_d = 1.25 * m \tag{3.1.12}$$

$$s = \frac{p}{2} + 2 * h_d * tan(\alpha) \tag{3.1.13}$$

Donde:

- l : longitud de la cremallera.
- z_r : número de dientes de la cremallera.
- p : paso.

- β : ángulo de la hélice.
- h_d : *dedendum*.
- s : ancho del fondo del diente.
- H_r : altura de la cremallera.
- r_t : altura hasta el inicio del diente.

Con la longitud y altura de la cremallera se procede a ubicar los puntos de construcción para formar un rectángulo, luego se unen con varias líneas para formar un objeto de tipo *wire*; seguidamente se crea el objeto de tipo *face* para luego extruirlo. A continuación se procede a construir el objeto de corte con el cual se obtendrá la cremallera. Para conformar este objeto se le asigna un largo mayor al de la cremallera, se ubican los puntos de la forma que se obtenga una cremallera opuesta a la inicial incorporando en los extremos la mitad del perfil de un diente con una extensión mayor al ancho del mismo. Se unen estos puntos con una línea para formar un objeto de tipo *wire*, se crea el objeto de tipo *face* y se extruye para formar el sólido del objeto. En la figura 3.7 se ilustra este proceso.

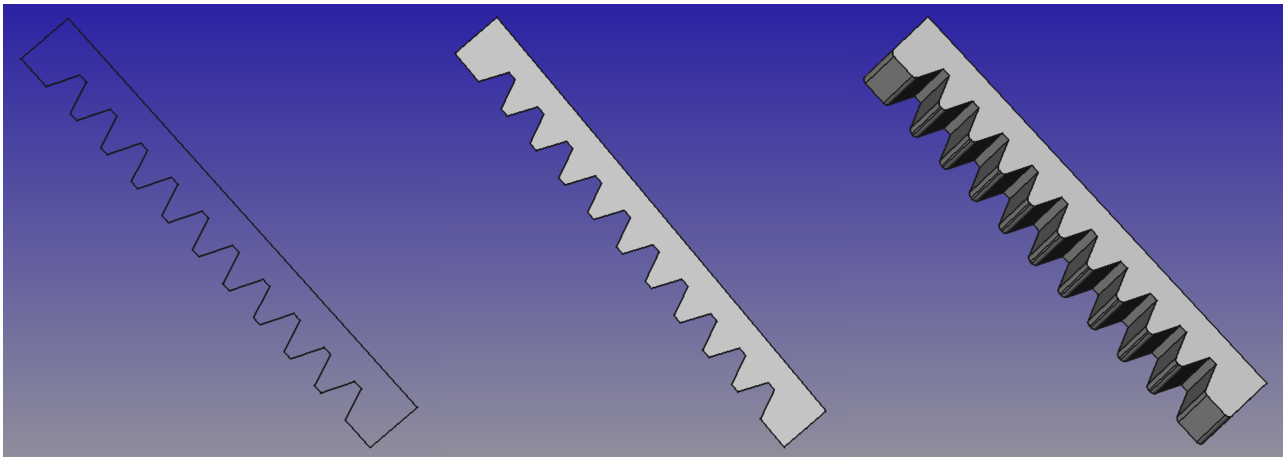


Figura 3.7. Secuencia de imágenes de la objeto de corte de la cremallera.

Para crear la cremallera se ubica la herramienta de corte encima del rectángulo, de forma tal que la línea que limita los dientes de la herramienta con el rectángulo de esta quede en el mismo lugar que la línea superior del rectángulo. Se procede, entonces, a cortar el rectángulo con la herramienta de corte obteniendo una cremallera de dientes rectos tal y como se muestra en la figura 3.8.

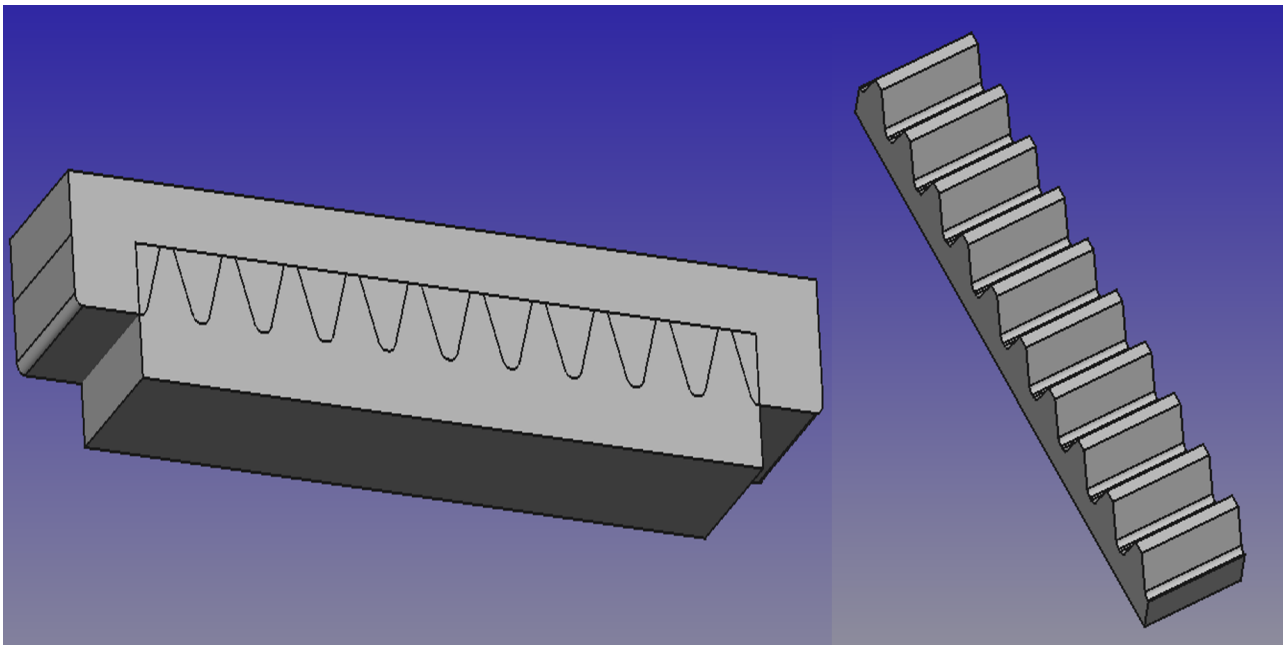


Figura 3.8. Secuencia de imágenes de conformación de la cremallera de dientes rectos.

Para obtener una cremallera de dientes helicoidales se le proporciona un mayor largo a la longitud del objeto de corte, se rota el objeto de tipo *face* de la herramienta en un plano inclinado con respecto al de construcción con el valor del ángulo de la hélice y se extruye el objeto con un valor tres veces superior al del rectángulo obteniendo una cremallera de dientes helicoidales como la de la figura 3.9.

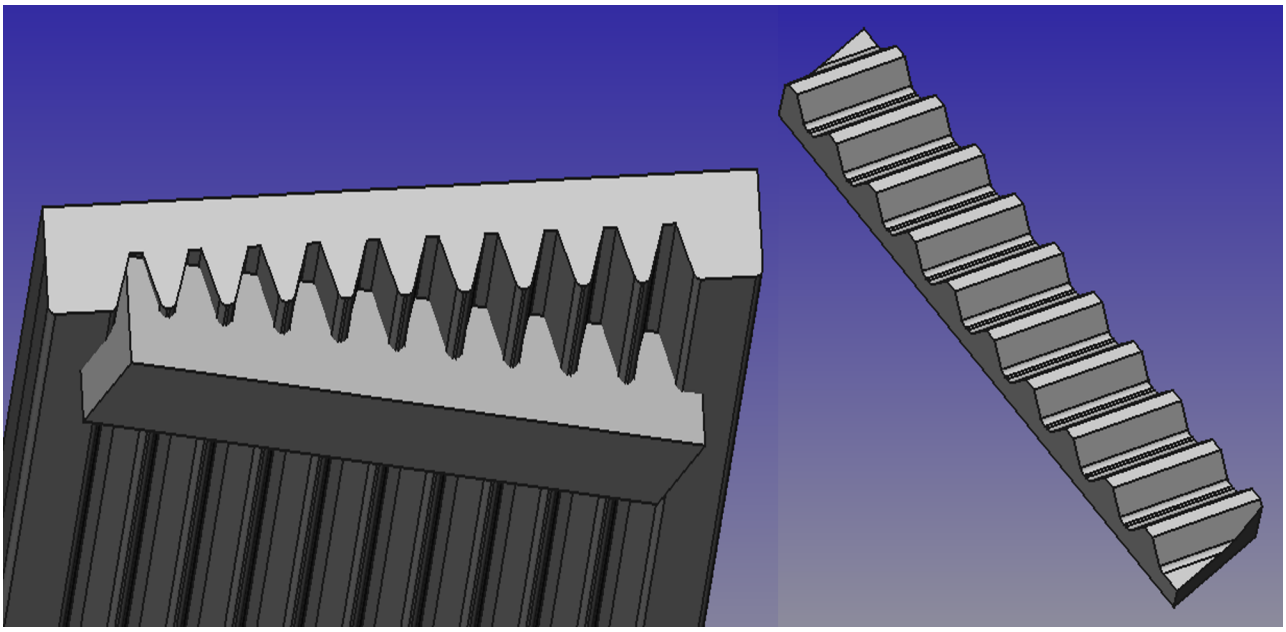


Figura 3.9. Secuencia de imágenes de conformación de la cremallera de dientes helicoidales.

El módulo implementado contiene 5 clases y 16 funciones para un total de 1304 líneas de código. A continuación, se describen las clases y funcionalidades implementadas para una mejor comprensión de la solución.

La clase **DlgRackGear:** implementa todo lo relacionado a la interfaz. Recibe los datos que introduce el usuario, verifica que estén correctos, calcula los datos fundamentales para la construcción del engranaje y los muestra en una tabla. Dicha clase contiene las funciones:

- **on_calcular_clicked:** Llama al método *excecuteCalculation* y si no hay errores en los datos introducidos llama a la función *showCalculation* para mostrar los resultados de los cálculos en la tabla de la interfaz.
- **on_aceptar_clicked:** Realiza la acción del botón *Accept* cerrando la ventana principal y mostrando el engranaje diseñado, al pulsarlo, cuando los datos están correctos. Si los datos no están correctos envía un mensaje de error en la ventana de notificaciones.
- **on_cancelarDesing_clicked:** Realiza la acción del botón *Cancel* cerrando la ventana principal al pulsarlo.
- **excecuteCalculation:** Ejecuta los cálculos de las propiedades de los engranajes y es el encargado de mostrar un mensaje en la ventana de notificaciones acercando al usuario al error cometido en la inserción de los datos o informando que los datos introducidos son correctos.
- **showCalculation:** Muestra los resultados de los cálculos en la tabla de la interfaz principal.
- **on_radioButtonRack_clicked:** Oculta los campos para introducir los datos del piñón.
- **on_radioButtonPinionRack_clicked:** Muestra los campos para introducir los datos del piñón en caso que estén ocultos.

La clase **HelicalSpur** se encarga de construir el piñón con dientes rectos o helicoidales. Esta clase contiene las siguientes funciones:

- **loadHelicalSpur:** Recibe la cremallera de la función *makeHelicalSpur* y la envía a *loadRackGear*.
- **makeHelicalSpur:** Utilizando los valores que recibe por parámetro construye el piñón de dientes rectos o helicoidales.
- **makeCentralHole:** Devuelve un sólido con el agujero central en forma de círculo de un sólido pasado por parámetros.
- **pointIntercectionC1C2:** Realiza la intersección de dos curvas en el espacio.

La clase **RackGear** es la principal e interactúa directamente con la interfaz y provee a la clase HelicalSpur la información necesaria para la construcción del piñón. La misma contiene las funciones:

- **loadRackGear:** Recibe el piñón de la función *loadHelicalSpur*, lo engrana con la cremallera que recibe de la función *makeRack* y visualiza el engrane en el visor.
- **loadRackGearOnly:** Recibe la cremallera de la función *makeRack* y la visualiza en el visor.

- **makeRack:** Utilizando los valores que recibe por parámetro construye la cremallera de dientes rectos o helicoidales.
- **mustExecute:** Clase que hereda de *Feature* y es la responsable de dejar al usuario editar los valores de la geometría y ver la edición en el visor.
- **execute:** Clase que hereda de *Feature* y es la responsable de visualizar el engranaje en el visor.

El módulo implementado contiene 5 clases y 16 funciones para un total de 1304 líneas de código. A continuación, se describen las clases y funcionalidades implementadas para una mejor comprensión de la solución.

La clase **DlgRackGear:** implementa todo lo relacionado a la interfaz. Recibe los datos que introduce el usuario, verifica que estén correctos, calcula los datos fundamentales para la construcción del engranaje y los muestra en una tabla. Dicha clase contiene las funciones:

- **on_calcular_clicked:** Llama al método *excecuteCalculation* y si no hay errores en los datos introducidos llama a la función *showCalculation* para mostrar los resultados de los cálculos en la tabla de la interfaz.
- **on_aceptar_clicked:** Realiza la acción del botón *Accept* cerrando la ventana principal y mostrando el engranaje diseñado, al pulsarlo, cuando los datos están correctos. Si los datos no están correctos envía un mensaje de error en la ventana de notificaciones.
- **on_cancelarDesing_clicked:** Realiza la acción del botón *Cancel* cerrando la ventana principal al pulsarlo.
- **excecuteCalculation:** Ejecuta los cálculos de las propiedades de los engranajes y es el encargado de mostrar un mensaje en la ventana de notificaciones acercando al usuario al error cometido en la inserción de los datos o informando que los datos introducidos son correctos.
- **showCalculation:** Muestra los resultados de los cálculos en la tabla de la interfaz principal.
- **on_radioButtonRack_clicked:** Oculta los campos para introducir los datos del piñón.
- **on_radioButtonPinionRack_clicked:** Muestra los campos para introducir los datos del piñón en caso que estén ocultos.

La clase **HelicalSpur** se encarga de construir el piñón con dientes rectos o helicoidales. Esta clase contiene las siguientes funciones:

- **loadHelicalSpur:** Recibe la cremallera de la función *makeHelicalSpur* y la envía a *loadRackGear*.
- **makeHelicalSpur:** Utilizando los valores que recibe por parámetro construye el piñón de dientes rectos o helicoidales.
- **makeCentralHole:** Devuelve un sólido con el agujero central en forma de círculo de un sólido pasado por parámetros.
- **pointIntercectionC1C2:** Realiza la intersección de dos curvas en el espacio.

La clase **RackGear** es la principal e interactúa directamente con la interfaz y provee a la clase HelicalSpur la información necesaria para la construcción del piñón. La misma contiene las funciones:

- **loadRackGear:** Recibe el piñón de la función *loadHelicalSpur*, lo engrana con la cremallera que recibe de la función *makeRack* y visualiza el engrane en el visor.
- **loadRackGearOnly:** Recibe la cremallera de la función *makeRack* y la visualiza en el visor.
- **makeRack:** Utilizando los valores que recibe por parámetro construye la cremallera de dientes rectos o helicoidales.
- **mustExecute:** Clase que hereda de *Feature* y es la responsable de dejar al usuario editar los valores de la geometría y ver la edición en el visor.
- **execute:** Clase que hereda de *Feature* y es la responsable de visualizar el engranaje en el visor.

El módulo implementado contiene 5 clases y 16 funciones para un total de 1304 líneas de código. A continuación, se describen las clases y funcionalidades implementadas para una mejor comprensión de la solución.

La clase **DlgRackGear:** implementa todo lo relacionado a la interfaz. Recibe los datos que introduce el usuario, verifica que estén correctos, calcula los datos fundamentales para la construcción del engranaje y los muestra en una tabla. Dicha clase contiene las funciones:

- **on_calcular_clicked:** Llama al método *excecuteCalculation* y si no hay errores en los datos introducidos llama a la función *showCalculation* para mostrar los resultados de los cálculos en la tabla de la interfaz.
- **on_aceptar_clicked:** Realiza la acción del botón *Accept* cerrando la ventana principal y mostrando el engranaje diseñado, al pulsarlo, cuando los datos están correctos. Si los datos no están correctos envía un mensaje de error en la ventana de notificaciones.
- **on_cancelarDesing_clicked:** Realiza la acción del botón *Cancel* cerrando la ventana principal al pulsarlo.
- **excecuteCalculation:** Ejecuta los cálculos de las propiedades de los engranajes y es el encargado de mostrar un mensaje en la ventana de notificaciones acercando al usuario al error cometido en la inserción de los datos o informando que los datos introducidos son correctos.
- **showCalculation:** Muestra los resultados de los cálculos en la tabla de la interfaz principal.
- **on_radioButtonRack_clicked:** Oculta los campos para introducir los datos del piñón.
- **on_radioButtonPinionRack_clicked:** Muestra los campos para introducir los datos del piñón en caso que estén ocultos.

La clase **HelicalSpur** se encarga de construir el piñón con dientes rectos o helicoidales. Esta clase contiene las siguientes funciones:

- **loadHelicalSpur:** Recibe la cremallera de la función *makeHelicalSpur* y la envía a *loadRackGear*.

- **makeHelicalSpur:** Utilizando los valores que recibe por parámetro construye el piñón de dientes rectos o helicoidales.
- **makeCentralHole:** Devuelve un sólido con el agujero central en forma de círculo de un sólido pasado por parámetros.
- **pointIntercectionC1C2:** Realiza la intersección de dos curvas en el espacio.

La clase **RackGear** es la principal e interactúa directamente con la interfaz y provee a la clase HelicalSpur la información necesaria para la construcción del piñón. La misma contiene las funciones:

- **loadRackGear:** Recibe el piñón de la función *loadHelicalSpur*, lo engrana con la cremallera que recibe de la función *makeRack* y visualiza el engrane en el visor.
- **loadRackGearOnly:** Recibe la cremallera de la función *makeRack* y la visualiza en el visor.
- **makeRack:** Utilizando los valores que recibe por parámetro construye la cremallera de dientes rectos o helicoidales.
- **mustExecute:** Clase que hereda de *Feature* y es la responsable de dejar al usuario editar los valores de la geometría y ver la edición en el visor.
- **execute:** Clase que hereda de *Feature* y es la responsable de visualizar el engranaje en el visor.

Clases y funciones implementadas

El módulo implementado contiene 5 clases y 16 funciones para un total de 1304 líneas de código. A continuación, se describen las clases y funcionalidades implementadas para una mejor comprensión de la solución.

La clase **DlgRackGear:** implementa todo lo relacionado a la interfaz. Recibe los datos que introduce el usuario, verifica que estén correctos, calcula los datos fundamentales para la construcción del engranaje y los muestra en una tabla. Dicha clase contiene las funciones:

- **on_calcular_clicked:** Llama al método *excecuteCalculation* y si no hay errores en los datos introducidos llama a la función *showCalculation* para mostrar los resultados de los cálculos en la tabla de la interfaz.
- **on_aceptar_clicked:** Realiza la acción del botón *Accept* cerrando la ventana principal y mostrando el engranaje diseñado, al pulsarlo, cuando los datos están correctos. Si los datos no están correctos envía un mensaje de error en la ventana de notificaciones.
- **on_cancelarDesing_clicked:** Realiza la acción del botón *Cancel* cerrando la ventana principal al pulsarlo.
- **excecuteCalculation:** Ejecuta los cálculos de las propiedades de los engranajes y es el encargado de mostrar un mensaje en la ventana de notificaciones acercando al usuario al error cometido en la inserción de los datos o informando que los datos introducidos son correctos.
- **showCalculation:** Muestra los resultados de los cálculos en la tabla de la interfaz principal.

- **on_radioButtonRack_clicked:** Oculta los campos para introducir los datos del piñón.
- **on_radioButtonPinionRack_clicked:** Muestra los campos para introducir los datos del piñón en caso que estén ocultos.

La clase **HelicalSpur** se encarga de construir el piñón con dientes rectos o helicoidales. Esta clase contiene las siguientes funciones:

- **loadHelicalSpur:** Recibe la cremallera de la función *makeHelicalSpur* y la envía a *loadRackGear*.
- **makeHelicalSpur:** Utilizando los valores que recibe por parámetro construye el piñón de dientes rectos o helicoidales.
- **makeCentralHole:** Devuelve un sólido con el agujero central en forma de círculo de un sólido pasado por parámetros.
- **pointIntercectionC1C2:** Realiza la intersección de dos curvas en el espacio.

La clase **RackGear** es la principal e interactúa directamente con la interfaz y provee a la clase HelicalSpur la información necesaria para la construcción del piñón. La misma contiene las funciones:

- **loadRackGear:** Recibe el piñón de la función *loadHelicalSpur*, lo engrana con la cremallera que recibe de la función *makeRack* y visualiza el engrane en el visor.
- **loadRackGearOnly:** Recibe la cremallera de la función *makeRack* y la visualiza en el visor.
- **makeRack:** Utilizando los valores que recibe por parámetro construye la cremallera de dientes rectos o helicoidales.
- **mustExecute:** Clase que hereda de *Feature* y es la responsable de dejar al usuario editar los valores de la geometría y ver la edición en el visor.
- **execute:** Clase que hereda de *Feature* y es la responsable de visualizar el engranaje en el visor.

3.1.3. Diagrama de componentes

Un diagrama de componentes representa las dependencias entre componentes
textitsoftware, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables; 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).

En la figura 3.10 se ilustra el diagrama de componentes de la solución.

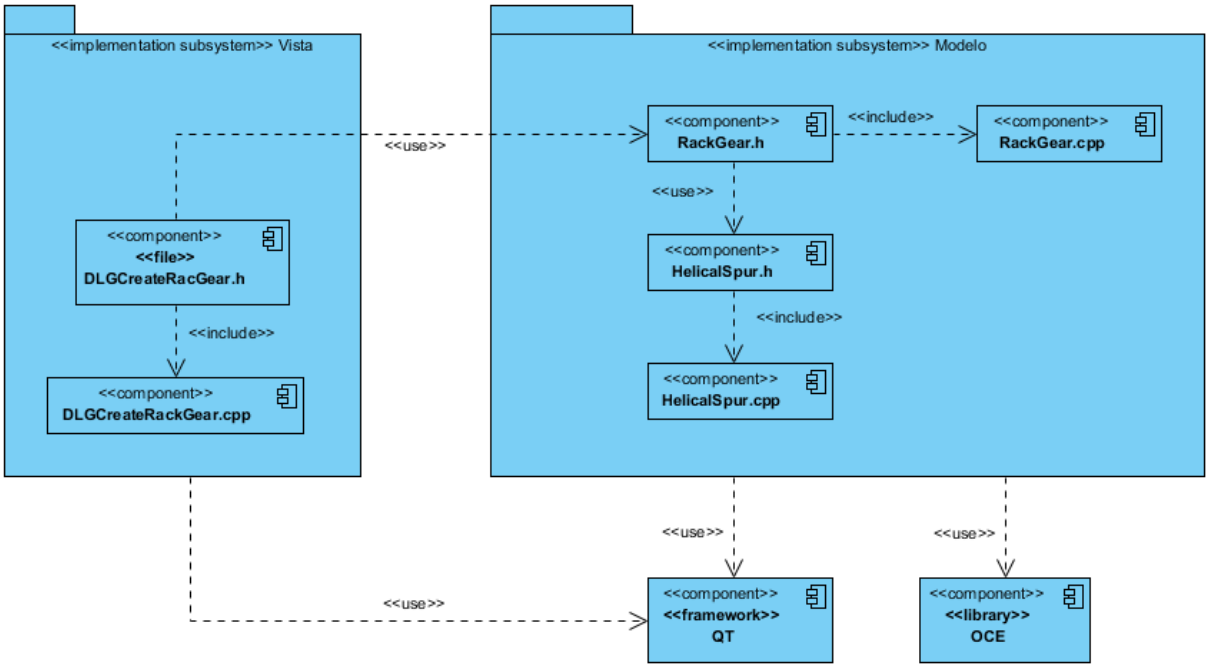


Figura 3.10. Diagrama de componentes.

3.2. Pruebas

Ian Sommerville en su libro ”Ingeniería del *software*” plantea que el proceso de prueba de *software* tiene dos objetivos distintos (Sommerville, 2005):

- Demostrar al desarrollador y al cliente que el *software* satisface sus requisitos. Esto significa que debería haber al menos una prueba para cada requerimiento o característica que se incorporará a la entrega del producto.
- Descubrir defectos en el *software* en el que el comportamiento de este es incorrecto, no deseable o no cumple su especificación. La prueba de defectos está relacionada con la eliminación de todos los tipos de comportamientos del sistema no deseables, tales como caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrupción de datos.

3.2.1. Niveles de las pruebas

Cuando se aplican un conjunto de 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 las pruebas son (Ruiz Tenorio, 2010):

- **Prueba Unitaria o de Unidad:** se centra en el proceso de verificación de la menor unidad del di-

seño 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 Funcionales:** 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 y funcionales.

3.2.2. Métodos 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 (Ruiz Tenorio, 2010):

- **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.

3.2.3. 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.

Para validar los requisitos funcionales de la aplicación se diseñan casos de prueba a partir de las historias de usuario, con el objetivo de mostrar la mayor cantidad posible de deficiencias en la aplicación.

En la tabla 3.1 se muestra el caso de prueba correspondiente a la historia de usuario "Introducir los parámetros comunes del engranaje piñón cremallera." Para un mayor estudio de estos ver el Anexo B.

Tabla 3.1. Caso de prueba de la Historia de Usuario # 1

Descripción general			
Permitir la inserción de los parámetros comunes del piñón y la cremallera.			
SC 1 Especificar los parámetros comunes del engranaje.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de especificar los parámetros comunes del engranaje	Se introducen los valores de los parámetros para modelar el engranaje piñón cremallera.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules.	*Acelertor/RackGear /Calculate*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelertor/RackGear /Cancel*

3.2.4. Pruebas Unitarias

Las pruebas unitarias se encuentran dentro de las técnicas de pruebas de caja blanca pues se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente (Pressman, 2005).

Para realizar las pruebas unitarias se utiliza el framework Qtest. En la figura 3.11 se muestra una imagen que evidencia la realización de las pruebas unitarias sobre el componente y la aceptación de todas las comprobaciones.

```
Debugging starts
********* 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::calculationPitch()
PASS   : TestUnitTest::calculationAddendum()
PASS   : TestUnitTest::calculationDedendum()
PASS   : TestUnitTest::calculationToothDepth()
PASS   : TestUnitTest::calculationMountingDistance()
PASS   : TestUnitTest::calculationRackLength()
PASS   : TestUnitTest::calculationTransversePressureAngle()
PASS   : TestUnitTest::calculationReferenceDiameter()
PASS   : TestUnitTest::calculationBaseDiameter()
PASS   : TestUnitTest::calculationTipDiameter()
PASS   : TestUnitTest::calculationRootDiameter()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 13 passed, 0 failed, 0 skipped, 0 blacklisted
********* Finished testing of TestUnitTest *********
Debugging has finished
```

Figura 3.11. Resultado de prueba unitaria realizada con Qt Test.

3.2.5. Resultados de las pruebas

Con la ejecución de las pruebas de caja negra se lograron identificar un grupo de no conformidades, las cuales se ilustran en la tabla 3.3.

Tabla 3.3. No Conformidades

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	Los radio button para seleccionar el sentido de la hélice no funcionan correctamente.	Baja	Resuelta
2	RF 2	El campo *Number of Theet* muestra números reales en vez de enteros.	Baja	Resuelta
3	RF 3	Los valores de las correcciones de las caras del piñón no muestran un mensaje de error al ser mal introducidos.	Baja	Resuelta
4	RF 4	No se modela la cremallera de dientes rectos de forma correcta.	Alta	Resuelta
5	RF 5	La cremallera de dientes helicoidales no se visualiza con la cantidad de dientes entrados por la interfaz.	Alta	Resuelta
6	RF 6	El perfil del diente del piñón de dientes rectos no se construyó correctamente.	Alta	Resuelta
7	RF 6	El engranaje piñón cremallera de dientes rectos no engrana correctamente.	Alta	Resuelta
8	RF 7	El perfil del diente del piñón de dientes helicoidales no se rotó correctamente.	Alta	Resuelta
9	RF 7	El engranaje piñón cremallera de dientes helicoidales no se visualiza correctamente.	Alta	Resuelta
10	RF 8	Los valores de la tabla no se actualizan al volver a presionar el botón *Calculate*.	Media	Resuelta

En el gráfico de la figura 3.12 se muestra un resumen correspondiente al proceso de pruebas.

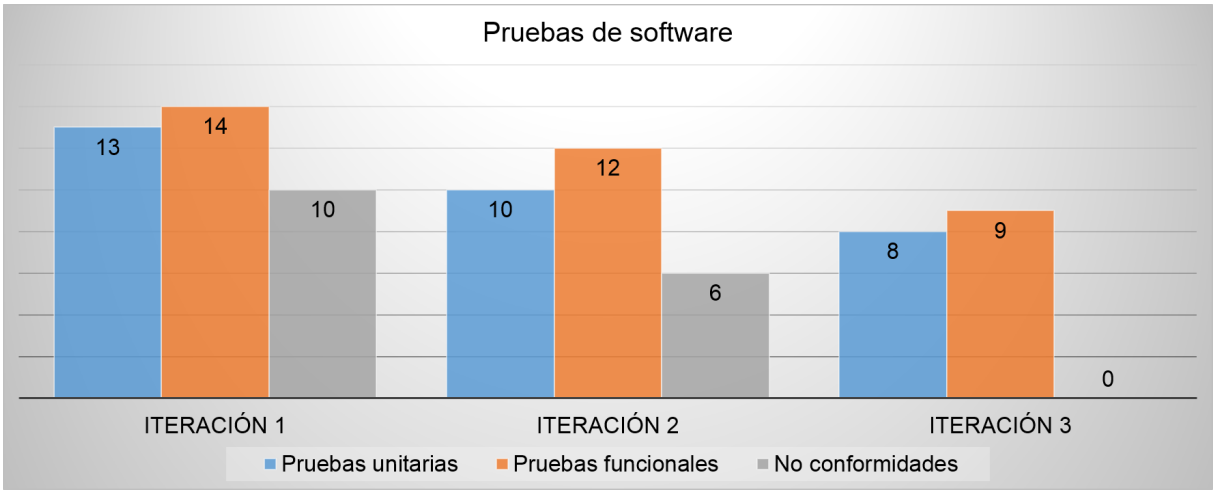


Figura 3.12. Relación de las iteraciones del proceso de pruebas.

3.3. Conclusiones del capítulo

En el presente capítulo se hizo referencia a los aspectos fundamentales en la implementación y la validación del módulo para el diseño de engranajes piñón cremallera. 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.

Conclusiones

De la investigación realizada se concluye que:

- Con el módulo desarrollado se logra acelerar el proceso de diseño del par cinemático piñón cremallera de 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 la cremallera de forma independiente.
- La caracterización de los sistemas de código abierto permitió identificar las potencialidades y limitaciones para el modelado de engranajes piñón cremallera, lo que permitió proyectar la idea de un módulo para acelerar el diseño de dicho mecanismo.

Recomendaciones

A partir del trabajo realizado y luego de haber analizado los resultados obtenidos se recomienda para un futuro perfeccionamiento:

- Incluir otros tipos de cremallera al engranaje.
- Incluir los materiales con los que se pueden fabricar este tipo de engranaje.
- Incluir los cálculos del engranaje piñón cremallera teniendo en cuenta las propiedades físicas de los materiales.

Glosario

acelerador de diseño Componente que automatiza las selecciones y la creación de la geometría, mejora la calidad del diseño inicial mediante la validación frente a los requisitos de diseño, y aumenta la normalización mediante la selección de los mismos componentes para las mismas tareas. 4

framework Plataforma, entorno o marco de trabajo. Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. 15–17, 21, 40

Acrónimos

AM métodos ágiles. 13
AUP Proceso Unificado Ágil. 13, 14

CAD Diseño Asistido por Computadora. 3, 5, 15
CAE Ingeniería Asistida por Computadora. 15
CAM Fabricación Asistida por Computadora. 15

GOF The Gang of Four. 26
GPL Licencia Pública General. 16
GRASP Patrones Generales de Software Para Asignar Responsabilidades. 25

IEEE Institute of Electrical and Electronics Engineers. 23

LGPL Licencia Pública General Reducida. 15, 16

OCCT Open CASCADE Technology. 15
OCE Opencascade Community Edition. 15, 17, 21

RUP Proceso Unificado Racional. 13

SIPII Soluciones Informáticas Para la Ingeniería y la Industria. 1, 4, 16, 19

UML Lenguaje Unificado de Modelado. 17, 19, 24

Referencias bibliográficas

Araworks (2018). *Ara Works. Precios para compra*. URL: http://araworks.es/precios-productos-solidworks/ (visitado 15-03-2018) (vid. pág. 4).

Asidek (2018a). *AutoCad 2019*. URL: https://www.asidek.es/producto/autocad-2019/ (vid. pág. 4).

— (2018b). *Autodesk Inventor 2017*. URL: https://www.asidek.es/producto/autodesk-inventor-2017/ (vid. pág. 4).

Barzaga Pérez, Yarisleydis (2011). «Componente de Interpretación de registros» (vid. pág. 19).

Computer Society, IEEE (2004). *Guide to the Software Engineering Body of Knowledge* (vid. pág. 23).

Costas Rodríguez, Daniel y Eduardo Gómez García (2011). «Simulador didactico de tallado de engranajes.» En: (vid. págs. 6, 7).

Edeki, Charles (2017). *Agile Unified Process*. URL: http://www.ijcsma.com/publications/september2013/V1I304.pdf (visitado 15-12-2018) (vid. pág. 13).

eFunda (2017). *eFunda: About Us*. URL: http://www.efunda.com/about/about.cfm (visitado 08-12-2017) (vid. pág. 13).

Engineering, Skyciv (2013). *Bending Moment and Shear Force Diagram Calculator*. URL: http://bendingmomentdiagram.com/ (visitado 13-12-2017) (vid. pág. 16).

Fernández Ferrer, Abel (2015). «Componente para la modelación de árboles escalonados en el Sistema CAD 2D». Universidad de las Ciencias Informáticas (vid. pág. 27).

FreeCAD (2017). *FreeCAD*. URL: http://freecadweb.org (visitado 07-12-2017) (vid. pág. 12).

GItLab (2018). *GItLab*. URL: https://about.gitlab.com/ (visitado 17-02-2018) (vid. pág. 17).

González, Antonio (2008). *Mecanismo de piñón cremallera.* URL: https://aprendemostecnologia.org/2008/09/04/mecanismo-de-pinon-cremallera/ (visitado 16-01-2018) (vid. pág. 2).

Guerrero, CA., JM. Suárez y LE. Gutiérrez (2009). *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* (Vid. pág. 26).

Inventor, Autodesk (2017). *Autodesk Inventor Profesional 2017*. URL: https://www.asidek.es/producto/autodesk-inventor-2017/ (visitado 15-03-2018) (vid. pág. 11).

Jacobson, Ivar, James Rumbaugh y Grady Booch (2000). *El Lenguaje Unificado de Modelado. Manual de Referencia* (vid. pág. 40).

Jeffries, R., A. Anderson y Hendrickson C. (2001). *Extreme Programming Installed.* (Vid. pág. 22).

Larman, Craig (1999). *UML y Patrones. Introducción al analisis y diseño orientado a objetos* (vid. págs. 24-26).

Michael, George (2015). *Kohara gear Company of Japan.* (Vid. págs. 8, 31, 33).

MITCalc (2017). *MITCalc - Mechanical, Industrial and Technical Calculations*. URL: http://www.mitcalc.com/ (visitado 08-12-2017) (vid. pág. 13).

OpenCascadeCommunityEdition (2017). *Open CASCADE Technology, The Open Source 3D Modeling Libraries*. URL: http://dev.opencascade.org/ (visitado 10-12-2017) (vid. pág. 15).

OpenCascadeTechnology (2017). *Open CASCADE Technology: Overview*. URL: http://dev.opencascade.org/doc/overview/html/index.html#OCCT_OVW_SECTION_1 (visitado 09-12-2017) (vid. pág. 15).

PLM, Siemens (2018). *Siemens. Ingenuity for life*. URL: https://www.plm.automation.siemens.com/store/en-us/solid-edge/ (visitado 15-03-2018) (vid. págs. 4, 11).

Pressman, Roger S. (2005). *Ingeniería del software. Un enfoque práctico.* 6ta Edición (vid. págs. 21, 23, 24, 43).

Qt, About (2015). *About Qt - Qt Wiki*. URL: https://wiki.qt.io/About_Qt (visitado 12-12-2017) (vid. pág. 16).

Rodríguez Sánchez, Tamara (2015). «Metodología de desarrollo para la actividad productiva de la UCI. Universidad de las Ciencias Informáticas». En: (vid. pág. 14).

Ruiz Tenorio, Roberto (2010). *Las pruebas de software y su importancia en las organizaciones.* (Vid. págs. 41, 42).

SolidWorks (2017). *SolidWorks_Corporation*. URL: http://www.solidworks.com.mx (visitado 06-12-2017) (vid. pág. 12).

Sommerville, Ian (2005). *Ingeniería de Software*. Séptima edición (vid. pág. 41).

Stroustrup, Bjarne (1997). *The C++ Programming Language*. Third Edition (vid. pág. 16).

Thirugnanam, A., Praphul Das y Lenin Rakesh (2014). «Design and Fabrication of Rack and Pinion Lift». En: *Bharath Institute of Science and Technology*. ISSN: 1990-9233 (vid. pág. 2).

Tulio Piovan, Marcelo (2014). *Trenes de engranajes reductores, planetarios y diferenciales.* (Vid. pág. 7).

VisualParadim (2017). *Software Design Tools for Agile Teams, with UML, BPMN and More*. URL: https://www.visual-paradigm.com/ (visitado 15-12-2017) (vid. pág. 17).

Generado con LaTeX: 20 de junio de 2018: 3:39pm

Apéndices

APÉNDICE A

Historias de usuario

Tabla A.1. Historia de usuario # 2

Historia de usuario	
Número: 1	**Nombre:** Introducir los parámetros de la cremallera.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 0.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 0.5 semanas
Descripción: 1. Objetivo • Permitir la inserción de los parámetros de la cremallera. **2. Acciones para lograr el objetivo (precondiciones y datos)** • Para insertar los parámetros de la cremallera hay que tener en cuenta los siguientes datos: número de dientes, ancho de la cara y altura de la cremallera. **3. Comportamiento válido y no válido** • El número de dientes, ancho de la cara y altura de la cremallera deben ser valores numéricos reales positivos y son valores obligatorios. **4. Flujo de acción** • Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz. • Si los datos están incorrectos se lanza un mensaje de error en letras rojas en la parte inferior de la interfaz que deberá acercar al usuario al error de construcción, dándole a la oportunidad nuevamente de realizar la acción en cuestión. • Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.	
Observaciones:	

Continuación de la página anterior

Tabla A.1. Continuación de la página anterior

Interfaz:

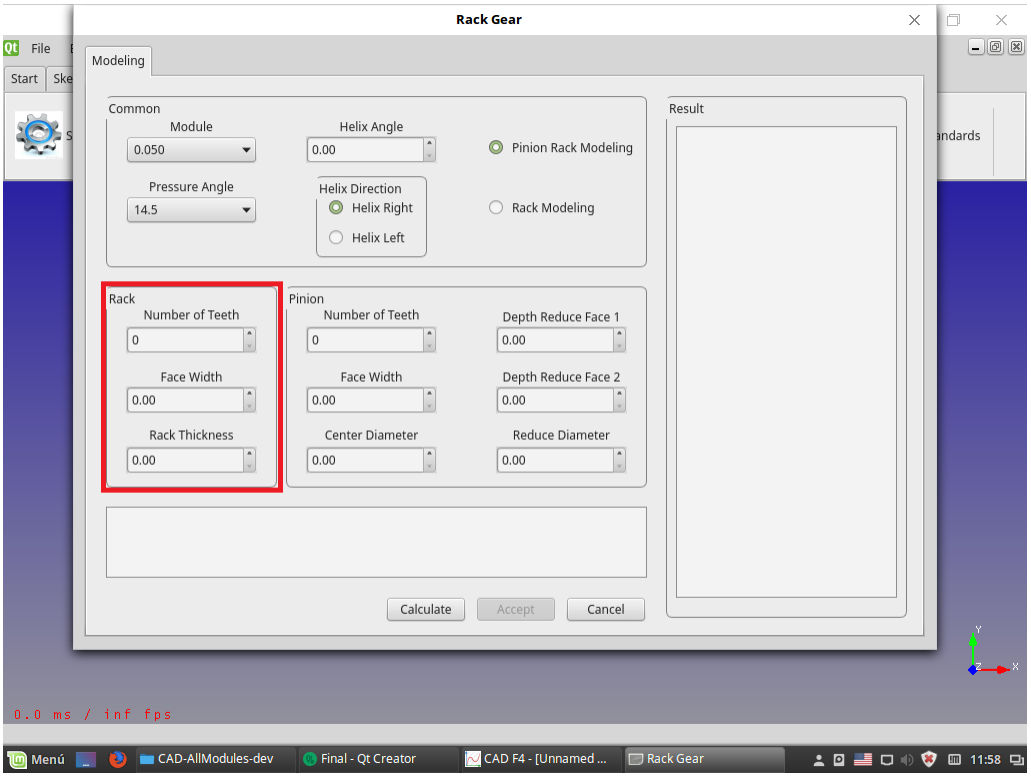


Tabla A.2. Historia de usuario # 3

Historia de usuario	
Número: 1	**Nombre:** Introducir los parámetros del piñón.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 0.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 0.5 semanas

Continuación de la página anterior

Tabla A.2. Continuación de la página anterior

Descripción: 1. Objetivo

- Permitir la inserción de los parámetros del piñón.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para insertar los parámetros del piñón hay que tener en cuenta los siguientes datos: *radio button Pinion Rack Modeling*, número de dientes, ancho de la cara y altura del piñón, además de la reducción de las caras 1 y 2 y la reducción del diámetro.

3. Comportamiento válido y no válido

- Debe estar marcado el *radio button Pinion Rack Modeling*
- El número de dientes, ancho de la cara y altura del piñón deben ser valores numéricos reales positivos y son valores obligatorios.
- La reducción de la cara 1, reducción de la cara 2 y reducción del diámetro deben ser valores numéricos reales positivos.

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz.
- Si los datos están incorrectos se lanza un mensaje de error en letras rojas en la parte inferior de la interfaz que deberá acercar al usuario al error de construcción, dándole a la oportunidad nuevamente de realizar la acción en cuestión.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Observaciones:

Continuación de la página anterior

Tabla A.2. Continuación de la página anterior

Interfaz:

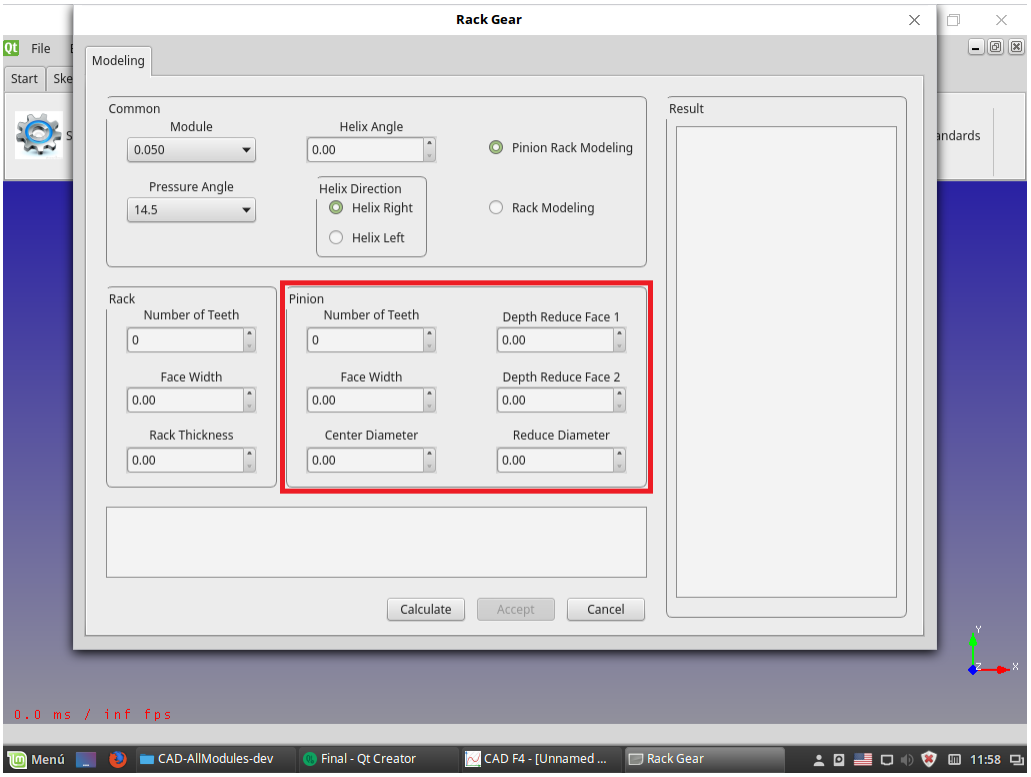


Tabla A.3. Historia de usuario # 4

Historia de usuario	
Número: 1	**Nombre:** Visualizar la cremallera de dientes rectos en tres dimensiones.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 1.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 1.5 semanas

Continuación de la página anterior

Tabla A.3. Continuación de la página anterior

Descripción: 1. Objetivo

- Permitir la visualización de la cremallera de dientes rectos en tres dimensiones.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para visualizar el engranaje de cremallera de dientes rectos en tres dimensiones hay que haber realizado los calculos correspondientes a este tipo de engranaje.

3. Comportamiento válido y no válido

-

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se activa el botón de acción *Accept*, el cual cierra la interfaz de construcción del engranaje y muestra el diseño del engranaje en el visor en 3 dimensiones.Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se actualizarán los datos correspondientes a los cálculos de dicho engranaje en una tabla que se encuentra en la parte derecha de la interfaz.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Observaciones:

Interfaz:

Tabla A.4. Historia de usuario # 5

Historia de usuario	
Número: 1	**Nombre:** Visualizar la cremallera de dientes helicoidales en tres dimensiones.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 4ta
Prioridad: Alta	**Tiempo Estimado:** 1.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 semanas

Continuación de la página anterior

Tabla A.4. Continuación de la página anterior

Descripción: 1. Objetivo

- Permitir la visualización de la cremallera de dientes helicoidales en tres dimensiones.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para visualizar el engranaje de cremallera de dientes helicoidales en tres dimensiones hay que haber realizado los calculos correspondientes a este tipo de engranaje.

3. Comportamiento válido y no válido

-

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se activa el botón de acción *Accept*, el cual cierra la interfaz de construcción del engranaje y muestra el diseño del engranaje en el visor en 3 dimensiones.Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se actualizarán los datos correspondientes a los cálculos de dicho engranaje en una tabla que se encuentra en la parte derecha de la interfaz.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Observaciones:

Interfaz:

Tabla A.5. Historia de usuario # 6

Historia de usuario	
Número: 1	**Nombre:** Visualizar el engranaje piñón cremallera de dientes rectos.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:** 2 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 3 semanas

Continuación de la página anterior

Tabla A.5. Continuación de la página anterior

Descripción: 1. Objetivo

- Permitir la visualización del engranaje de cremallera piñón dientes rectos en tres dimensiones.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para visualizar el engranaje de cremallera piñón dientes rectos en tres dimensiones hay que haber realizado los calculos correspondientes a este tipo de engranaje.

3. Comportamiento válido y no válido

-

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se activa el botón de acción *Accept*, el cual cierra la interfaz de construcción del engranaje y muestra el diseño del engranaje en el visor en 3 dimensiones.Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se actualizarán los datos correspondientes a los cálculos de dicho engranaje en una tabla que se encuentra en la parte derecha de la interfaz.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Observaciones:

Interfaz:

Tabla A.6. Historia de usuario # 7

Historia de usuario	
Número: 1	**Nombre:** Visualizar el engranaje piñón cremallera de dientes helicoidales.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 5ta
Prioridad: Alta	**Tiempo Estimado:** 3 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 4 semanas

Continuación de la página anterior

Tabla A.6. Continuación de la página anterior

Descripción: 1. Objetivo

- Permitir la visualización del engranaje de cremallera piñón dientes helicoidales en tres dimensiones.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para visualizar el engranaje de cremallera piñón dientes helicoidales en tres dimensiones hay que haber realizado los calculos correspondientes a este tipo de engranaje.

3. Comportamiento válido y no válido

-

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se activa el botón de acción *Accept*, el cual cierra la interfaz de construcción del engranaje y muestra el diseño del engranaje en el visor en 3 dimensiones.Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se actualizarán los datos correspondientes a los cálculos de dicho engranaje en una tabla que se encuentra en la parte derecha de la interfaz.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Observaciones:

Interfaz:

Tabla A.7. Historia de usuario # 8

Historia de usuario	
Número: 1	**Nombre:** Visualizar los resultados del calculo de los engranajes.
Programador: Reimnel Drake Bueno	**Iteración Asignada:** 6ta
Prioridad: Alta	**Tiempo Estimado:** 0.5 semanas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 0.5 semanas
Descripción:	

Continuación de la página anterior

Tabla A.7. Continuación de la página anterior

Observaciones: 1. Objetivo

- Permitir la visualización de los resultados del calculo de los engranajes.

2. Acciones para lograr el objetivo (precondiciones y datos)

- Para visualizar los resultados del calculo de los engranajes hay que haber realizado los cálculos correspondientes.

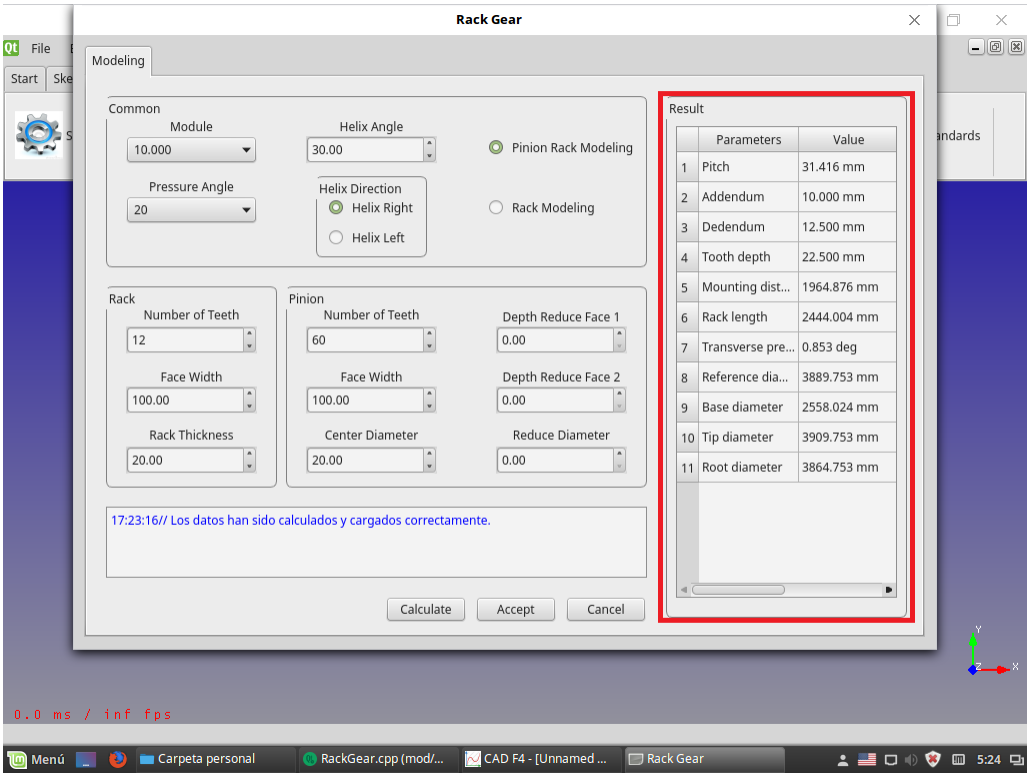
3. Comportamiento válido y no válido

-

4. Flujo de acción

- Cuando el usuario introduzca correctamente los datos necesarios y seleccione la opción *Calculate* se mostrará un mensaje en letras azules, en una pequeña área blanca que se encuentra insertada en la parte inferior de la interfaz y, a su vez, se actualizarán los datos correspondientes a los cálculos de dicho engranaje en una tabla que se encuentra en la parte derecha de la interfaz.
- Si se selecciona el botón *Cancel* se cancela la acción y se cierra la ventana.

Interfaz:



APÉNDICE B

Casos de prueba

Tabla B.1. Caso de prueba de la Historia de Usuario # 2

Descripción general			
Permitir la inserción de los parámetros de la cremallera.			
SC 1 Especificar los de la cremallera.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de especificar los parámetros de la cremallera.	Se introducen los valores de los parámetros para modelar la cremallera.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules.	*Acelerator/RackGear /Calculate*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator/RackGear /Cancel*

Tabla B.3. Caso de prueba de la Historia de Usuario # 3

Descripción general			
Permitir la inserción de los parámetros del piñón.			
SC 1 Especificar los parámetros del piñón.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de especificar los parámetros del piñón.	Se introducen los valores de los parámetros para modelar el piñón.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules.	*Acelerator*/*RackGear* /*Calculate*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*

Tabla B.5. Caso de prueba de la Historia de Usuario # 4

Descripción general			
Permitir la visualización de la cremallera de dientes rectos en tres dimensiones.			
SC 1 Visualizar la cremallera de dientes rectos en tres dimensiones.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de visualizar la cremallera de dientes rectos en tres dimensiones.	Se presiona el botón *Accept*.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules y se cierra la interfaz de construcción del engranaje y se muestra el diseño del engranaje en el visor en 3 dimensiones..	*Acelerator*/*RackGear* /*Accept*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*

Tabla B.7. Caso de prueba de la Historia de Usuario # 5

Descripción general			
Permitir la visualización de la cremallera de dientes helicoidales en tres dimensiones.			
SC 1 Visualizar la cremallera de dientes helicoidales en tres dimensiones.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de visualizar la cremallera de dientes helicoidales en tres dimensiones.	Se presiona el botón *Accept*.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules y se cierra la interfaz de construcción del engranaje y se muestra el diseño del engranaje en el visor en 3 dimensiones..	*Acelerator*/*RackGear* /*Accept*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*

Tabla B.9. Caso de prueba de la Historia de Usuario # 6

Descripción general			
Permitir la visualización del engranaje piñón cremallera de dientes rectos en tres dimensiones.			
SC 1 Visualizar del engranaje piñón cremallera de dientes rectos en tres dimensiones.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de visualizar el engranaje piñón cremallera de dientes rectos en tres dimensiones.	Se presiona el botón *Accept*.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules y se cierra la interfaz de construcción del engranaje y se muestra el diseño del engranaje en el visor en 3 dimensiones..	*Acelerator*/*RackGear* /*Accept*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*

Tabla B.11. Caso de prueba de la Historia de Usuario # 7

Descripción general			
Permitir la visualización del engranaje piñón cremallera de dientes helicoidales en tres dimensiones.			
SC 1 Visualizar el engranaje piñón cremallera de dientes helicoidales en tres dimensiones.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de visualizar el engranaje piñón cremallera de dientes helicoidales en tres dimensiones.	Se presiona el botón *Accept*.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules y se cierra la interfaz de construcción del engranaje y se muestra el diseño del engranaje en el visor en 3 dimensiones..	*Acelerator*/*RackGear* /*Accept*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*

Tabla B.13. Caso de prueba de la Historia de Usuario # 8

Descripción general			
Permitir la visualización de los resultados del calculo de los engranajes.			
SC 1 Visualizar los resultados del calculo de los engranajes.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de especificar los parámetros comunes del engranaje	Se introducen los valores de los parámetros para modelar el engranaje deseado.	Si los datos no están correctos se muestra un mensaje en letras rojas en la ventana ubicada en la parte inferior de la interfaz que debe acercar al usuario al error cometido. Si los datos están correctos se muestra un mensaje en letras azules y se visualizan los valores en la tabla *Result*.	*Acelerator*/*RackGear* /*Calculate*
EC 1.2 Opción de Cancelar	Selecciona la opción *Cancel*	Elimina los datos creados y cierra la ventana.	*Acelerator*/*RackGear* /*Cancel*