Universidad de las Ciencias Informáticas
Facultad 4

Componente para la modelación de engranajes cilíndricos de dientes rectos

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

Autor:
Sandy García Santos

Tutores:
Dr.C Augusto Cesar Rodríguez Medina
Ing. Leonel Hernández López

La Habana, junio de 2015
"Año 57 de la Revolución"

Declaración de autoría

Declaro ser único autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2015.

_____

Sandy García Santos

Firma del Autor

_____

Dr.C Augusto Cesar Rodríguez Medina

Firma del Tutor

_____

Ing. Leonel Hernández López

Firma del Tutor

"Yo he preferido hablar de cosas imposibles, porque de lo posible se sabe demasiado…"

Silvio Rodríguez Domínguez.

Agradecimientos

A mis tutores Augusto y Leonel por toda su ayuda.

A mi mamá, mi papá y mis abuelos.

A Adielys.

A Ocleidy.

A toda mi familia.

A mis compañeros de proyecto Angel y Abel.

A todos mis compañeros de grupo y de apartamento.

Resumen

El presente trabajo contiene los resultados del proceso de desarrollo de un componente para generar el modelo geométrico de un sistema de engranajes cilíndricos con dientes rectos, utilizando la tecnología *OpenCascade*, el *framework* de Qt y el lenguaje de programación C++; con el requerimiento de una arquitectura flexible que facilite el empleo del módulo, para poder utilizarlo en cualquier aplicación de código abierto destinada al diseño asistido por computadora.

Palabras clave:

Acelerador de diseño, diseño asistido por computadora (*Computer Aided Design*).

Índice

Introducción

El proceso de investigación y desarrollo, cuyo resultados se exponen en los capítulos de este documento, tuvo su origen en las necesidades de un proyecto de la Facultad 4, cuyo propósito es el desarrollo de una aplicación para el diseño asistido por computadoras, destinado a sectores de la industria nacional vinculados a la actividad de diseño. En ese contexto, una de las encomiendas fue automatizar el proceso de modelación geométrica de engranajes cilíndricos con dientes rectos y perfiles de envolventes utilizando la tecnología *OpenCascade* y el *framework* Qt. El documento está estructurado en tres capítulos: en el primero se expone la fundamentación teórica de la investigación, se define la metodología, la arquitectura, las tecnologías y las herramientas; en el segundo son descritos los principales conceptos relacionados con la investigación a través de un modelo de dominio, se realizó la descripción de la solución y se identifican las características del sistema y los aspectos funcionales los cuales se agrupan en historias de usuario, se presentan los patrones de diseño existentes en el componente, los prototipos de las interfaces de usuario y la metodología de cálculo para engranajes cilíndricos con dientes rectos; y en el tercero se define el estándar de codificación, se presentan las tareas de ingeniería, se definen los tipos de pruebas y los casos de prueba realizados al componente.

Capítulo I: Fundamentación teórica de la investigación

Este trabajo está relacionado con un proyecto de investigación destinado a obtener un sistema informático para el dibujo técnico mecánico en dos dimensiones (2D), este tipo de aplicación (CAD) es ampliamente utilizada en las etapas de diseño por parte de los ingenieros de diversos perfiles (1). Actualmente se utilizan sistemas propietarios para estas labores, pero no de manera institucional, los más conocidos y difundidos en Cuba son AutoCAD e Inventor**,** ambos de AutoDesk.

Como se verá en el apartado 1.1 cuando se profundiza en la situación problémica que da origen a la investigación, existen dificultades que conducen a sostener la idea de la necesidad de remplazar los mencionados sistemas para evitar conflictos por el uso indebido de éstos. Como básicamente se trata de sistemas para el diseño y el dibujo técnico, resulta conveniente conocer algunos aspectos de esa actividad para comprender su importancia en el ámbito de la ingeniería.

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

En todas las ramas de la ingeniería, como regla, es de amplio empleo el lenguaje gráfico, este consiste en plasmar información técnica del producto que se quiere desarrollar o documentar mediante recursos geométricos. Normalmente, la literatura se refiere a esto como dibujo técnico, este se utiliza en muchas ramas como la mecánica, electricidad, hidráulica, entre otras.

Al revisar datos históricos se aprecia que el dibujo técnico tradicional se hacía, en sus inicios, utilizando carbón, regla, cartabón, compás y semicírculo; estos implementos reciben la denominación de instrumentos de trazado y la documentación relativa a un producto podía estar conformada por un conjunto de pliegos de papel en el que se plasmaban los planos constructivos e información técnica y tecnológica en general.

Entre los siglos **XIX** y **XX** las tecnologías para el dibujo estaban confinadas a una estructura básica, compuesta por el proyectista, con la mesa de dibujo y sus instrumentos de trazado. Precisamente las mesas de dibujo han sufrido transformaciones para tratar de facilitar la productividad y el desempeño de los proyectistas, así podemos ver que en una época se dibujaba con algunos instrumentos y después algunos de estos estaban incorporados a la propia mesa.

Primeramente se utilizaban creyones de carbón para el trazado, después lápices con diferentes creyones de dureza, para los diferentes tipos de línea y finalmente se

entintaban los planos; fueron muy utilizados los sistemas de centropenes y plumillas. Estas tecnologías para el dibujo, dieron un salto significativo con la aparición de los sistemas de cómputo digital y fueron distinguiéndose algunos productos que durante varias décadas se han mantenido en el mercado (2); una de las empresas productoras de tecnologías para el diseño asistido por computadora es AutoDesk, cuyo producto insignia fue durante muchos años el AutoCAD (en estos momentos es Inventor). Hubo muchos sistemas con la misma filosofía de AutoCAD, que en primer lugar pretendía automatizar el proceso de elaboración de planos que se hacía por parte de numerosos proyectistas en mesas de dibujo, de esta forma hubo un avance considerable en la productividad al no tener que empezar a dibujar desde cero desechando un pliego inservible; con estos sistemas se podía borrar sin dañar los pliegos y se simplificaba el proceso al no tener que manipular numerosos documentos, se podían deshacer acciones y hacer copias de forma fácil, reproduciendo los datos en otro fichero que podía ser después reutilizado. Esta forma de hacer se mantuvo en la década del 80 del siglo **XX** y persiste aún, aunque de forma limitada.

En la década del 90 del siglo **XX** empiezan a aparecer sistemas para el modelado en tres dimensiones, respondiendo a una nueva concepción en la filosofía de diseño, sustentada en la forma natural en que piensa el hombre, el cual al imaginar no piensa en una vista concreta del objeto, sino en el propio objeto, es decir, en sus imágenes y no en la forma que se traza. Por ese motivo fue posible aumentar la productividad del trabajo de los proyectistas, debido a que es más fácil, “moldear” o “esculpir” un cuerpo en la pantalla de un computador y esta actividad fluye de forma natural; esto fue posible gracias a la implementación de técnicas para el trazado paramétrico, lo que facilitaba la modificación de los objetos gráficos dinámicamente. Antiguamente una figura o dibujo se acotaba y estas cotas se puede decir que eran *conducidas*[1] y posteriormente, con las técnicas paramétricas, aparece el concepto de cotas *conductoras*[2]; en la práctica, un objeto puede cambiar sus dimensiones dinámicamente al editar o reeditar las dimensiones expuestas en su cota (2).

AutoCAD es una herramienta propietaria, lo que significa que los ingenieros nunca son dueños del sistema, sino que reciben autorización para su uso por un tiempo cuando compran la licencia correspondiente.

[1]Dimensiones que se asignan a un objeto en ausencia del dibujo paramétrico.
[2] Dimensiones que se asignan a un objeto en presencia de dibujo paramétrico.

En nuestro país está extendido el uso del AutoCAD (2), pero no de forma institucional, debido a la prohibición de utilizar e importar todo tipo de sistemas privativos por las restricciones del bloqueo económico impuesto por los EEUU hace más de 50 años. Esto significa que las actividades de proyecto, ya sea realización o exportación de planos no tiene sustento en una herramienta soberana, pues las copias que se utilizan son copias piratas y "*crackeadas*". Las actividades de diseño mecánico y la ingeniería en general son socio-productivas, es decir, involucran a grupos de personas de diferentes ramas y perfiles; están tan generalizadas, que se pueden encontrar cientos de entidades dedicadas a esas líneas de trabajo; por ejemplo, en La Habana, pueden encontrarse numerosas empresas como astilleros, centros de servicios técnicos, fabriles y de construcción prácticamente en todos los ministerios (industria eléctrica, metal-mecánica, azucarera, talleres automotrices, de mantenimiento industrial y una lista extensa de otras entidades). La importancia de esto es tan grande que en las carreras técnicas, el AutoCAD se enseña como parte de la asignatura gráfica por computadora a pesar de ser un sistema privativo (3).

De aquí se derivan una serie de efectos negativos, por ejemplo: no es posible comercializar proyectos elaborados con herramientas privativas sin peligro de litigios o de acusaciones fuera del ámbito nacional por violar la propiedad intelectual, entre otras.

En varios sistemas como el AutoCAD, que se especializa en el dibujo en dos dimensiones y modelado en tres dimensiones (1); ciertas estructuras mecánicas complejas, es necesario construirlas partiendo de primitivas como la recta, el cuadrado, el triángulo y el círculo, lo cual es un proceso engorroso, pero productos como Catia[3], Solid Edge, Solid Works y Autodesk Inventor, los cuales cuentan con un módulo que automatiza la construcción de estructuras y piezas mecánicas complejas como engranajes cilíndricos y helicoidales, árboles escalonados, resortes de compresión y de extensión, varios tipos de tornillos, sistemas de poleas, entre otros. A estos módulos se le conoce, en algunas aplicaciones CAD como "*Aceleradores de diseño*", en otras, por "*Generadores de componente*", pero siempre su función es la misma, facilitarle el trabajo al usuario del sistema. Los sistemas mencionados son privativos, por lo que se hace imposible su uso de forma legal, por lo que se ha optado en el país por la alternativa de usar sistemas de código abierto.

[3] *Computer-Aided Three Dimensional Interactive Applications*

También existen herramientas de código abierto para el diseño asistido por computadora, el LibreCAD es una aplicación para el diseño en dos dimensiones con cierta similitud al AutoCAD**,** el FreeCAD se aproxima más a sistemas como Inventor y SolidEdge en sus funcionalidades para el modelado en tres dimensiones y otras, como son LibreCAD y FreeCAD; las mismas no cuentan con ningún modulo que automatice el proceso de creación de estructuras mecánicas complejas como las anteriormente mencionadas, por estos motivos, para poder confeccionar dichas estructuras, hay que realizarla desde las primitivas correspondientes, lo cual es muy difícil, siendo la única forma de poder exportar planos de diferentes estructuras por nuestro país.

Las aplicaciones a las cuales se les realizó la captura de requisitos, presentan aceleradores de diseño de varios tipos de engranajes, de resortes de compresión y de extensión de árboles, entre otras entidades mecánicas. Los engranajes son una de las piezas mecánicas de mayor uso en toda la industria debido a que permiten grandes transmisiones de potencia desde el eje de una fuente de energía hasta otro eje situado a cierta distancia. Realizan un trabajo sin pérdidas de energía y proporcionan a las máquinas una graduación utilizable de relaciones de velocidad. Por estas razones, los aceleradores de diseño de engranajes cilíndricos de dientes rectos poseen esa gran importancia para las diferentes ramas de la industria y la mecánica actualmente y, de una manera más sencilla, se pueden modelar varios engranajes acortando considerablemente el tiempo utilizado para este fin.

Partiendo de este análisis se formula el siguiente **problema de investigación**: Inexistencia de componente para automatizar la modelación en dos dimensiones de engranajes cilíndricos de dientes rectos en el proyecto “Sistema para el dibujo técnico mecánico en dos dimensiones, asistido por computadora”.

Para solucionar el problema planteado se propone como **objetivo general**, desarrollar un componente para la modelación de engranajes cilíndricos de dientes rectos para ser integrado en el sistema para el dibujo técnico mecánico en dos dimensiones, asistido por computadora.

El **objeto de estudio** de la investigación se centra en la modelación de entidades, teniendo como **campo de acción**, la modelación de engranajes cilíndricos de dientes rectos en el proyecto “Sistema para el dibujo técnico mecánico en dos dimensiones, asistido por computadora”.

A partir del objetivo general definido, se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico conceptual que sustenta la investigación.
- Análisis y diseño del componente.
- Implementación y prueba del componente.

Para dar cumplimiento a los objetivos específicos se proponen como **tareas de investigación**:

- Asimilación de los principales conceptos y tecnologías que se requieren para el desarrollo del componente.
- Elaboración del estado del arte del trabajo de diploma.
- Descripción detallada de los diferentes aspectos que conforman el componente.
- Obtención de requisitos a partir del acelerador de diseño de engranajes cilíndricos de dientes rectos de la aplicación **Autodesk Inventor**.
- Modelación del flujo de datos del componente.
- Diseño de la interfaz gráfica del componente.
- Elaboración de la estructura de clases que conforman el componente.
- Implementación del componente destinado a modelar y visualizar engranajes cilíndricos de dientes rectos.
- Visualización de los resultados del cálculo geométrico del engranaje con las funciones de *OpenCascade*.
- Comprobación del componente implementado.

Para organizar de una mejor forma el trabajo de desarrollo, se designaron las siguientes **preguntas de investigación**:

- ¿Cómo crear las circunferencias de construcción?
- ¿Cómo trazar un perfil de diente con la envolvente de círculo (Involuta)?
- ¿Cómo determinar el ancho de un diente del engranaje?

- ¿Cómo determinar y posicionar el punto medio del diente del engranaje?
- ¿Cómo establecer un eje simétrico?
- ¿Cómo determinar la altura del diente?
- ¿Cómo establecer el espacio entre dientes adecuado?
- ¿Cómo crear los demás dientes del engranaje partir de la construcción de un primer diente?

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:** permitió analizar diferentes sistemas CAD y llevarlos a su mínima expresión para poder comprenderlos completamente; desde el código fuente en el caso de los sistemas de código abierto. De esta manera se pudo realizar una captura de requisitos con las principales funcionalidades de dichos sistemas CAD**.**
- **Observación:** Posibilito realizar las pruebas al producto desarrollados, además de los principales problemas que presentan las herramientas CAD en software libre.

1.2 Aspectos generales acerca de los sistemas para el diseño y la ingeniería asistidos por computadoras

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) al pasar de los años han surgido disímiles tecnologías y herramientas que facilitan el trabajo a los desarrolladores. Se ha apreciado entre ellas, un especial auge de las conocidas como tecnologías CAD/CAE/CAM, que por sus siglas en inglés equivalen a "**C**omputer **A**ided **D**esign", "**C**omputer **A**ided **E**ngineering" y "**C**omputer **A**ided **M**anufacturing" respectivamente, o lo que es lo mismo; tecnologías de diseño, ingeniería y fabricación asistidos por computadora (1).

El CAD atiende prioritariamente aquellas tareas exclusivas del diseño, tales como el dibujo técnico y la documentación del mismo, pero normalmente permite realizar otras tareas complementarias relacionadas principalmente con la presentación y el análisis del diseño realizado. El término CAE engloba el conjunto de herramientas informáticas que

permiten analizar y simular el comportamiento del producto diseñado, y por último el término CAM se refiere al uso de computadoras para ayudar en todas las fases de la fabricación de un producto, incluyendo la planificación del proceso y la producción, mecanizado, calendarización, administración y control de calidad, con una intervención mínima del operario (1).

1.2.1 Componente para automatizar operaciones de diseño. (aceleradores de diseño)

Los componentes para automatizar operaciones de diseño (aceleradores del diseño, como es conocido en Autodesk), son herramientas que permiten la creación de piezas mecánicas, a partir del diseño y análisis de relaciones mecánicas de las piezas en lugar de descripciones geométricas. Dentro del acelerador del diseño se puede encontrar manuales de ingeniería y calculadores mecánicos, con estas herramientas se puede automatizar el proceso de la creación de piezas basadas en atributos del mundo real, como por ejemplo, velocidad, potencia y propiedades del material (5).

El acelerador de diseño se visualiza en forma de caja de diálogo y contiene diferentes pestaña que encapsula diferentes elementos y datos a través de los cuales permite diseñar, analizar y crear rápidamente componentes de maquinaria habituales basándose en las historias de usuario, especificaciones, atributos del mundo real y condiciones de uso como potencia, velocidad, par, propiedades del material, temperaturas de funcionamiento y condiciones de lubricación, entre otras. Para el diseño de cada pieza mecánica que se encuentra en el acelerador del diseño existe un generador de componentes, solo se realizará el generador de componentes de engranajes, en el cual se pueden realizar, ensamblajes de engranajes y optimizar su función mecánica.

1.2.2 Generalidades sobre engranajes

Una rueda dentada es un mecanismo de forma circular que transmite movimiento mediante dientes (6). Los dientes rodean la rueda en todo su perímetro. Existen diferentes tipos de ruedas dentadas dependiendo de su forma y colocación de los dientes; como por ejemplo ruedas dentadas cónicas, helicoidales y cilíndricas (6). La transmisión de movimiento entre ejes se tiene que realizar con ruedas dentadas que tengan el mismo paso y el mismo módulo de dientes. En un sistema de dos ruedas dentadas homólogas se llama *corona* al engrane de mayor diámetro y *piñón* al de menor diámetro (6).

Si el eje conductor tiene menor diámetro que el eje conducido se genera un mecanismo reductor y si el eje conducido es de menor diámetro se genera un mecanismo acelerador.

1.2.2.1 Engranajes cilíndricos de dientes rectos

Los engranajes cilíndricos rectos son el tipo de engranaje más simple y corriente que existe (6). Se utilizan generalmente para velocidades pequeñas y medias; a grandes velocidades, si no son rectificados, o ha sido corregido su tallado, producen ruido cuyo nivel depende de la velocidad de giro que tengan; sus elementos son los siguientes (2):

- **Diente 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:** el módulo de un engranaje es una característica de magnitud que se define como la relación entre la medida del diámetro primitivo expresado en milímetros y el número de dientes. En los países anglosajones se emplea otra característica llamada *Diametral Pitch*[4], que es inversamente proporcional al módulo. El valor del módulo se fija mediante cálculo de resistencia de materiales en virtud de la potencia a transmitir y en función de la relación de transmisión que se establezca. El tamaño de los dientes está normalizado. El módulo está indicado por números. Dos engranajes que engranen tienen que tener el mismo módulo.
- **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 vano 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. Se simboliza como (*Z*). Es fundamental para calcular la relación de transmisión. El número de dientes de un engranaje no debe estar por debajo de 18 dientes cuando el ángulo de presión es 20º ni por debajo de 12 dientes cuando el ángulo de presión es de 25º.

[4] El diametral pitch o paso diametral es el cociente entre el número de dientes (*Z*) y el diámetro primitivo (*d*), expresado en pulgadas.

- **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.
- **Flanco:** es la cara interior del diente, es su zona de rozamiento.
- **Altura del diente:** es la suma de la altura de la cabeza (*adendum*) más la altura del pie (*dedendum*).
- **Ángulo de presión:** el que forma la línea de acción con la tangente a la circunferencia de paso, φ (20º o 25º son los ángulos normalizados).
- **Largo del diente:** es la longitud que tiene el diente del engranaje.
- **Distancia entre centro de dos engranajes:** es la distancia que hay entre los centros de las circunferencias de los engranajes.
- **Relación de transmisión (Rt):** es la relación de giro que existe entre el piñón conductor y la rueda conducida. La *Rt* puede ser reductora de velocidad o multiplicadora de velocidad. La relación de transmisión recomendada tanto en caso de reducción como de multiplicación depende de la velocidad que tenga la transmisión con los datos orientativos que se indican (2):
  - **Velocidad lenta:** $R_t = \frac{1}{10}$
  - **Velocidad normal:** $R_t = \frac{1}{7} - \frac{1}{6}$
  - **Velocidad elevada:** $R_t = \frac{1}{4} - \frac{1}{2}$

- **Fórmulas constructivas de los engranajes rectos** (6)**.**
  - **Diámetro primitivo*:*** $D_p = Z \cdot M$
  - **Módulo:** $M = \frac{D_p}{Z}$
  - ***Paso circular:*** $P_c = \pi \cdot M$
  - ***Número de dientes:*** $Z = \frac{D_p}{M}$
  - ***Diámetro exterior:*** $D_e = (Z + 2) \cdot M$

- ***Espesor del diente:*** $E = \frac{P_c}{2}$
- ***Diámetro interior:*** $D_i = D_p - 2{,}50 \cdot M$
- ***Pie del diente:*** $1{,}25 \cdot M$
- ***Cabeza del diente:*** M
- ***Altura del diente:*** $2{,}25 \cdot M$
- ***Distancia entre centros:*** $\frac{(D_p - d_p)}{2}$
- ***Ecuación general de transmisión:*** $N \cdot Z = n \cdot z$

1.3 Sistemas informáticos para el diseño asistido por computadora

La producción de sistemas de tipo CAD a nivel mundial está muy diversificada. Independientemente de las tecnologías utilizadas, existen *software* especializados en la minería y al mismo tiempo se pueden encontrar soluciones orientadas a la arquitectura, la mecánica o incluso diseñados para la medicina o el audiovisual (3).

A la hora de realizar una descripción de las tendencias de desarrollo de *software* de tipo CAD hasta la actualidad, hay que mencionar la controversia existente entre el *software* privativo y en sistemas de código abierto. El *software* privativo al ser desarrollados por empresas muy bien respaldadas económicamente, siempre se ha caracterizado por ser una solución muy potente o profesional, pero al mismo tiempo traen como principal inconveniente el problema de las licencias y la dificultad económica que esto representa. Por otro lado, existen otras alternativas que demuestran que se pueden realizar proyectos muy profesionales bajo la filosofía de sistemas de código abierto(3).

En el proceso de investigación que se lleva a cabo a nivel de proyecto se logró identificar como una línea estratégica importante durante el desarrollo, la reutilización de todo el código disponible con funcionalidades probadas en otros sistemas de código abierto; de esta manera será posible reducir los tiempos hasta la obtención de las metas trazadas. Por ejemplo, se puede reutilizar código disponible en varios *software* como por ejemplo FreeCAD, LibreCAD y Salome Meca el cual es un sistema CAE (*Computer Aided Engineering*).

Con el fin de obtener conclusiones que ayuden en el desarrollo de la investigación, se analizarán varios sistemas CAD con licencia libre y privativa y así poder elegir de ellos los requerimientos más efectivos, eficaces y económicos a usar.

1.3.1 Sistemas para el diseño asistido por computadoras con licencia privada

- Catia **(***computer-aided three dimensional interactive application***)**

  Catia es un programa informático de diseño, fabricación e ingeniería asistida por computadora comercial realizado por *Dassault Systèmes*. El programa está desarrollado para proporcionar apoyo desde la concepción del diseño hasta la producción y el análisis de productos. Está disponible para *Microsoft Windows*, Solaris, IRIX y HP-UX. Provee una arquitectura abierta para el desarrollo de aplicaciones o para personalizar el programa. Las interfaces de programación de aplicaciones se pueden programar en Visual Basic y C++. Fue inicialmente desarrollado para servir en la industria aeronáutica, se ha hecho un gran hincapié en el manejo de superficies complejas (4).
  Catia también es ampliamente usada en la industria del automóvil para el diseño y desarrollo de componentes de carrocería. Concretamente empresas como el Grupo VW (Volkswagen, Audi, SEAT y Škoda), BMW, Renault, Peugeot, Daimler AG, Chrysler, Smart y Porsche hacen un amplio uso del programa. La industria de la construcción también ha incorporado el uso del *software* para desarrollar edificios de gran complejidad formal; el Museo Guggenheim Bilbao, en España, es un hito arquitectónico que ejemplifica el uso de esta tecnología (4).

- **ArchiCAD**

  ArchiCAD desarrollado por la empresa húngara Graphisoft que comenzó en 1982 originalmente para Macintosh. Es un *software* CAD de modelado de información de construcción (BIM, *Building Information Modeling*) disponible para sistemas operativos Macintosh y Windows. BIM es un paradigma del dibujo asistido por computador que permite un diseño basado en objetos inteligentes y en tercera dimensión. De este modo facilita el trabajo en tres dimensiones y dos dimensiones a la vez, pudiendo ahorrar tiempos en presentaciones y etapas del proyecto arquitectónico y permitiendo actualizar automáticamente toda la documentación de manera instantánea, sin la intervención del usuario para cambiar manualmente todas las vistas, planos y metrados (5).
  Fundamentalmente ArchiCAD, es un *software* por el cual se puede construir edificaciones, inicialmente en un ámbito de dos dimensiones, basándose en

parámetros básicos, tales como altura, largo, espesor y elevación. En el caso de muros, altura, ancho, largo y elevación en el caso de objetos, además el uso de este conlleva un conocimiento básico de términos usados en la arquitectura, puesto que los ámbitos más específicos (puertas, ventanas, escaleras, forjados y techos) se usan parámetros tales como: alfeizar, telar, paso, contrapaso, entre otros. La vista en plantas es la principal diferencia operacional respecto a otros programas de CAD analíticos como AutoCAD o Microstation, no es tomada desde una cámara en el infinito, sino que se radica en un corte paralelo al horizonte, este mismo tiene su inicio en una determinada altura y finaliza donde se ubica el forjado o base de piso (5).

ArchiCAD permite trabajar al usuario con representaciones dos dimensiones o tres dimensiones en pantalla. Los diseños en d*os* dimensiones pueden ser exportados en cualquier momento, incluso en el modelo; la base de datos siempre almacena los datos en tres dimensiones. Planos, alzados y secciones son generados desde el modelo del edificio virtual de tres dimensiones y son constantemente actualizados. Los diseños detallados están basados en porciones alargadas del modelo, con detalles en 2D añadidos (5).

Desarrolladores externos y algunos fabricantes para arquitectura han desarrollado bibliotecas de componentes arquitectónicos para usar en ArchiCAD, gracias a que el programa incluye un lenguaje de descripción geométrica (GDL) usado para crear nuevos componentes. La última apuesta de la compañía se centra en este aspecto, poniendo a disposición de los usuarios una web que funcionará como almacén online de objetos. Esto permite el intercambio de objetos desarrollados por distintos usuarios, así como por empresas de mobiliario, que se encarguen de elaborar versiones virtuales de sus productos (5).

- **Autodesk Inventor**

Autodesk Inventor es un paquete que se basa en técnicas de modelado paramétrico de sólidos en tres dimensiones producido por la empresa de *software* Autodesk. Compite con otros programas de diseño asistido por computadora como SolidWorks, Pro/ENGINEER, Catia y Solid Edge. Entró en el mercado en 1999, muchos años después que los antes mencionados y se agregó a las Series de Diseño Mecánico de Autodesk como una respuesta de la empresa a la creciente migración de su base de clientes de diseño mecánico en dos dimensiones hacia la

competencia, permitiendo que los computadoras personales ordinarias puedan construir y probar montajes de modelos extensos y complejos (6).

Los usuarios comienzan diseñando piezas que se pueden combinar en ensamblajes, y corrigiendo las mismas, pueden obtenerse diversas variantes. Como modelador paramétrico, no debe ser confundido con los programas tradicionales de CAD; Inventor se utiliza en diseño de ingeniería para producir y perfeccionar productos nuevos, mientras que en programas como AutoCAD se conducen solo las dimensiones. Un modelador paramétrico permite modelar la geometría, dimensión y material de manera que si se alteran las dimensiones, la geometría se actualiza automáticamente basándose en las nuevas dimensiones. Esto permite que el diseñador almacene sus conocimientos de cálculo dentro del modelo, a diferencia del modelado no paramétrico, que está más relacionado con un “tablero de bocetos digitales”. Inventor también tiene herramientas para la creación de piezas metálicas (6).

Los bloques de construcción cruciales de Inventor son las piezas, se crean definiendo las características, que a su vez se basan en bocetos (dibujos en dos dimensiones). Este sistema de modelado es mucho más intuitivo que en ambientes antiguos de modelado, en los que para cambiar dimensiones básicas era necesario generalmente suprimir el archivo entero y comenzar de cero (6).

Como parte final del proceso, las partes se conectan para hacer ensamblajes, los ensamblajes pueden consistir en piezas u otros ensamblajes. Las piezas son ensambladas agregando restricciones entre las superficies, bordes, planos, puntos y ejes. Por ejemplo, si uno coloca un piñón sobre un eje, una restricción insertada podría agregarse al eje y el piñón, haciendo que el centro del eje sea el centro del piñón. La distancia entre la superficie del piñón y del extremo del eje se puede también especificar con la restricción insertada. Otras restricciones incluyen Coincidencia, Nivelación, inserción, ángulo, tangente, transicional, movimiento, sistema de coordenadas de usuario (6).

Las últimas versiones de Inventor incluyen funcionalidades que poseían muchos modeladores de tres dimensiones de mediano y alto nivel, utiliza el Gestor de Formas (Shape Manager) como su kernel de modelaje geométrico, el cual pertenece a Autodesk y fue derivado del kernel de modelaje ACIS; además, incluye, en la versión profesional, las herramientas necesarias para crear piezas de plástico y sus respectivos moldes de inyección. Cuenta también con análisis de

tensiones por elementos finitos y análisis dinámicos, creación y análisis de estructuras, piping y cableado. Su combinación con Autodesk Vault y Autodesk 360 la hacen líder en el mercado del diseño mecánico (6).

- **SolidWorks**

SolidWorks es un *software* CAD para modelado mecánico en tres dimensiones, desarrollado en la actualidad por SolidWorks Corp., una filial de Dassault Systèmes, S.A. (Suresnes, Francia), para el sistema operativo Microsoft Windows. Su primera versión fue lanzada al mercado en 1995 con el propósito de hacer la tecnología CAD más accesible (7).

El programa 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 CAD. El proceso consiste en trasvasar la idea mental del diseñador al sistema CAD, "construyendo virtualmente" la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada (7).

- **Solid Edge**

Solid Edge es un programa parametrizado de diseño asistido por computadora de piezas tridimensionales, Presentado en 1996, inicialmente fue desarrollado por Intergraph como uno de los primeros entornos basados en CAD para Windows NT, ahora pertenece y es desarrollado por Siemens AG. Su kernel de modelado geométrico era originalmente ACIS, pero fue cambiado a Parasolid, el núcleo Parasolid es desarrollado actualmente por Siemens PLM *software* y es usado ampliamente como el motor geométrico de otras herramientas CAD (SolidWorks, IronCAD y MoldFlow). Recientemente adquirido por Siemens AG está empezando a formar parte de todas sus plantas de producción e ingeniería. Solid Edge permite el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros (8).

Este es uno de los paquetes instados a enterrar el uso masivo del CAD 2D dando paso al CAD 3D, con las consiguientes ventajas a todos los niveles del trabajo. A través de *software* de terceras partes, es compatible con otras tecnologías PLM (8).

- **Autodesk AutoCAD**

Autodesk AutoCAD es un *software* CAD utilizado para dibujo en dos dimensiones y modelado en tres dimensiones. Actualmente es desarrollado y comercializado por la empresa Autodesk (9).

El nombre AutoCAD surge como creación de la compañía Autodesk, en que Auto hace referencia a la empresa creadora del *software* y CAD a Diseño Asistido por Computadora (por sus siglas en inglés *Computer Aided Design*), teniendo su primera aparición en 1982. AutoCAD es un *software* reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos de edificios o la recreación de imágenes en tres dimensiones; es uno de los programas más usados por arquitectos, ingenieros, diseñadores industriales y otros (9).

1.3.2 Sistemas para el diseño asistido por computadoras con licencia 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 (10).

FreeCAD 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 (10).

A diferencia de los CAD analíticos tradicionales, como pueden ser AutoCAD o *Microstation*, FreeCAD es un CAD 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 (10).

Como muchos modernos modeladores CAD 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 polilinea, el arco de circunferencia, el círculo, el punto y la recta, además de módulos de parametrización y de mayado de entidades.

- **LibreCAD**

LibreCAD es una aplicación informática de código libre de diseño asistido por computadora (CAD) para diseño en dos dimensiones. Funciona en los sistemas operativos GNU/Linux, Mac OS X, Solaris y Microsoft Windows (11).

LibreCAD fue desarrollado a partir de un foro de *QCad Community Edition*. El desarrollo de LibreCAD está basado en las bibliotecas Qt4, pudiendo ser ejecutado en varias plataformas de manera idéntica (11).

Buena parte de la interfaz y de los conceptos sobre su uso son similares a los de AutoCAD, haciendo el uso de este más cómodo para usuarios con experiencia en ese tipo de programas CAD comerciales (11).

LibreCAD utiliza el formato del archivo de AutoCAD DXF internamente y para guardar e importar archivos, así como permite la exportación de estos en varios formatos. Al igual que en el *software* de diseño asistido por computadora FreeCAD**,** el LibreCAD posee funcionalidades como la polilinea, la recta, el circulo y el punto, las cuales han sido reutilizadas en el Proyecto “Sistema CAD 2D”.

1.4 Estudio de metodologías y estándares de desarrollo de software

Se entiende por metodología una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del *software* y, a su vez, los estándares de desarrollo por una estructura aplicada al desarrollo de un producto de *software* que contempla varios modelos a seguir para el establecimiento de un proceso para el desarrollo de *software*, cada uno de los cuales describe un enfoque diferente para diferentes actividades que tienen lugar durante el proceso. Algunos autores consideran un modelo de ciclo de vida, un término más general que un determinado proceso para el desarrollo de *software* (12).

1.4.1 Metodología Programación Extrema

La metodología XP es la más destacada de los procesos ágiles de desarrollo de *software* formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (12).

Posee varias características, las cuales son (13):

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas: frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera (el código es revisado y discutido mientras se escribe) es más importante que la posible pérdida de productividad inmediata.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- *Corrección* de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que en más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Como aspectos positivos y negativos se encuentran (13):
Ventajas:

- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas:

- Delimitar el alcance del proyecto con el cliente.

A pesar de los beneficios que se pueden obtener del uso de todas las prácticas en conjunto, la metodología no obliga, ni exige la aplicación estricta de todas ellas.
A partir del análisis realizado en el contexto del proyecto “Sistema para el dibujo técnico mecánico en dos dimensiones, asistido por computadora (Sistema CAD 2D)” se puede concluir que se tomará de XP**,** exclusivamente, los principios de programación, las pruebas ATDD (*Acept Test Driver Development*) y la estandarización de código.

1.4.2 Metodología Scrum

En Scrum se aplican de manera regular un conjunto de mejores prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos (13).
Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de *software*. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de *software* ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una

duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de *software* como RUP o XP (12).

Scrum se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar *software* que realmente resuelva las necesidades, aumentando la satisfacción del cliente (12).

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales (13).

Como ventajas se encuentran (13):

- Entrega mensual (o quincenal) de resultados (los requisitos más prioritarios en ese momento, ya completados) lo cual proporciona las siguientes ventajas:
  - Gestión regular de las expectativas del cliente y basadas en resultados tangibles.
  - Resultados anticipados.
  - Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.
  - Gestión sistemática del Retorno de Inversión (ROI).
  - Mitigación sistemática de los riesgos del proyecto.
- Productividad y calidad.
- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado.

En el contexto del proyecto “Sistema para el dibujo técnico mecánico en dos dimensiones, asistido por computadora (Sistema CAD 2D)” se decidió que lo más apropiado para complementar los aspectos a utilizar de XP**,** era tomar de Scrum la Pila de “Sprint” y la Pila de Producto.

1.4.3 Lenguaje de modelado unificado

Lenguaje de modelado unificado (UML) *"…es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software."* (14).

El UML proporciona a los desarrolladores un vocabulario que incluye tres categorías: elementos, relaciones y diagramas (14).

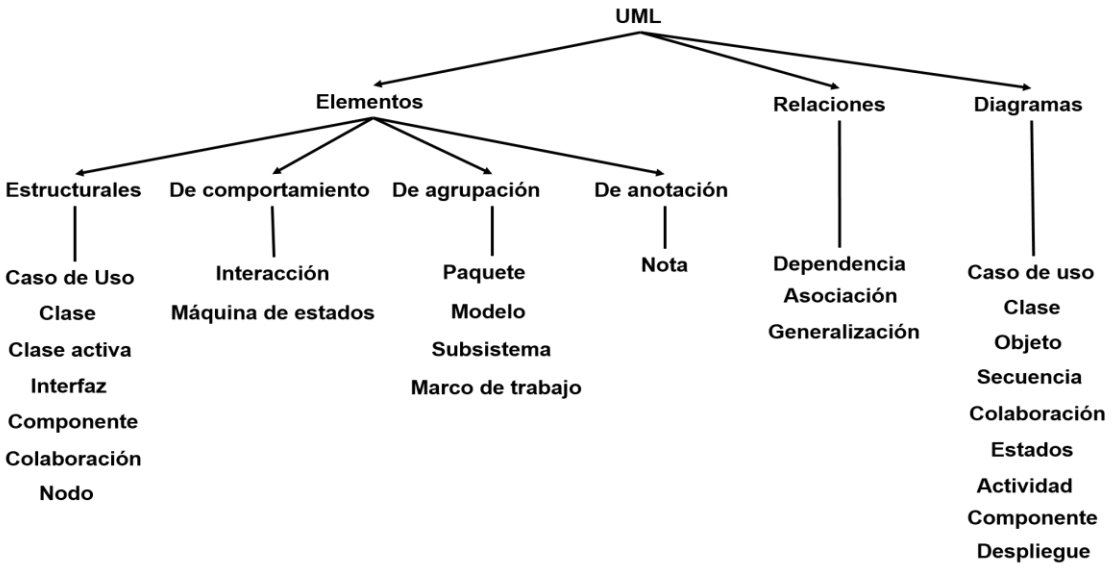


Figura 1: Vocabulario de UML *(14)*.

El UML está pensado para ser empleado en herramientas interactivas de modelado visual que tengan generadores de código y/o generadores de informes. La especificación de UML no define un proceso estándar pero está ideado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (14).

La capacidad de UML para modelar cualquier sistema de *software* y el hecho de que haya sido adoptado por el Grupo de Gestión de Objetos como un estándar desde noviembre de 1997, permitieron determinarlo como el lenguaje de modelado para desarrollar la propuesta de solución (14).

Se escogió como lenguaje de modelado UML porque es posible establecer una serie de requerimientos y estructuras necesarias para modelar un sistema de *software* previo al proceso intensivo de escribir código. UML es un lenguaje que posee más características

visuales que programáticas, las que facilitan a integrantes de un equipo participar e intercomunicarse fácilmente, siendo estos integrantes los analistas, diseñadores, especialistas de área y desde luego los programadores. Una de las características más importantes que hacen que se escoja UML como lenguaje de modelado es que está diseñado para uso con *software* orientado a objetos y tiene un uso limitado en otro tipo de cuestiones de programación.

1.4.4 Herramienta para el modelado. Visual Paradigm v8.0

Las herramientas para el modelado son un conjunto de programas y ayudas que brindan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software* (Investigación preliminar, Análisis, Diseño, Implementación e Instalación).

Existen varias herramientas para el modelado entre las que se encuentran Rational Rose, ArgoUML y Visual Paradigm. En el desarrollo de esta investigación será utilizada Visual Paradigm para crear los diagramas que serán elaborados en cada flujo de trabajo propuesto por la metodología y los prototipos de interfaces que se utilizarán.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas. Puede ser utilizado en diferentes plataformas lo que es una ventaja debido a que el componente será desarrollado en Ubuntu. Permite realizar ingeniería directa e inversa y todo tipo de diagramas.

1.5 Estudio de estilo y patrón arquitectónico para el desarrollo de software

Los patrones arquitectónicos o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de *software* en Ingeniería de *software*. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de *software*, que consta de subsistemas, sus responsabilidades e interrelaciones; mientras que los Estilos Arquitectónicos describen una categoría del sistema que contiene: un conjunto de componentes que realiza una función requerida por el sistema, un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se puede integrar los componentes que forman el sistema;

y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes (15).

1.5.1 Estilo de llamada y retorno

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

- **Arquitecturas en capas**

Por decisión de proyecto se seleccionó el patrón arquitectónico **por capas**, debido a que el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario se separan en las capas necesarias, siendo esto lo requerido por el componente a desarrollar.

Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados como categorías mayores del catálogo, o, por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. El estilo en capas se definen como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Instrumentan así una vieja idea de organización estratigráfica. En algunos ejemplares, las capas internas están ocultas a todas las demás, menos para las capas externas adyacentes, y excepto para funciones puntuales de exportación; en estos sistemas, los componentes implementan máquinas virtuales en alguna de las capas de la jerarquía. En otros sistemas, las capas pueden ser sólo parcialmente opacas.

En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. El uso de arquitecturas en capas, explícitas o implícitas, es frecuentísimo; hay cerca de cien patrones que son variantes del patrón básico de capas (15).

En un estilo en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción. Los diagramas de sistemas clásicos en capas dibujaban las capas en adyacencia, sin conectores, flechas ni interfaces; en algunos casos se suele representar la naturaleza jerárquica del sistema en forma de círculos concéntricos.

Las restricciones topológicas del estilo pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma; se supone que si esta exigencia se relaja, el estilo deja de ser puro y pierde algo de su capacidad heurística;

también se pierde, naturalmente, la posibilidad de reemplazar una capa sin afectar a las restantes, disminuye la flexibilidad del conjunto y se complica su mantenimiento.

Las formas más rígidas no admiten ni siquiera *pass-through*: cada capa debe hacer algo, siempre. En la literatura especializada hay multitud de argumentos a favor y en contra del rigor de esta clase de prescripciones. A veces se argumenta que el cruce superfluo de muchos niveles involucra eventuales degradaciones de performance; pero muchas más veces se sacrifica la pureza de la arquitectura en capas precisamente para mejorarla: colocando, por ejemplo, reglas de negocios en los procedimientos almacenados de las bases de datos, o articulando instrucciones de consulta en la capa de la interface del usuario (15).

1.6 Tendencias y tecnologías actuales para el desarrollo de software. Selección de las herramientas y lenguajes de desarrollo

Las tecnologías están presentes en todos los procesos de la sociedad y de la economía del ser humano. Su principal propósito es transformar el entorno, para adaptarlo mejor a las necesidades y deseos del hombre. Las TIC son aquellas herramientas computacionales e informáticas que procesan, almacenan, sintetizan, recuperan y presentan información en la más variada forma; el desarrollo alcanzado por estas tecnologías, trae consigo un continuo perfeccionamiento de las herramientas informáticas, por tanto, al implementar cualquier tipo de *software*, se hace necesario realizar un estudio detallado en relación a cualquier recurso a utilizar. Seguidamente se describen las tecnologías empleadas en la presente investigación, realizando un análisis de las principales características, ventajas y desventajas de cada una de ellas, con el fin de realizar una mejor selección que pueda sostener el desarrollo de la solución propuesta.

1.6.1 Lenguaje C++

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[5]. C++ es

[5] Un Lenguaje de programación multiparadigma es el cual soporta más de un paradigma de programación. Permiten crear programas usando más de un estilo de programación.

un lenguaje de programación maduro y de gran velocidad de compilación y presenta las siguientes características (16):

- 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. Posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel (16):

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

C++ fue el lenguaje seleccionado porque además de que lo utiliza el *framework* seleccionado (Qt), posee características que lo realzan de los demás lenguajes. Existen varias bibliotecas con algoritmos desarrollados en C++, simplemente se toman y se adaptan a la solución. También se seleccionó por las ventajas que brinda la programación orientada a objetos y por el conocimiento y la experiencia adquiridos con este lenguaje.

1.6.2 Doxygen

Doxygen es un generador de documentación para C++, C, Java, Python, entre muchos otros. Es multiplataforma, funciona en Linux, Windows y Mac OS X. El mismo es utilizado para gestionar la documentación de los métodos o clases que puedan surgir a partir del desarrollo de la aplicación. Genera un buscador de documentación en línea y/o un manual de referencia de un conjunto de archivos fuente documentados. También tiene soporte para archivos HTML, PDF con hipervínculos, archivos RTF de Word y otros. La documentación es extraída directamente de la fuente, lo cual hace mucho más fácil mantener la documentación consistente y actualizada con la fuente. Doxygen es desarrollada bajo Mac OS X y Linux, pero está hecho para ser altamente portátil (17).

1.6.3 Framework Qt

Qt es un *framework* de desarrollo para aplicaciones multiplataforma que simplifica mucho el desarrollo de aplicaciones en C++ de forma nativa, también puede ser utilizado en otros lenguajes, funciona en las principales plataformas 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 (19).

Este *framework* [6]está compuesto por bibliotecas Qt (clases en C++), además se pueden crear formularios visualmente con *Qt Designer* y presenta un acceso rápido a la documentación a través de *Qt Assistant*. Permite una traducción rápida con su *Qt Linguist* y simplifica el proceso de construcción de proyectos en las diferentes plataformas soportadas. La Interfaz de Programación de Aplicaciones(API) de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML; cuenta con una API multiplataforma unificada para la manipulación de archivos y otras para el manejo de ficheros, además de estructuras de datos tradicionales (19).

Otras características:

- QT constituye además una biblioteca para la creación de interfaces gráficas. Se distribuye bajo una licencia libre GPL (o QPL), además de la LGPL, que permite incorporar QT en una aplicación open-source.
- Se encuentra disponible para una gran número de plataformas: Linux, MacOs X, Solaris, HP-UX, UNIX con X11, Windows.
- Es orientado a objetos y el lenguaje para el que se encuentra disponible es C++.
- Es una biblioteca que se basa en los conceptos de *widgets* (objetos), Señales-Slots y Eventos (Ej.: clic del ratón).
- Las señales y los *slots* es el mecanismo para que unos *widgets* se comuniquen con otros.
- Los *widgets* pueden contener cualquier número de hijos. El *widget* superior (también conocido como *top-level*), puede ser cualquiera, sea ventana, botón, etc.
- Compatibilidad multiplataforma con un sólo código fuente.
- Fácil de internacionalizar.

[6] El *framework* en el desarrollo de *software*, es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado.

- Arquitectura lista para plugins.

Con todo lo anterior expuesto se puede decir que Qt dispone de tres grandes ventajas ante otras librerías rivales (19):

- Qt es completamente gratuito para aplicaciones de código abierto, y una de las licencias mencionadas anteriormente, la LGPL, permite que se utilice gratuitamente con fines comerciales.
- Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (MacOs X y Solaris) como también para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código y la aplicación se verá y actuará mejor que una aplicación nativa.
- Qt tiene una extensa librería con clases y herramientas para la creación de aplicaciones con interfaz de usuario enriquecidas. Estas librerías y clases están bien documentadas, son muy fáciles de usar y tienen una gran herencia de programación orientada a objetos lo cual hace de la programación de interfaces gráficas un trabajo más ameno.

A continuación otras de las características más importantes de este *framework* que ayudan a la selección del mismo (19):

- Compatibilidad multiplataforma con un sólo código fuente.
- Rendimiento en C++.
- Disponibilidad del código fuente.
- Excelente documentación.
- Arquitectura lista para plugins.

Con este *framework* se pueden crear de forma rápida y sencilla interfaces gráficas de usuario; tiene disponible como lenguaje C++, siendo este el lenguaje seleccionado para la implementación del componente. También el uso de esta biblioteca ahorra una enorme cantidad de tiempo y esfuerzo, pues de lo contrario se tendrían que desarrollar algunos componentes desde cero. Estas son las principales características que debe tener la aplicación que se desarrolla, lo cual dio paso a la selección del mismo para la realización de la aplicación de este trabajo.

1.6.4 OpenCascade y Ocaf

Dentro de las variantes que han surgido para la implementación y trabajo con las tecnologías **CAD/CAE/CAM** en código libre, está la tecnología *OpenCascade* (1).

OpenCascade (abreviatura de las siglas en inglés *Computer Aided Software for Computer Aided Design and Engineering*), desarrollada inicialmente a principios de los años 90 por "Matra Datavision", es una aplicación orientada a usuarios que se desenvuelven dentro del área del diseño, como ingenieros, arquitectos y diseñadores. Gracias a sus bondades es posible diseñar y modificar cualquier modelo bidimensional o tridimensional deseado (1).

Cuenta con una licencia de funcionamiento GPL, lo que permite utilizarla de forma ilimitada en cualquier ordenador. Se comporta como un entorno de desarrollo integrado, orientado a la edición y el diseño de superficies y objetos bidimensionales y tridimensionales, utilizando para su diseño una gran cantidad de opciones de personalización que facilitan enormemente el modelado de estas figuras u objetos (1).

Cuenta con la capacidad de realizar análisis gráficos de cualquier superficie, lo que se traduce en estimar y simular los datos generados por las curvas de los gráficos de forma numérica y rápida, para facilitar los estudios de ingeniería; además de la posibilidad de importar y exportar figuras, viene con un paquete completo de efectos especiales y texturas que se pueden añadir a los diseños de modo que se puedan personalizar por completo (1).

La manera más sencilla de trabajar con *OpenCascade* es a través de su *framework* OCAF[7], que por sus siglas en inglés. OCAF es una plataforma fácil de usar y que permite el desarrollo rápido de aplicaciones sofisticadas de diseño. Una aplicación típica desarrollada con OCAF puede tener un modelador geométrico en dos o tres dimensiones, herramientas de fabricación o de análisis, y aplicaciones de simulación o herramientas de ilustración. Cuenta con un sinnúmero de características que la convierten en una opción atractiva a la hora de hacer una selección de tecnología para el trabajo con **CAD/CAE/CAM**, y que justifican su selección en este caso, como son (1):

- Facilidad de diseño de la arquitectura de la aplicación.
- Definición de los componentes y la forma en que cooperan.
- Definición del modelo de datos capaz de soportar las funcionalidades requeridas.
- Estructuración del *software* en relación a:

[7] Open CASCADE Application *Framework*.

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

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

1.6.5 Acis y Parasolid

Parasolid es un núcleo de modelado geométrico desarrollado originalmente por ShapeData, ahora propiedad de Siemens PLM *Software* (anteriormente UGS Corp.).

Las capacidades de Parasolid incluyen utilidades de creación y edición de modelos como operadores booleanos, característica de soporte de modelado, superficie avanzada, engrosamiento y vaciamiento, la mezcla y el fileteado y el modelado de hoja.

Para utilizar Parasolid efectivamente, los usuarios necesitan tener conocimientos fundamentales de geometría computacional y su topología.

Parasolid incluye líneas ocultas, teselación y consultas de datos de modelos. Cuando se exporta desde el paquete de *software* de los padres. La mayoría de los archivos de Parasolid pueden comunicarse y migrar sólo sólidos en tres dimensiones y / o datos de superficie, Parasolid actualmente no puede comunicar y migrar datos en dos dimensiones tales como líneas y arcos (20).

La *ACIS 3D Modeler* (ACIS) es un núcleo de modelado geométrico desarrollado por Spatial Corporation, parte de *Dassault Systèmes*. ACID es utilizado por muchos desarrolladores de *software* en industrias tales como el diseño asistido por ordenador (CAD), fabricación asistida por ordenador (CAM), ingeniería asistida por ordenador (CAE), Arquitectura, ingeniería y construcción (AEC), Máquina de medición por coordenadas (CMM), animación en tres dimensiones, y la construcción naval. ACIS cuenta con una arquitectura orientada a objetos C ++ abierta, que permite, capacidades de modelado en

tres dimensiones robustos (21).

ACIS se utiliza para construir aplicaciones con características de modelado híbrido, ya que integra modelo de estructura metálica, la superficie y la funcionalidad de modelado de sólidos, y un rico conjunto de operaciones geométricas. Como núcleo geométrico, **ACIS** es un sistema de segunda generación (21).

La línea de productos **ACIS** se ha diseñado utilizando la tecnología de componentes de *software*, lo que permite que una aplicación utilice sólo los componentes que requiere. En algunos casos, más de un componente está disponible para un fin determinado, por lo que los desarrolladores de aplicaciones pueden utilizar el componente que mejor se adapte a sus necesidades.

1.7 Conclusiones del capítulo

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

Capítulo II: Propuesta de solución

En el presente capítulo se expone la propuesta de solución que incluye el modelo de domino, la descripción del componente, la etapa de planificación con las características del sistema y los aspectos funcionales los cuales se agrupan en historias de usuario y la etapa de diseño con los patrones arquitectónicos, los prototipos de interfaces y la metodología de cálculo para engranajes cilíndricos de dientes rectos.

2.1 Modelo de dominio

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 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 (22).

Por exigencias del proyecto se estimó necesario generar este artefacto para un mejor entendimiento de la propuesta de solución a desarrollar.

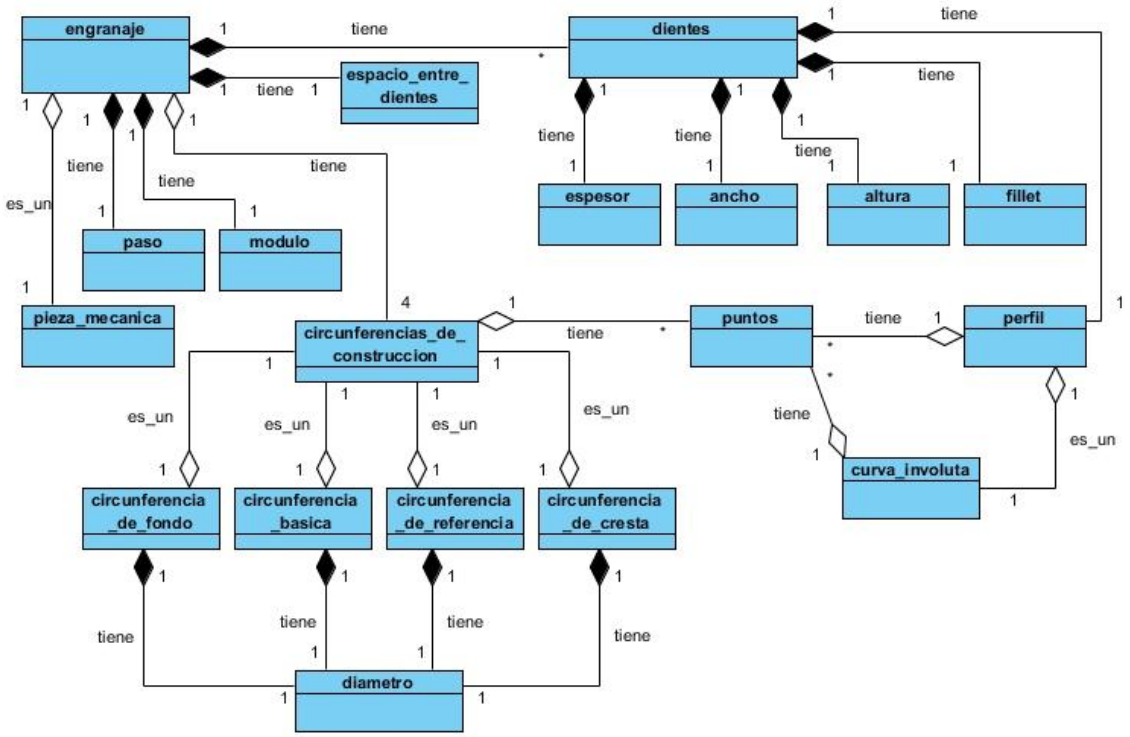


Fig. 1 Diagrama del Modelo de Dominio.

En el diagrama anterior 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, la cual se genera por el posicionamiento de puntos en el plano que se obtienen de dos ecuaciones paramétricas que se mostraran más adelante.

2.2 Descripción del componente

El componente elaborado permite la modelación de engranajes cilíndricos de dientes rectos con perfil de Involuta para ser insertado en cualquier sistema CAD compatible con *OpenCascade*; la cual cuenta con varias características que serán explicadas en los párrafos que siguen.

El componente muestra una interfaz que posee dos pestañas en la parte superior izquierda llamadas *Design* y *Calculation*. En *Design* se encuentran los datos y parámetros relativos al diseño de engranajes; el usuario puede seleccionar los valores de los datos, de listas desplegables en el caso del ángulo de presión (con valores entre $14{,}5^{o}$ y 30^{o}) el módulo (con un rango de datos entre 0,050 mm hasta 100 mm) la relación de transmisión (con un rango de datos entre 1.000 ul y 40.000 ul) y la guía de diseño que es la opción que se encarga de habilitar los campos para rectificar ciertos parámetros según las normas establecidas. También posee campos para introducir los valores deseados como el número de dientes del piñón y de la rueda secundaria, o las correcciones de unidades de cada rueda y las totales.

La interfaz posee varios botones de acción, el de cancelar con la designación *Cancel* permite salir del componente de construcción de engranajes al instante, el de calcular designado como *Calculate* activa los cálculos correspondientes, si el engranaje no está correctamente calculado, en una pequeña área blanca que esta insertada en la interfaz se lanza un mensaje de error en letras rojas que deberá acercar al usuario al error de construcción; cuando el cálculo del engranaje está correcto, se muestra un mensaje en letras azules en la misma área blanca, y a su vez se activan dos botones de acción, el botón *Preview* y el botón *OK*, *Preview* abre otra ventana donde se muestran todos los

datos calculados de ambas ruedas antes de realizar el diseño correspondiente, mientras que *OK* cierra la interfaz de construcción de engranajes y muestra, en tres dimensiones, el diseño del engranaje en un grid, el cual es la interfaz principal del proyecto CAD2D.

En la pestaña *Calculation* se encuentran todos los datos relacionados al cálculo físico del engranaje. Existe una lista desplegable con varios métodos de cálculo de fuerza según normas diferentes la cual, según su selección, utilizando datos como la velocidad angular, el torque, la vida útil del engranaje, potencia, valores de distintos materiales, tipos de materiales, elasticidad, entre otras propiedades físicas que el usuario puede definir numéricamente, se realiza el cálculo adecuado y se muestra en el área de resultados de la derecha luego de pulsar el botón Calcular, mostrando una vista de todos los valores físicos del engranaje antes de ser visualizado, justo como en la pestaña anterior.

2.3 Planificación

La planificación se basó en un análisis detallado del trabajo a realizar y su descomposición en tareas. Parte por tanto de un proyecto de obra, o de unos requisitos detallados de lo que se quiere hacer. Sobre esa información se desarrolla un plan adecuado a los recursos y tiempos disponibles; y durante la construcción se sigue de cerca la ejecución para detectar posibles desviaciones y tomar medidas para mantener el plan, o determinar qué cambios va a experimentar. Se trata por tanto de una gestión “predictiva”, que vaticina a través del plan inicial cuáles van a ser la secuencia de operaciones de todo el proyecto, su coste y tiempos.

Su principal objetivo es conseguir que el producto final se obtenga según lo “previsto”; y basa el éxito del proyecto en los tres puntos apuntados: agendas, costes y calidad.

2.3.1 Requerimientos del componente

Un requerimiento de *software* constituye una restricción que se determina con alta precisión, y debe ser satisfecha por el producto final. Existen dos tipos de requerimientos, los aspectos funcionales y las características del sistema.

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

Las características del sistema permiten definir las cualidades que el *software* debe tener con el objetivo de brindar a los usuarios una mayor usabilidad, disponibilidad y rendimiento.

2.3.1.1 Aspectos funcionales

A continuación se describen los aspectos funcionales del componente.

1 Especificar parámetros de entrada.

1.1 El usuario deberá introducir el valor del número de dientes del piñón.

1.2 El usuario deberá introducir el valor del número de dientes de la rueda.

1.3 El usuario deberá introducir el valor del módulo para el sistema de engranajes.

1.4 El usuario deberá introducir el valor de la relación de transmisión para el sistema de engranajes.

1.5 El usuario deberá introducir el valor del ángulo de presión.

1.6 El usuario deberá introducir el valor del ancho de la cara del piñón.

1.7 El usuario deberá introducir el valor del ancho de la cara de la rueda.

1.8 El usuario deberá introducir el valor de las correcciones de unidades del piñón.

1.9 El usuario deberá introducir el valor de las correcciones de unidades de la rueda.

2. Realizar cálculo del engranaje.

2.1 Realizar el cálculo de las circunferencias necesarias para la construcción del engranaje.

2.2 Realizar el cálculo de la curva de involuta para el perfil del diente.

2.3 Realizar el cálculo de la distancia entre centros.

2.4 Realizar el cálculo de espaciado entre dientes.

2.5 Realizar el cálculo del ancho de los dientes.

3. Visualizar entidades.

3.1 Se debe visualizar el perfil del diente.

3.2 Modelación del núcleo de la rueda.

3.3 Modelación del núcleo del piñón.

3.4 Visualizar el ensamble del piñón con la rueda en 3 dimensiones.

4. Comprobar resultados del cálculo del engranaje.

4.1 El piñón y la rueda deben quedar ensamblados.

4.2 Las correcciones de unidades de la rueda y el piñón deben coincidir con la corrección de unidades total una vez que se sumen.

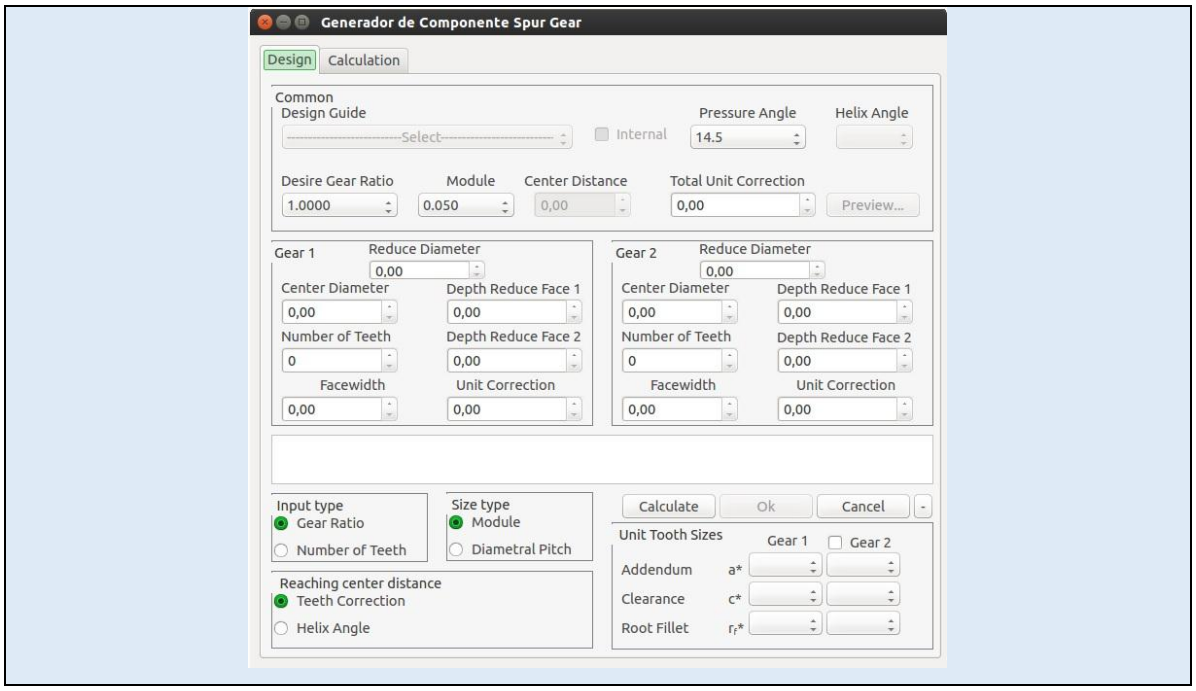
4.3 El componente debe poder montarse en Salome y en freecad.

2.3.1.1.1 Historias de usuario

Dentro de la fase de planificación, se realizan las historias de usuario, estas tienen el mismo propósito que los casos de uso. Las mismas son escritas por los propios clientes, tal y como ellos visualicen las necesidades del sistema. Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán el proceso de creación de los *test* de aceptación que son empleados para verificar que las historias de usuarios han sido implementadas correctamente (13).

Tabla 1: Historia de Usuario Realizar cálculo del engranaje.

Historia de usuario	
Número: 2	**Nombre del aspecto:** Realizar cálculo del engranaje.
Programador: Sandy García Santos	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 25 días
Riesgo en Desarrollo: Alto	**Tiempo Real:** 5 semanas
Descripción: El sistema realizará los cálculos adecuados para determinar los puntos de la curva de involuta, la distancia entre centros del piñón y la rueda secundaria y el ancho de los dientes del piñón y de la rueda secundaria; así como la creación de construcción.	
Observaciones:	
Prototipo de interfaz:	



2.3.1.2 Características del sistema

- **Software**

1. Tener instalada cualquier distribución de Ubuntu que sea de 64 bits.
2. Tener en el equipo la biblioteca Oce (*OpenCascade Community Edition*).

- **Hardware**

3. La estación de trabajo deberá tener como mínimo 1 GB dedicado a video.

- **Diseño e implementación**

5. El lenguaje de programación utilizado será C++.
6. QtCreator como entorno de desarrollo.
7. *Framework* Qt.
8. El perfil del diente está determinado por una curva de Involuta.
9. La unidad de medida utilizada es el metro, específicamente milímetro.

- **Apariencia o interfaz externa**

10. Debe existir un botón llamado *Preview* en el cual se muestren todos los datos del cálculo del engranaje antes de ser visualizado.
11. Debe existir un botón llamado *OK,* el cual generará una vista del engranaje
12. Debe existir un campo en el cual se muestren notificaciones al usuario.

13. Los botones *OK* Y *Preview* deberán permanecer bloqueados hasta que no se calcule el engranaje.
14. Cuando se seleccione una opción en el campo Design Guide, según la norma mecánica en cada caso, se bloquean y se desbloquean ciertos campos de la interfaz principal que estarán en relación con dicha norma.

- **Usabilidad**

15. El sistema podrá ser utilizado por cualquier persona que posea conocimientos básicos de mecánica.
16. Se podrá cancelar la operación desde la interfaz principal.

2.4 Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida. En el diseño se modeló el sistema y se encontró su forma para que soporte todos los requerimientos o aspectos, incluyendo los no funcionales y las restricciones que se le suponen.
Una entrada esencial en el diseño es el resultado de la planificación, que proporciona una comprensión detallada de los requisitos o aspectos. Además, impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se le da forma al componente (22).

El diseño crea una entrada apropiada y un punto de partida para actividades de implementación y se es capaz de descomponer dichas actividades en partes más manejables, que pueden ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.

2.4.1 Patrones de diseño

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

- **Experto:** mediante su uso, se asignan responsabilidades a la clase que cuenta con la información necesaria. Se evidencia en la clase: make_spur_gear.cpp.
- **Creador:** permite crear objetos de una clase determinada. Es utilizado en algunas de las clases controladoras para crear instancias de formularios y entidades.
- **Controlador:** se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. Este patrón se evidencia en las clases qoccharnesswindow.cpp, qoccapplication.cpp y qoccinputoutput.cpp.

2.4.2 Prototipos de interfaces de usuario

El diseño de la interfaz de usuario se puede definir como: *"el conjunto de pasos que seguirá el usuario, durante todo el tiempo que interactúe con el componente, detallando lo que verá en cada momento, y las acciones que realizará, así como las respuestas que el sistema dará"* (22).

A continuación se muestran los prototipos de interfaz del componente:

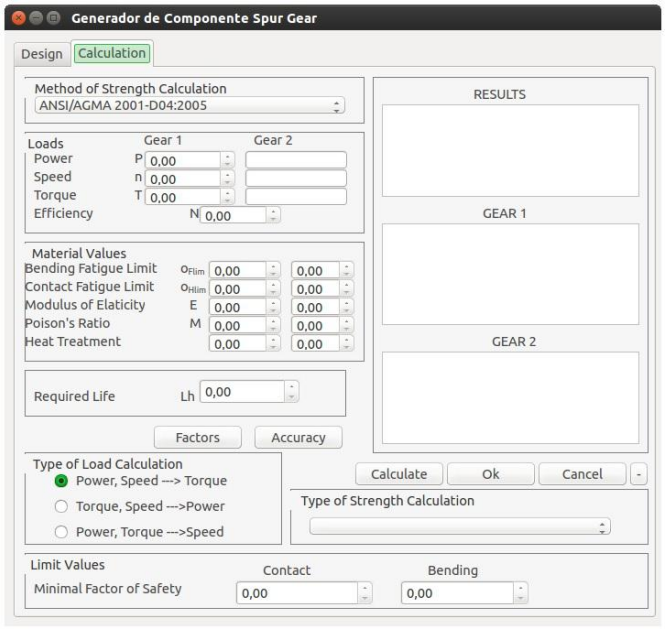


Figura 2: En la pestaña Calculation de la ventana principal es donde se recogen todos los datos asociados al cálculo físico del engranaje.

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

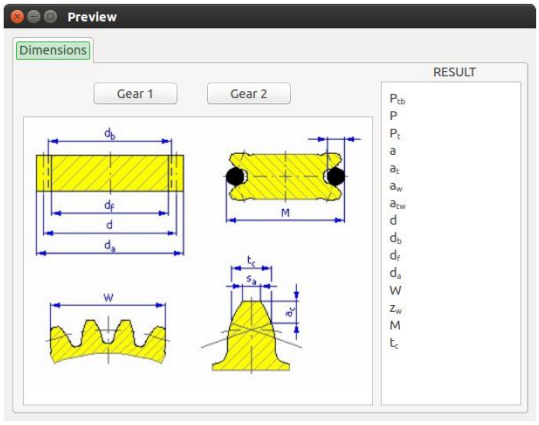


Figura 3: La ventana Preview muestra valores del cálculo del diseño del engranaje antes de ser visualizado.

La ventana "*Preview*" es la encargada de mostrar los datos del cálculo del diseño del engranaje antes de ser visualizado, posee dos botones los cuales están asociados a ambas ruedas, y una vez pulsados, se muestra en el área blanca de la derecha los valores del cálculo en cada campo respectivamente.

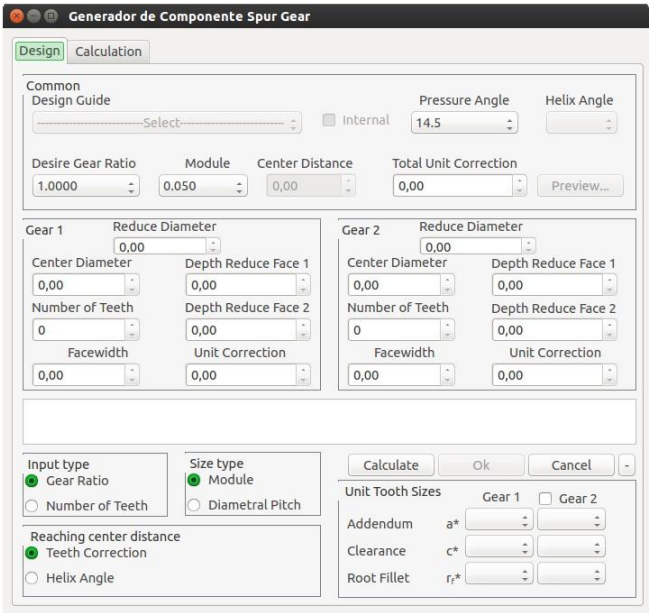


Figura 4: En la pestaña Design de la ventana principal es donde se recogen todos los datos asociados al cálculo del diseño del engranaje.

En la ventana “Design” se muestran los parámetros necesarios para el cálculo del diseño del engranaje, así como la “Design Guide” o guía de diseño la cual se encarga de determinar que parámetros están activos en cada caso y cuales no; una vez pulsado el botón “Calcular”, se lanza un mensaje en el área blanca en la parte inferior de la ventana, que de acuerdo al tipo de mensaje, está disponible el acceso a la ventana de “Preview” y a la visualización mediante le botón “OK”.

2.4.3 Construcción de un engranaje cilíndrico de dientes rectos. Metodología de cálculo

Los engranajes cilíndricos de dientes rectos sirven para transmitir movimiento circular o lineal (caso de cremalleras) entre dos ejes paralelos. Es una forma de mejorar la rotación entre dos cilindros que tienen sus caras en contacto y que, por lógica, se producirían deslizamientos.

- **Creando circunferencias de construcción.**

Para crear un engranaje se necesitan cuatro circunferencias de construcción:

- **Circunferencia de Referencia**

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

$$d = zm \tag{1}$$

Dónde:

d.- diámetro de la circunferencia de referencia.

z.- número de dientes.

m.- módulo.

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

$$da = d + (2m) \tag{2}$$

- **Circunferencia básica**

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

$$db = d \cos \alpha \quad (3)$$

Donde:

α.- ángulo de presión.

- **Circunferencia de fondo**

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

$$df = d - (2m) \quad (4)$$

- **Creando un diente de un engranaje cilíndrico de dientes rectos**

Para crear un diente de un engranaje cilíndrico de dientes rectos hay que tener en cuenta varios elementos:

- **Perfil del diente**

Hay cinco formas de construir el perfil de un diente de un engranaje, las cuales son las siguientes:

1) Mediante la curva "*Cicloide*":

Se construye situando un punto fijo P_0 en la circunferencia llamada *ruleta*, la cual rueda por la línea llamada *base,* a cada paso el punto fijado P_0 se encuentra en una nueva posición que luego de ser unido con los puntos anteriores crea una nueva curva llamada Cicloide.

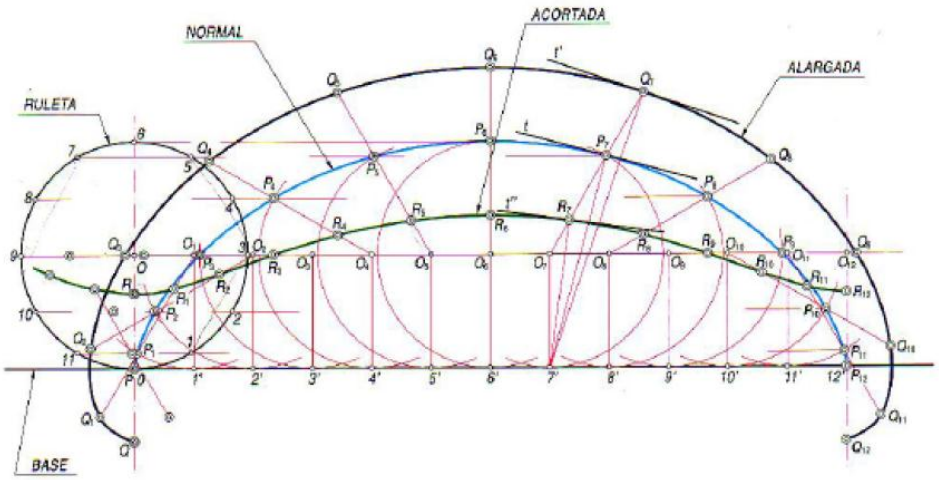


Figura 5: La trayectoria azul muestra la Cicloide.

2) Mediante la curva “Epicicloide”

Se construye situando un punto fijo P_0 en la circunferencia llamada *ruleta*, la cual rueda por la curva llamada *base,* a cada paso el punto fijado P_0 se encuentra en una nueva posición que luego de ser unido con los puntos anteriores crea una nueva curva llamada Epicicloide.

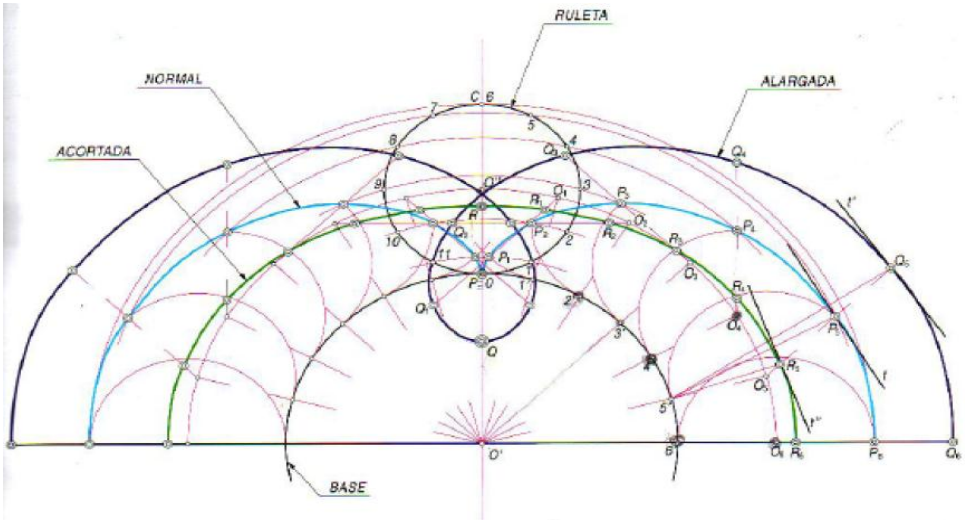


Figura 6: La trayectoria azul muestra la Epicicloide.

3) Mediante la curva “Hipocicloide”

Se construye situando un punto fijo P_0 en la circunferencia llamada *ruleta*, la cual rueda por la curva llamada *base,* a cada paso el punto fijado P_0 se encuentra en una nueva posición que luego de ser unido con los puntos anteriores crea una nueva curva llamada Hipocicloide.

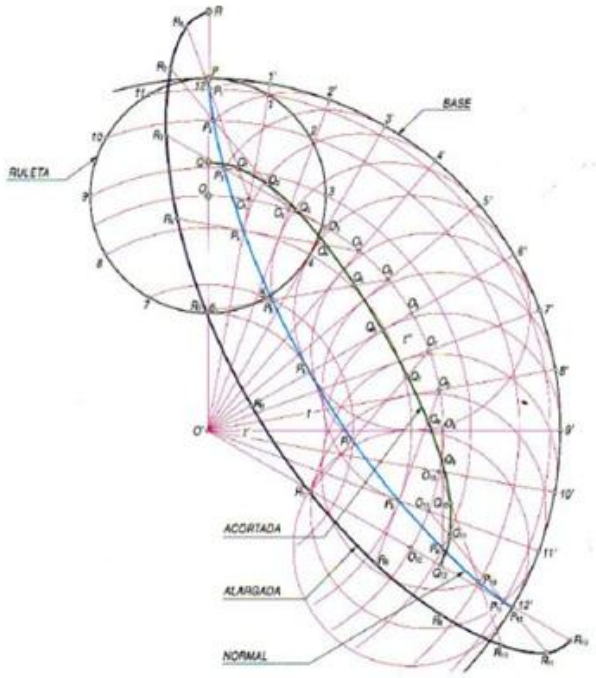


Figura 7: La trayectoria azul muestra la Hipocicloide.

4) Mediante la curva “Pericicloide”

Se construye situando un punto fijo P_0 en la curva llamada *ruleta*, la cual rueda por la circunferencia llamada *base,* a cada paso el punto fijado P_0 se encuentra en una nueva posición que luego de ser unido con los puntos anteriores crea una nueva curva llamada Pericicloide.

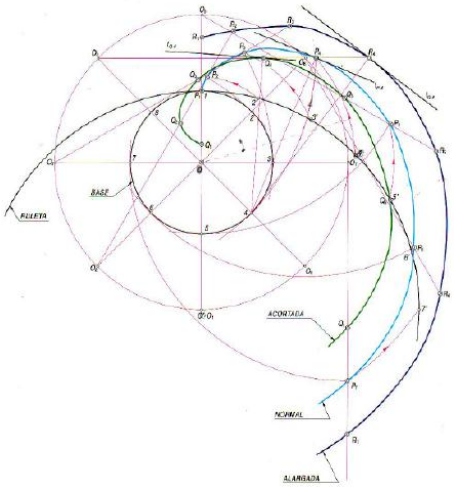


Figura 8: La trayectoria azul muestra la Pericicloide.

5) Mediante la curva “Involuta”

La envolvente de circulo o Involuta es la curva descrita por cualquier punto de una recta que rueda sin resbalar sobre una circunferencia de radio r (construcción del hilo tenso que se desenrolla de un tambor cilíndrico)

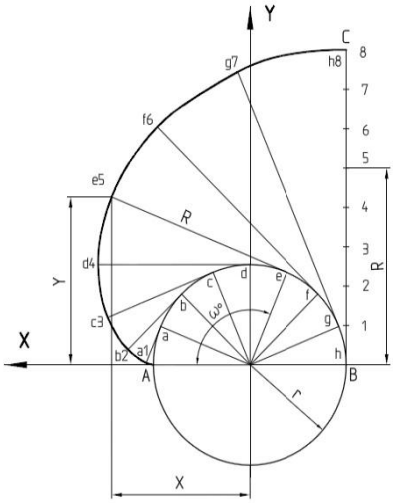


Figura 9: Trayectoria de la envolvente de circulo o Involuta.

De todas estas formas de crear perfiles de dientes de engranajes, se decidió utilizar la curva de Involuta debido a que es la curva más utilizada para este fin actualmente en todos los sistemas estudiados en el capítulo anterior.

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

$$x = \left[\frac{d_b}{2}\right] * [\cos\alpha + (\alpha * \sin\alpha)] \quad (5)$$

$$y = \left[\frac{d_b}{2}\right] * [\sin\alpha - (\alpha * \cos\alpha)] \quad (6)$$

Donde d_b es el diámetro de la circunferencia base y α es el ángulo del punto que genera la tangente en cada caso. El ángulo se va incrementando hasta llegar al valor deseado, cuando $\alpha > 90^o$ no es necesario seguir incrementando el ángulo.

Luego de tener creados todos los puntos, se unen mediante segmentos para formar la curva Involuta y dicha curva se une a la circunferencia básica en caso de tener mayor diámetro que la circunferencia de fondo, de no ser así, la curva Involuta se une a la circunferencia de fondo como anteriormente se mencionó.

- **Construyendo el diente completo**

Luego de tener trazado el perfil principal del diente, se necesita calcular el perfil del diente por el otro lado, es decir la imagen de la curva de Involuta, lo cual se puede lograr aplicando una simetría al perfil original para que gire sobre un eje determinado que es radial a la circunferencias de construcción.

El paso del diente es el espacio entre dos perfiles de "involuta" de dos dientes contiguos como se aprecia en la figura 2 y se calcula de la siguiente manera:

$$p = \pi * m \tag{7}$$

Por reglas de construcción de engranajes el ancho del diente es la mitad del paso, por lo que se puede establecer que el punto final del diente (punto que marca el ancho del diente) está a una distancia w medido por la circunferencia de referencia en milímetros:

$$w = \left(\frac{\pi * m}{2}\right) \tag{8}$$

La siguiente figura representa un sector dentado de una rueda.

Figura 10: Paso.

Es necesario recorrer la circunferencia de referencia para llegar hasta el punto que marca el ancho del diente, por lo que se utilizan las ecuaciones paramétricas de una circunferencia:

$$x = \left(\frac{d}{2}\right) * \cos\theta \tag{9}$$

$$y = \left(\frac{d}{2}\right) * \sin\theta \tag{10}$$

Obteniendo los valores del punto $P(x;y)$, en cada caso se calcula la distancia desde el punto inicial P_0 el cual se halla en el intercepto de la curva Involuta original con la circunferencia de referencia, hasta el punto P con la ecuación de distancia Euclidiana:

$$D(P_1P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (11)$$

Cuando la distancia obtenida coincida con el valor de w, se toma el punto como final; entonces es hora de saber cuál es el punto medio del diente medido por la circunferencia de referencia, y se halla utilizando la fórmula del punto medio o punto equidistante:

$$X_{media} = \frac{x_1+x_2}{2} \quad (12)$$

$$Y_{media} = \frac{y_1+y_2}{2} \quad (13)$$

Una vez hallado el punto medio, se está en condiciones de construir el eje simétrico sobre el cual girará la curva Involuta; dicho eje simétrico no es más que un segmento con origen en el punto $(0;0;0)$ y que pase por el punto medio del diente.

Ya teniendo la simetría exacta solo falta unir la Involuta original y la Involuta imagen con la circunferencia de cresta que le pone tope al diente del engranaje. Hallando el intercepto de la circunferencia de cresta con la Involuta original y con la involuta imagen se obtienen dos puntos, P1 y P2 respectivamente, los cuales se utilizan para la construcción de un arco de circulo con radio r_a y con origen en el punto P1 y con final en el punto P2. Con esto logrado ya se tiene el diente del engranaje cilíndrico de dientes rectos.

$$r_a = \frac{d_a}{2} \quad (14)$$

- **Construyendo un engranaje cilíndrico de dientes rectos**

Con un diente concluido aún nos queda obtener los demás dientes, los cuales, mediante un pequeño procedimiento de cálculo y rotación quedaran idénticos al original.

Hallando un valor q, se obtendrá el ángulo en el cual se debe comenzar la construcción de un diente para que todos queden equidistantes entre sí:

$$q = \frac{360^o}{Z} \tag{15}$$

Se debe poner el primer diente a una distancia angular de q_1 contando a partir del eje x, el siguiente diente empezara a una distancia angular de:

$$q_1 + q_2 \tag{16}$$

Contando a partir del eje x, así sucesivamente con todos, siguiendo la siguiente ecuación:

$$q_c = q_{c-2} + q_{c-1} \tag{17}$$

Tomando c como el valor angular en cada caso; esto se repetirá z veces garantizando que se cumpla para todos los dientes que posea el engranaje. Luego se une cada diente con la circunferencia de fondo en caso que sea mayor su radio que el de la circunferencia básica, en otro caso, se unirán con dicha circunferencia básica.

Una vez terminada la creación de los dientes y la unión con su núcleo central[8], ya se tiene un engranaje cilíndrico de dientes rectos.

2.5 Conclusiones del capítulo

En este capítulo se presentó la propuesta de solución, la cual constituye un elemento fundamental para dar paso a la implementación del sistema. El modelo de dominio ofreció una visión de los principales conceptos asociado a los engranajes cilíndricos de dientes rectos y las relaciones que se establecen entre ellos y facilitó la identificación de los aspectos funcionales. Fue posible generar las historias de usuario y de esta manera comprender mejor los requerimientos del componente. Se identificaron los patrones de diseño presentes en el componente y se presentaron los prototipos de interfaz de usuario así como la explicación de la metodología de cálculo paso a paso.

[8] Llámese núcleo central a la circunferencia que sirve como base a los dientes del engranaje.

Capítulo III: Discusión de los resultados

En el presente capítulo se exponen cada uno los componentes que integran las etapas de implementación y prueba de la aplicación a desarrollar. Según las pautas que presentan las metodologías de desarrollo **XP** y **Scrum**, el proceso de implementación se caracteriza por ser iterativo, de modo que se describen cada una de las iteraciones comprobando que se cumplen cada uno de los objetivos planteados en el plan de iteraciones. Finalmente se procede a realizar conjuntamente con el cliente y el equipo de desarrollo, las pruebas correspondientes a cada historia de usuario, verificando que el sistema cumple con todas las funcionalidades definidas.

3.1 Implementación

Dentro del ciclo de vida de un *software*, se encuentra la fase de implementación; en esta fase se involucran personas, herramientas y recursos con el fin de completar todo el trabajo realizado previamente durante el ciclo de vida.

En esta fase se establece el estándar de codificación que se utilizara según las normas establecidas, se realiza la implementación de las historias de usuario planteadas anteriormente y se determinan las tareas de ingeniería que se realizan para su desarrollo. Estas tareas se describen, según la metodología "Scrum", en forma de ficha que contendrá la fecha de inicio, el nombre, el número identificador y la duración en días de la tarea, también contendrá el número de la historia de usuario a la que pertenece, el número del sprint, el programador responsable y el estado en que se encuentra.

3.1.1 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe quedar como si un único programador hubiera escrito todo el código de una sola vez. Al inicio de un desarrollo de *software*, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Para un programador comprender bien un sistema de *software*, influye directamente la legibilidad del código fuente. La mantenibilidad del código es la facilidad con que el sistema de *software* puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento (23).

- **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.

Ejemplo:

```
class MakeSpurGear
{
  ...
};
```

- ✓ Siempre se declara para todas las clases implementadas su respectivo destructor de clase.

Ejemplo:

```
~QoccViewerContext();
```

- ✓ La declaración de funciones o métodos siempre comenzara en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.

Ejemplo:

```
vector<TopoDS_Shape> makePinnon(...);
```

- ✓ Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.

Ejemplo:

```
double alpha = 0.0;
```

- ➢ ***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 CamelCase-lowerCamelCase.

Ejemplo:

```
vector<TopoDS_Shape> MakeSpurGear::makePinnon(double a, double m, double z1, ...)
{
  ...
}
```

- ✓ 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.

Ejemplo:

```
MakeSpurGear::MakeSpurGear(Handle_AIS_InteractiveContext theContext)
{
    context=theContext;
}
```

- **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.

Ejemplo:

```
da = d + (2.0 * m);
```

```
QString texto1 = QTime::currentTime().toString("h:mm:ss");
ui->textBrowser->setText(texto1);
```

- **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.

Ejemplo:

```
while(!found && i < puntosInvoluta.size())
{
  ...
}

 if(i == 1)
 {
   ...
 }
```

- **Comentarios en el código según el estándar de C++**
  - ✓ Comentarios pequeños.

Ejemplo:

```
/*fin de mirror*/
```

  - ✓ Otros comentarios.

Ejemplo:

```
/*
 *FIN DE PRUEBA
 */
```

  - ✓ Comentario de versión, descripción de clase y otras características de la clase o paquete.

Ejemplo:

```
/*
 ****************************************
 *Fin de Modelacion de dientes Gear1      *
 ****************************************
*/
```

3.1.2 Tareas de Ingeniería (Pila de sprint o "sprint backlog")

Es una lista que descompone las funcionalidades en tareas que debe realizar el equipo para construir un incremento: una parte completa y operativa del producto. Cada tarea de la pila de sprint tiene asignada una persona, y la indicación del tiempo que aún falta para terminarla. Durante el "*sprint*" el equipo actualiza a diario en la pila los tiempos pendientes de cada tarea. Es útil porque descompone el proyecto en unidades de tamaño adecuado para determinar el avance a diario e identificar riesgos y problemas sin necesidad de procesos complejos de gestión.

Tabla 2: Pila de Sprint. Sprint 3 "Visualizar entidades".

Sprint	Inicio	Duración (días)	Backlog
3	29-03-2015	6	3
ID	**Tarea**	**Delegado**	**Estado**
1	Visualizar el perfil del diente.	Sandy García Santos	Completado
2	Modelar el núcleo de la rueda secundaria.	Sandy García Santos	Completado
3	Modelar el núcleo del piñón.	Sandy García Santos	Completado
4	Visualizar el ensamble del piñón y la rueda secundaria.	Sandy García Santos	Completado

3.1.3 Descripción de la solución

Para la modelación geométrica de engranajes cilíndricos con dientes rectos se utilizaron una serie de bibliotecas las cuales permitieron la posterior visualización en la aplicación así como la manipulación de los datos, las cuales se documentaron con *Doxygen* para la persistencia de la información y se presentan a continuación:

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

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

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

BRepPrimAPI_MakeCylinder.hxx: Posee las características para la modelación de secciones cilíndricas.

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

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

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

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

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

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

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

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

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

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

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

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

3.2 Pruebas de Software

Las pruebas de *software* son un elemento crítico para la garantía de calidad del *software* y representa una revisión final de las especificaciones, del diseño y de la codificación. Estas constituyen un conjunto de herramientas, técnicas y métodos que evalúan el desempeño

de un programa. Estas involucran las operaciones del sistema, evaluando los resultados bajo condiciones controladas, lo que hace que la realización de pruebas al *software* sea un factor de vital importancia. Las pruebas de *software* son las investigaciones técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Son una actividad más en el proceso de control de calidad. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de *software*. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo (23).

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son las especificaciones para el comportamiento deseado y la funcionalidad de un sistema. Describen, por una HU dada, cómo el sistema se encarga de ciertas condiciones e insumos y con qué tipo de resultados (24). Los clientes junto a un miembro del equipo de desarrollo son los responsables de verificar que los resultados de estas pruebas sean los correctos para así tomar decisiones acerca de las mismas. Una HU no se puede considerar terminada hasta que no pase los test de aceptación. Es recomendable publicar los resultados de las pruebas de aceptación para que todo el equipo de desarrollo esté al tanto de esta información.

Tabla 3: Prueba de aceptación. Historia de usuario "Visualizar entidades".

Prueba de aceptación	
Nombre de HU: Visualizar entidades.	**Código:** CP_3
Nombre de la persona que realiza la prueba: Sandy García Santos	
Descripción del caso de prueba: Permite visualizar el engranaje cilíndrico de dientes rectos.	
Condiciones de ejecución: El usuario deberá haber realizado el cálculo del engranaje y dar click en el botón "*OK*".	

Entrada / Pasos de ejecución: ▪ Se debe visualizar el perfil del diente. ▪ Modelación del núcleo de la rueda. ▪ Modelación del núcleo del piñón. ▪ Visualizar el ensamble del piñón con la rueda.
Resultado esperado: Se visualizará el engranaje cilíndrico de dientes rectos en la pantalla.
Evaluación de la prueba: Satisfactoria.

3.2.2 Pruebas unitarias

Las pruebas unitarias clasificadas como pruebas de caja blanca dentro de la metodología de desarrollo **XP**, también llamadas pruebas modulares, permiten determinar si un módulo del sistema está listo y correctamente terminado. El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces o flujo de datos entre componentes, teniéndose bien claro los siguientes aspectos a la hora de su aplicación:

- Los datos de entrada son conocidos por el Encargado de Pruebas y estos deben ser preparados con minuciosidad, debido a que el resultado de las pruebas dependen de éstos.
- Conocer qué componentes interactúan en cada caso de prueba.
- Conocer de antemano qué resultados debe devolver el componente según los datos de entrada utilizados en la prueba.

Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos el módulo supera la prueba.

3.2.1.1 Biblioteca Qtest

Para la realización de las pruebas unitarias en el lenguaje C++ se utilizó la biblioteca Qtest, la cual es una práctica que consiste en tomar una pequeña porción de código y someterlo a unas pruebas de programación para probar su corrección. Por lo general se someten a pruebas, funciones o pequeñas funcionalidades específicas, con un conjunto de parámetros controlados que servirán para hacer las comprobaciones necesarias en

función de los valores de retorno de la funcionalidad testeada. Dentro de sus utilidades se pueden observar:

- Ayuda a producir un código de mayor calidad.
- Detección rápida de errores cuando se desarrollan nuevas funcionalidades o se realizan cambios en el código.
- Sirve como pequeña fuente de documentación sobre qué espera que haga el código.

Realizar pruebas unitarias permite al programador a escribir el código en pequeñas porciones con el fin de que puedan ser probadas independientemente.
En la siguiente figura se muestra el resultado de las pruebas unitarias aplicadas con **Qtest** a varias funcionalidades separadas en catorce porciones para lograr una mayor comprobación del código.

```
********* Start testing of TestUnitTest *********
Config: Using QTest library 4.8.6, Qt 4.8.6
PASS   : TestUnitTest::initTestCase()
PASS   : TestUnitTest::testPressureAngle()
PASS   : TestUnitTest::testModule()
PASS   : TestUnitTest::testNumberTeeth()
PASS   : TestUnitTest::testFacewidth()
PASS   : TestUnitTest::testCentralDiameter()
PASS   : TestUnitTest::testReduceDiameter()
PASS   : TestUnitTest::testReduceDepth()
PASS   : TestUnitTest::testStep()
PASS   : TestUnitTest::testCircContruction()
PASS   : TestUnitTest::testReduceDepthFacewidth()
PASS   : TestUnitTest::testReduceDpthBottomDiameter()
PASS   : TestUnitTest::testReduceDepthReduceDiameter()
PASS   : TestUnitTest::testMakePinnon()
PASS   : TestUnitTest::testSecondaryWheel()
PASS   : TestUnitTest::testLoadSpurGear()
PASS   : TestUnitTest::cleanupTestCase()
Totals: 17 passed, 0 failed, 0 skipped
********* Finished testing of TestUnitTest *********
```

Figura 11: Pruebas realizadas con Qtest.

3.2.3 Resultados de la pruebas

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

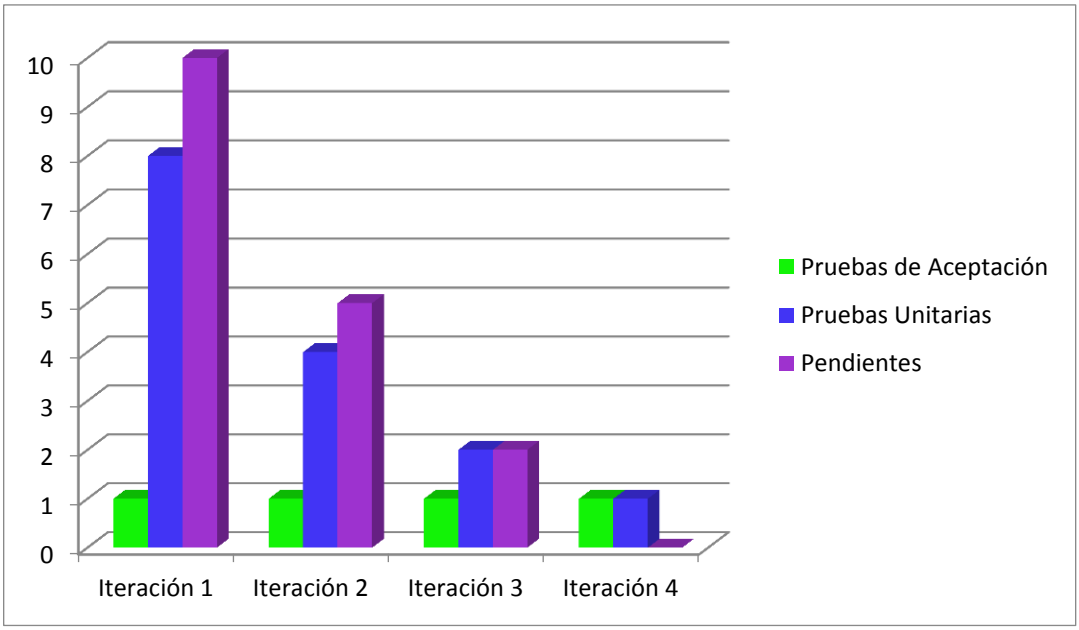


Figura 12: Resultados de las pruebas aplicadas al software en las distintas iteraciones.

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

3.3 Conclusiones del capítulo

En este capítulo se presentó la discusión de los resultados, la cual se divide en dos etapas, implementación y pruebas.

En la etapa de implementación fue posible presentar el estándar de codificación que se utilizó según las normas establecidas, así como las tareas de ingeniería o *Pila de Sprint*, las cuales se definieron según la metodología Scrum y se utilizaron para descomponer el

proyecto en unidades de tamaño adecuado para determinar el avance a diario e identificar riesgos y problemas sin necesidad de procesos complejos de gestión.

En la etapa de prueba del componente se realizaron pruebas unitarias y de aceptación, mediante las cuales se pudieron corregir ciertos errores en el componente.

Conclusiones generales

- Se desarrolló un componente que al ser introducido en la práctica, permitirá simplificar considerablemente el trabajo del diseñador mecánico.
- El componente se ha diseñado de manera que pueda ser integrado en cualquier aplicación de código abierto basada en la tecnología *OpenCascade***.**
- Es una solución única en el país, primera de este tipo que se conoce; tampoco existe, hasta el momento de concluir este trabajo, un desarrollo similar en las aplicaciones de código abierto disponibles (LibreCAD y FreeCAD).

Recomendaciones

- Concluir la implementación del cálculo a resistencia del engranaje.
- Incorporar al módulo las opciones para modelar engranajes internos (en este momento se modelan engranajes externos).
- Agregar al módulo de engranajes cilíndricos la variante de dientes helicoidales.

Glosario de términos

- **CAD**: Computer Aided Design
- **Catia**: *Computer-Aided Three Dimensional Interactive Applications*
- **OpenCascade**: Open Computer Aided Software for Computer Aided Design and Engineering
- **Framework**: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar
- **CAE**: Computer Aided Engineering
- **CAM**: Computer Aided Manufacturing
- **Diametral Pitch**: El diametral pitch o paso diametral es el cociente entre el número de dientes (Z) y el diámetro primitivo (d), expresado en pulgadas
- **Tab**: Pestaña que encapsula diferentes elementos y datos

Referencias bibliográficas

1. **Autodesk_AutoCAD.** AutoCAD. *AutoCAD.* [En línea] Autodesk, 1982. [Citado el: 02 de febrero de 2015.] http://www.autodesk.es/autocad.

2. Historia del diseño asistido por computadora. [En línea] 14 de 09 de 2014. [Citado el: 11 de 02 de 2015.]

3. Instituto superior politecnico Jose Antonio Hecheverria. [En línea] [Citado el: 11 de 02 de 2015.] www.cujae.edu.cu.

4. **Estrada, José Angel Lores.** *Módulo de transformación a entidades 2D.* La Habana : s.n., 2012.

5. **Inventor, Autodesk.** Autodesk Inventor. *Autodesk Inventor.* [En línea] Autodesk, 1999. [Citado el: 02 de febrero de 2015.] http://www.autodesk.com.

6. **TLAZALO, EMMANUEL JAÉN.** *CALCULO Y DISEÑO DE ENGRANES.* Veracruz : XALAPA, 2011. 1.

7. **Salazar, Miguel Arcia.** *Desarrollo de un componente visual para Qt utilizando el framework OpenCascade.* Ciudad de la Habana : s.n., 2011.

8. **Catia_Dassault_Systèmes.** Catia. *Catia.* [En línea] Dassault Systèmes. [Citado el: 02 de febrero de 2015.] http://www.3ds.com/products/catia.

9. **ArchiCAD_Graphisoft.** ArchiCAD. *ArchiCAD.* [En línea] Graphisoft, 1982. [Citado el: 02 de febrero de 2015.] http://www.graphisoft.com/products/archicad.

10. **SolidWorks_Corporation.** SolidWorks. *SolidWorks Corporation.* [En línea] Dassault Systèmes, 1993. [Citado el: 02 de febrero de 2015.] http://www.solidworks.com.mx.

11. **SolidEdge_Intergraph.** SolidEdge. [En línea] Siemens PLM Software, 1996. [Citado el: 02 de febrero de 2015.] www.siemens.com/solidedge.

12. **FreeCAD.** FreeCAD. *FreeCAD.* [En línea] [Citado el: 02 de febrero de 2015.] http://freecadweb.org.

13. **Comunidad_LibreCAD.** LibreCAD. *LibreCAD.* [En línea] [Citado el: 02 de febrero de 2015.] http://librecad.org/cms/home.html.

14. **Solis A, Camilo J y Figueroa D, Roberth G.** *Metodologias de Desarrollo de SW.* 2011.

15. **Pressman, Roger S.** *Software Engineering.* s.l. : Higher Education, 2010. 978-0-07-337597-7.

16. **Jacobson, Ivar, Rumbaugh, James y Booch, Grady.** *El lenguaje unificado de modelado. Manual de referencia.* 2000.

17. **Allen , Robert y Garlan, David .** *A formal approach to software architectures.* 1992.

18. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

19. **Stroustrup, Bjarne.** *The C++ Programming Language.* New Jersey : AddisonWesley, 1997. 0201889544.

20. **Heesch, Dimitri van.** Doxigen. [En línea] [Citado el: 09 de febrero de 2015.] http://www.doxygen.org.

21. CCCC. [En línea] [Citado el: 09 de febrero de 2015.] http://cccc.sourceforge.net.

22. **Pérez, Yarisleydis Barzaga.** *Componente de Interpretación de registros.* La Habana : s.n., 2011.

23. Parasolid. [En línea] ShapeData. [Citado el: 09 de febrero de 2015.] http://www.plm.automation.siemens.com/en_us/products/open/parasolid.

24. ACIS. [En línea] Spatial Corporation. [Citado el: 09 de febrero de 2015.] www.spatial.com.

25. *Charla 1: Introduccion a los patrones de diseño. Algunos patrones básicos.* 18.

26. **HENDRICKSON, E.** *Agility for Testers.* Pacific Northwest Software Quality Conference : s.n., 2011.

27. **KOSKELA, LASSE.** *TEST DRIVEN, Practical TDD and Acceptance TDD for Java Developers.* s.l. : Manning Publication Co., 2008. 1-932394-85-0.

28. **Ivar Jacobson, Grady Booch,James Rumbaygh.** *El proceso unificado de desarrollo de software.* Madrid : s.n., 2000.

29. **Pressman, Roger.** *Ingeniería de software. Un enfoque práctico.* 2002.

30. *Tecnologias Qt, OpenCascade y Ocaf.* **Meyer, Lisandro Damián Nicanor Pérez.** Buenos Aires : s.n., 2007. 1.

31. **MacKay, David, Cipolla, Roberto y Love, Tim.** *Tutorial Guide to C++ Programming.* 2008. 2008.2.

32. **García Hernandez, Alex y Lamas López, Roanny.** *Sistema para la planificación y control del entrenamiento en deportes de combate.* La Habana : s.n., 2014.

33. OpenCascade Tecnology. *3D modeling & numerical simulation.* [En línea] 2000-2015. http://www.opencascade.org/.

Anexos

Tabla 4: Historia de usuario 1 "Especificar parámetros de entrada".

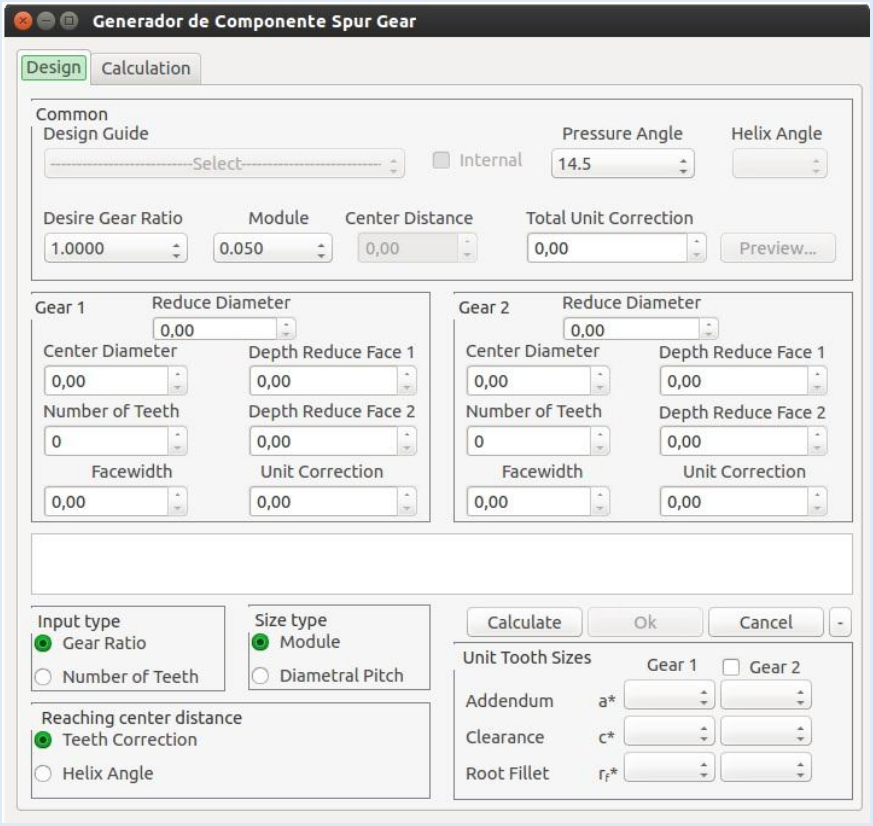
<table>
<tr><th colspan="3">Historia de usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre del aspecto: Especificar parámetros de entrada.</td></tr>
<tr><td colspan="2">Programador: Sandy García Santos</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo Estimado: 9 días</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Alto</td><td>Tiempo Real: 1,8 semanas</td></tr>
<tr><td colspan="3">Descripción: El sistema permite especificar parámetros de entrada para poder realizar el cálculo del engranaje.</td></tr>
<tr><td colspan="3">Observaciones: El usuario deberá introducir los datos correspondientes al número de dientes del piñón, el número de dientes de la rueda, el valor del módulo para el sistema de engranajes, valor de la relación de transmisión, el valor del ángulo de presión, el valor del ancho de la cara del piñón, el valor del ancho de la cara de la rueda, el valor de las correcciones de unidades del piñón y el valor de las correcciones de unidades de la rueda.</td></tr>
<tr><td colspan="3">Prototipo de interfaz:
 
</td></tr>
</table>

Tabla 5: Historia de usuario 3 "Visualizar entidades".

Historia de usuario	
Número: 3	**Nombre del aspecto:** Visualizar entidades.
Programador: Sandy García Santos	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 6 días
Riesgo en Desarrollo: Medio	**Tiempo Real:** 1,2 semanas
Descripción: El sistema permitirá visualizar entidades mecánicas tales como engranajes cilíndricos de dientes rectos.	
Observaciones: Se debe modelar el perfil del diente del engranaje, el núcleo del piñón, el núcleo de la rueda secundaria y el ensamble del piñón con la rueda secundaria para ser visualizados.	
Prototipo de interfaz:	

Tabla 6: Historia de usuario 4 "Comprobar resultados del cálculo del engranaje".

Historia de usuario	
Número: 4	**Nombre del aspecto:** Comprobar resultados del cálculo del engranaje.
Programador: Sandy García Santos	**Iteración Asignada:** 3
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Medio	**Tiempo Real:** 1 semanas
Descripción: El sistema mostrara los valores de los diferentes parámetros del engranaje separados por rueda.	
Observaciones: El piñón y la rueda secundaria deben quedar correctamente ensamblados, los datos del engranaje deben coincidir con los cálculos realizados, la corrección de unidades del piñón y de la rueda secundaria deben estar en relación con la corrección de unidades totales y el componente debe poder cargarse en Salome y en FreeCAD.	
Prototipo de interfaz:	

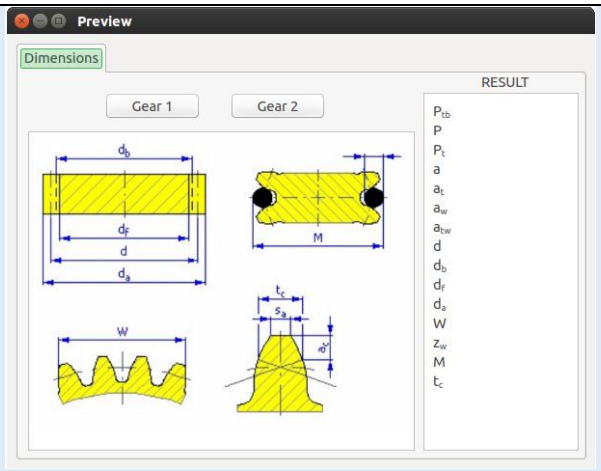


Tabla 7: Pila de Sprint. Sprint 1, 2 y 4.

Sprint	Inicio	Duración (días)	Backlog
1	21-02-2015	9	1
ID	**Tarea**	**Delegado**	**Estado**
1	Crear campo para introducir el número de dientes del piñón.	Sandy García Santos	Completado
2	Crear campo para introducir el número de dientes de la rueda secundaria.	Sandy García Santos	Completado
3	Crear lista desplegable con los valores del módulo para que el usuario seleccione el que desee para el sistema de engranaje.	Sandy García Santos	Completado
4	Crear lista desplegable con los	Sandy García Santos	Completado

	valores de la relación de transmisión para que el usuario seleccione la que desee para el sistema de engranaje.		
5	Crear lista desplegable con los valores del ángulo de presión para que el usuario seleccione el que desee para el sistema de engranaje.	Sandy García Santos	Completado
6	Crear campo para introducir el valor del ancho de la cara del piñón.	Sandy García Santos	Completado
7	Crear campo para introducir el valor del ancho de la cara de la rueda secundaria.	Sandy García Santos	Completado
8	Crear campo para introducir el valor de la corrección de unidades del piñón.	Sandy García Santos	Completado
9	Crear campo para introducir el valor de la corrección de unidades de la rueda secundaria.	Sandy García Santos	Completado

Sprint	Inicio	Duración (días)	Backlog
2	03-03-2015	25	2

ID	Tarea	Delegado	Estado
1	Calcular las circunferencias de construcción del engranaje.	Sandy García Santos	Completado
2	Calcular los puntos de la curva de involuta como perfil del diente.	Sandy García Santos	Completado
3	Calcular la distancia entre centros.	Sandy García Santos	Completado
4	Calcular el espacio entre dientes.	Sandy García Santos	Completado
5	Calcular el ancho de los dientes.	Sandy García Santos	Completado

Sprint	Inicio	Duración (días)	Backlog
4	05-04-2015	5	4

ID	Tarea	Delegado	Estado
1	Ensamblar el piñón y la rueda secundaria.	Sandy García Santos	Completado
2	Establecer la corrección de unidades totales.	Sandy García Santos	Completado
3	Montar el componente en el	Sandy García Santos	Completado

	FreeCAD y Salome.		

Tabla 8: Prueba de aceptación 1. Historia de usuario "Especificar parámetros de entrada".

Prueba de aceptación	
Nombre de HU: Especificar parámetros de entrada.	**Código:** CP_1
Nombre de la persona que realiza la prueba: Sandy García Santos	
Descripción del caso de prueba: Permite especificar parámetros de entrada.	
Condiciones de ejecución: El usuario deberá abrir la ventana "*Spur Gears*".	
Entrada / Pasos de ejecución: ▪ Introducir el valor del número de dientes del piñón. ▪ Introducir el valor del número de dientes de la rueda. ▪ Introducir el valor del módulo para el sistema de engranajes. ▪ Introducir el valor de la relación de transmisión para el sistema de engranajes. ▪ Introducir el valor del ángulo de presión. ▪ Introducir el valor del ancho de la cara del piñón. ▪ Introducir el valor del ancho de la cara de la rueda. ▪ Introducir el valor de las correcciones de unidades del piñón. ▪ Introducir el valor de las correcciones de unidades de la rueda.	
Resultado esperado: En cada campo deben estar los valores adecuados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 9: Prueba de aceptación 2. Historia de usuario "Realizar cálculo del engranaje".

Prueba de aceptación	
Nombre de HU: Realizar cálculo del engranaje.	**Código:** CP_2
Nombre de la persona que realiza la prueba: Sandy García Santos	
Descripción del caso de prueba: Permite calcular todos los parámetros necesarios para la construcción del engranaje.	

Condiciones de ejecución: El usuario deberá de haber introducido todos los parámetros iniciales correctamente.
Entrada / Pasos de ejecución: ▪ Realizar el cálculo de las circunsferencias necesarias para la construcción del engranaje. ▪ Realizar el cálculo de la curva de involuta para el perfil del diente. ▪ Realizar el cálculo de la distancia entre centros. ▪ Realizar el cálculo de espaciado entre dientes. ▪ Realizar el cálculo del ancho de los dientes.
Resultado esperado: Cuando se pulsa el botón "*Calcular*", el sistema muestra un mensaje en dependencia de si se introdujeron bien los resultados o no.
Evaluación de la prueba: Satisfactoria.

Tabla 10: Prueba de aceptación 4. Historia de usuario "Comprobar resultados del cálculo del engranaje".

Prueba de aceptación	
Nombre de HU: Comprobar resultados del cálculo del engranaje.	**Código:** CP_4
Nombre de la persona que realiza la prueba: Sandy García Santos	
Descripción del caso de prueba: Permite visualizar los valores de los diferentes parámetros del engranaje y comprobar los resultados de los cálculos.	
Condiciones de ejecución: El usuario deberá haber realizado el cálculo del engranaje y dar click en el botón "*Preview*".	
Entrada / Pasos de ejecución: ▪ El piñón y la rueda deben quedar ensamblados. ▪ Las correcciones de unidades de la rueda y el piñón deben coincidir con la corrección de unidades total una vez que se sumen. ▪ Se mostraran todos los valores necesarios para el cálculo del engranaje.	
Resultado esperado: Al cargar en "Salome" el modelo exportado, debe poder mallarse completamente.	
Evaluación de la prueba: Satisfactoria.	

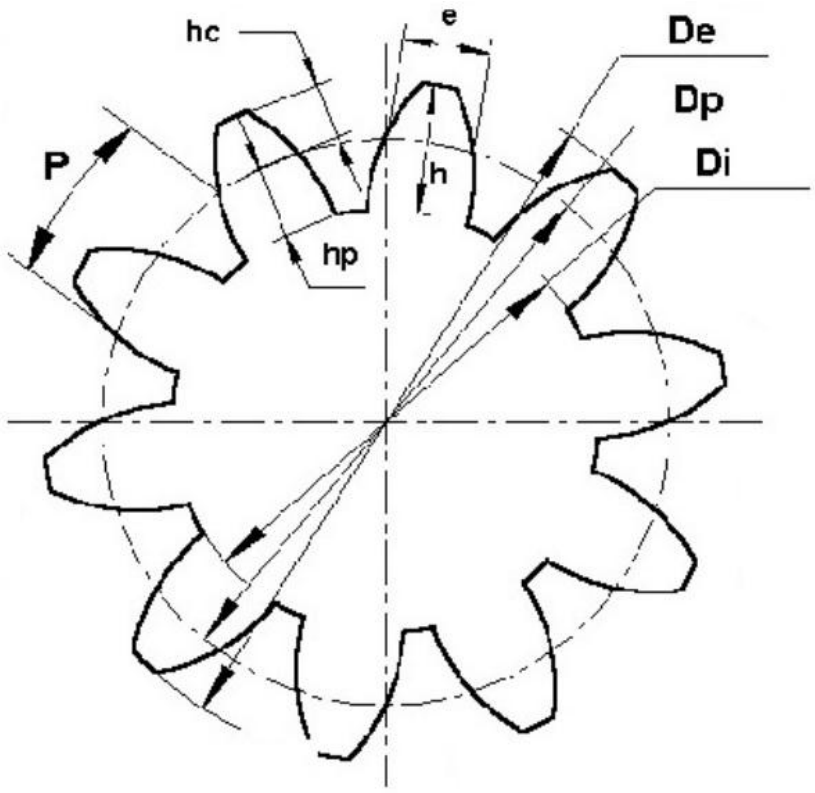


Figura 13: Representación de la metodología de cálculo de engranajes cilíndricos de dientes rectos.

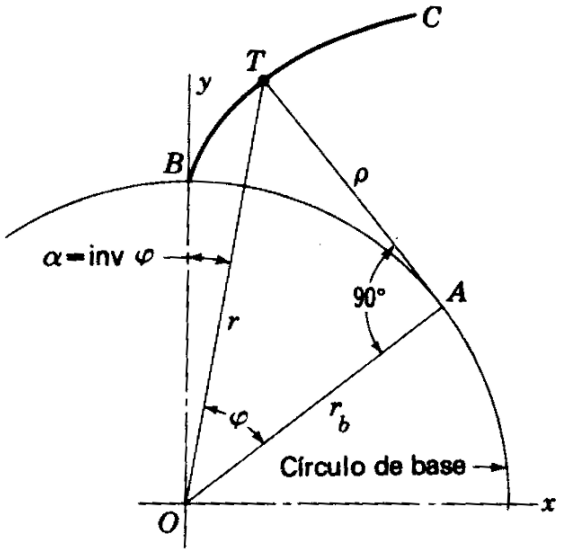


Figura 14: Creación del perfil de envolvente de círculo utilizando sus ecuaciones paramétricas.

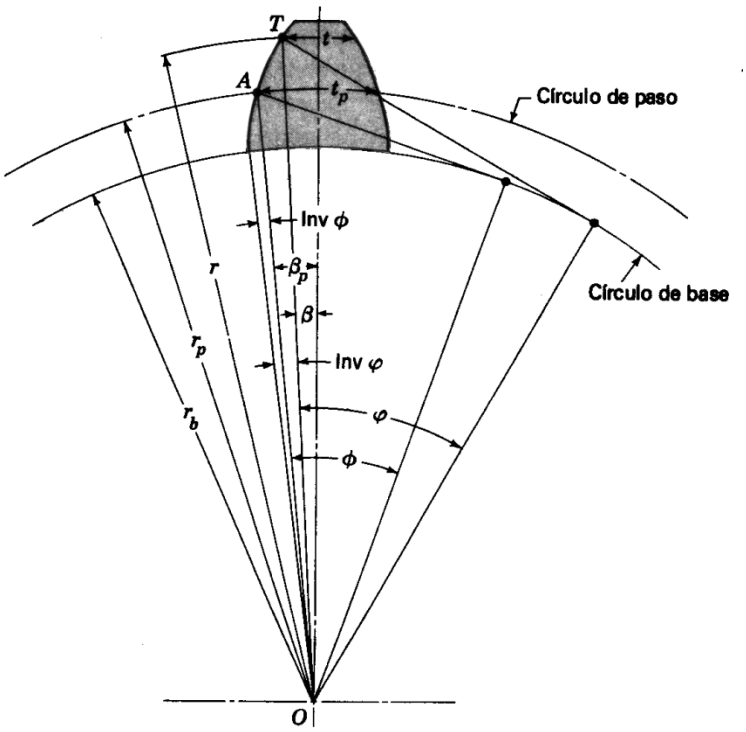


Figura 15: Creando un diente del engranaje cilíndrico con perfil de envolvente de circulo.

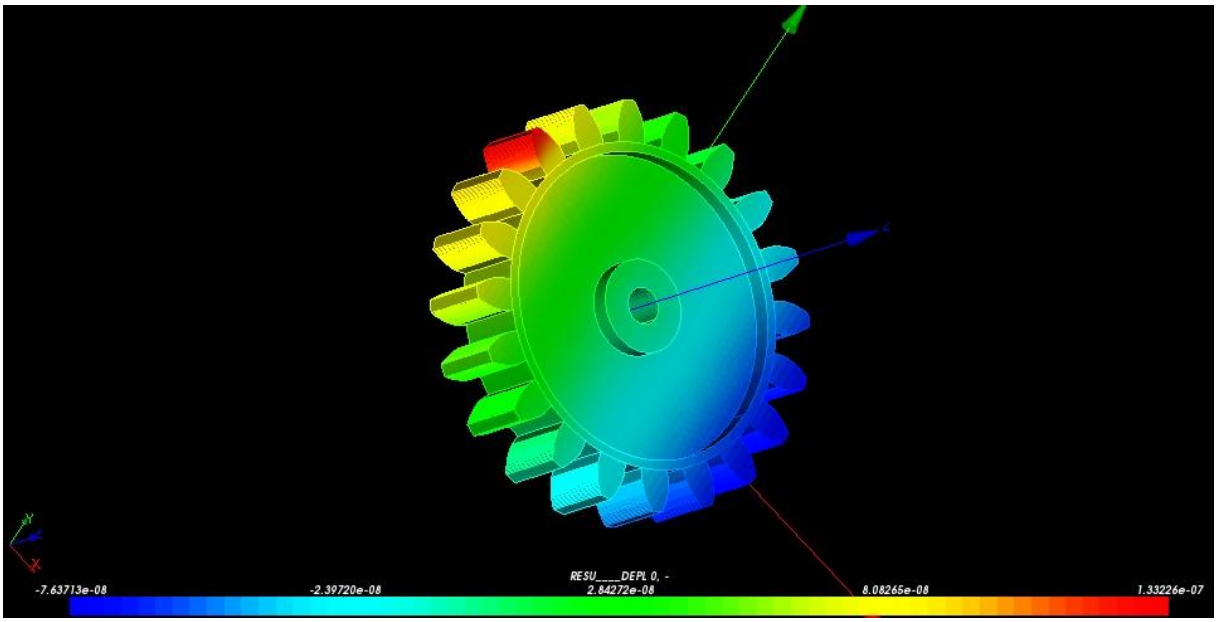


Figura 16: Simulación de fuerza del engranaje generado con el componente realizada con el software Salome-Meca.

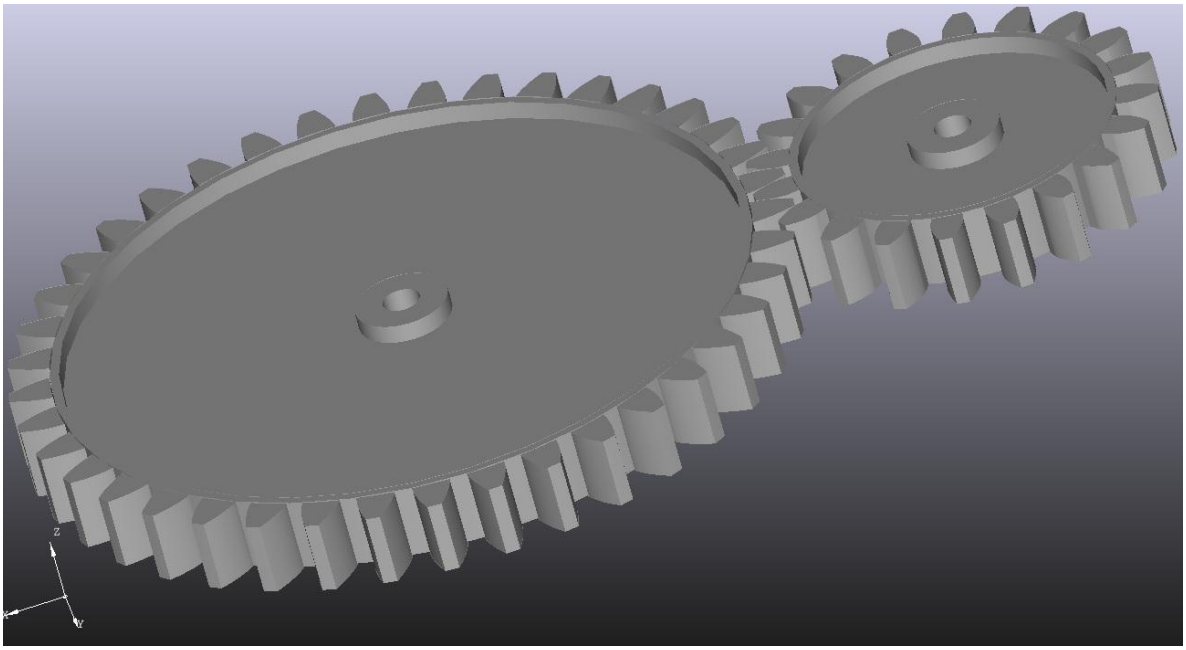


Figura 17: Engranaje cilíndrico de dientes rectos generado por el componente.