Universidad de las Ciencias Informáticas

Facultad 4

Componente para la modelación de árboles escalonados en el Sistema CAD 2D

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

Autor: Abel Fernández Ferrer.

Tutores: Dr. Augusto Cesar Rodríguez Medina.

Ing. Lianet Cylwik López.

La Habana, junio del 2015.

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Abel Fernández Ferrer

Firma del Autor

_____ _____

Dr.C Augusto Cesar Rodríguez Medina

Firma del tutor

Ing. Lianet Cylwik López

Firma de la cotutora

“Los grandes espíritus siempre han tenido que luchar contra la oposición feroz de mentes mediocres”

Albert Einstein

AGRADECIMIENTOS

Primero que todo, agradecer a mis padres y a mi familia en general por el apoyo brindado durante estos cinco años de carrera, por los cientos de consejos que me dieron y los miles que les faltan por darme. A mi tutor que como ningún otro que haya visto y conocido, ha sido la guía para la correcta elaboración de esta investigación, siempre dando el apoyo necesario, tomando su tiempo libre fuera de su horario laboral para revisar el trabajo, por demostrarnos que no hay problemas sin solución sino personas sin actitud, por todos sus consejos y por su gran paciencia, muchas gracias. A mi tutora por aconsejarme tantas cosas para un correcto formato y de vez en vez organizar mis ideas, por apoyarme, y sobre todo ser muy paciente. Agradecer a mis amigos, a todos los profesores que me han ayudado mucho durante este trabajo de tesis, dando sus minutos y compartiendo su sabiduría con este joven aprendiz, a todos, muchas gracias.

DEDICATORIA

RESUMEN

El presente trabajo consiste en el desarrollo de un componente para la modelación geométrica de árboles escalonados utilizando la tecnología **OpenCascade** y el *framework* de **Qt**, para su posterior integración en la aplicación “Sistema para el dibujo técnico mecánico en dos dimensiones asistido por computadora”, la cual posee como premisa brindar una herramienta de código abierto que asista los procesos de diseño en la industria cubana. El desarrollo de la propuesta fue guiado por la metodología Scrum con XP, UML como lenguaje de modelado y C++ para la implementación, utilizándose además QtCreator como Entorno de Desarrollo Integrado y *Visual Paradigm for UML* como herramienta CASE. La investigación finaliza con las pruebas unitarias y de aceptación del software, las cuales demostraron la correcta implementación de todas sus funcionalidades.

Palabras clave: árboles escalonados, diseño, interfaz, plataforma, herramienta.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Este trabajo estuvo condicionado por las necesidades de un proyecto de investigación recientemente creado en la facultad 4, destinado a resolver carencias de productos informáticos en la industria nacional; la situación consiste en que los trabajos de ingeniería y diseño se realizan en las empresas cubanas con *software* propietarios, pero sin licencias de uso, lo que implica un problema en el tema de la soberanía tecnológica.

Aunque el proyecto mencionado tiene como objetivo desarrollar una alternativa de AutoCAD con funcionalidades para el diseño paramétrico, se ha evaluado la posibilidad y factibilidad de implementar componentes que permitan modelar piezas de elevado nivel de complejidad morfológica de forma automatizada, lo que sería un paso de avance en el incremento de la productividad de los ingenieros. Para el cumplimiento de este propósito se indicó implementar un componente para la modelación de árboles escalonados, utilizando la tecnología **OpenCascade** y el *framework* **Qt**, lo que constituyó el eje central de esta tesis.

El cuerpo del documento se estructuró en tres capítulos: el primero contiene la fundamentación teórica del proceso de investigación y desarrollo, en el que se expone el resultado del análisis realizado a los sistemas existentes, identificándose las tecnologías necesarias para obtener la solución; en el segundo se describen los principales conceptos asociados al uso de la metodología SCRUM con XP, así como la planificación del proceso y los artefactos generados; en el capítulo 3 se relaciona el proceso de ejecución de las iteraciones efectuadas para la implementación además del análisis de los resultados mediante pruebas de aceptación y unitarias guiadas por el principio de desarrollo establecido en la metodología XP.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se exponen los aspectos principales que fundamentan, desde el punto de vista teórico, la investigación realizada; consta de los resultados del estudio preliminar realizado sobre los sistemas para el diseño asistido por computadora (*Computer Aided Design*), tanto propietarios como de código abierto, además de exponer los resultados del estudio referente a las metodologías ágiles así como los *frameworks* de modelado existentes que soportan el desarrollo de aplicaciones 3D.

1.1 Diseño de la investigación

El primer paso para cumplir con las exigencias del proyecto en que está insertado este trabajo fue obtener información sobre la situación existente, para poder identificar las necesidades reales de la industria nacional en la rama del diseño asistido por computadoras, cuyo término en inglés es "*Computer Aided Design*" (referenciado con frecuencia por su acrónimo CAD).

El uso del diseño asistido por ordenador fue algo que encontró su desarrollo hace tan sólo medio siglo atrás, pues antes de esto, casi todos los dibujos se realizaban utilizando papel, lápiz y mesas de dibujo. Cuando se precisaba realizar cambios, era necesario borrar y volver a dibujar y si el cambio era importante, se repetía el dibujo por completo; si un cambio afectaba a otros documentos (planos de montaje, planos de conjunto, entre otros) se tenía que buscar a mano en cada uno de ellos y modificarlos, constituyendo esto una situación la cual, ralentizaba a veces los procesos productivos. No fue hasta principios de los años 60 cuando surge en los laboratorios Lincoln del *Massachusetts Institute of Technology* (MIT) el primer sistema gráfico para el diseño llamado "Sketchpad", aunque por el alto precio de los ordenadores destinados a esta tecnología, solo algunas compañías de aviación o automóviles desarrollaron esta herramienta. Durante los años 70 se inició la migración de la pura investigación hacia su uso comercial, aunque todavía el software de este tipo se mantenía siendo desarrollado por grupos internos de grandes fabricantes de automoción y aeroespaciales como General Motors, Mercedes-Benz, Renault, Nissan, Toyota, Lockheed, McDonnell-Douglas, Dassault, por citar algunos. Para los años 80 lo que había comenzado como un tema de investigación fue floreciendo comercialmente con el avance de las computadoras personales, convirtiéndose así en una dura competencia entre diferentes firmas comerciales. En la actualidad se evidencian grandes brechas entre las soluciones existentes para del dibujo asistido por ordenador, en las cuales las privativas poseen características novedosas

y revolucionarias, mientras que las libres y de código abierto presentan poco desarrollo y funciones bastante restringidas como para situarse en las exigencias actuales del mercado.

Actualmente la industria nacional no cuenta con un sistema que sustente los procesos de diseño en las confecciones de la industria, por lo que el mismo se ve obligado a funcionar sobre plataformas sin licencia de uso, incurriendo así en leyes de derecho internacional y restringiendo la comercialización de los productos resultantes. Por esta razón en la Universidad de las Ciencias Informáticas después de varios intentos por parte de otras áreas, surge el proyecto experimental "Sistema CAD2D", con el objetivo de crear un producto para la modelación en tres dimensiones que tribute al desarrollo del diseño enriqueciéndose de las mejores prácticas ya existentes. En el actual proceso de desarrollo se desean generar distintos componentes llamados aceleradores de diseño, los cuales garantizan un mejor desempeño en la confección de estructuras de alta complejidad morfológica, lo cual tributaría sustancialmente a un aumento de la productividad y enriquecería notablemente los trabajos. Dadas estas condiciones y en el contexto actual del proyecto se ha asignado el siguiente **problema a resolver**:

Inexistencia de un componente para la modelación de árboles escalonados en el sistema para el dibujo técnico mecánico en dos dimensiones asistido por computadora (SistemaCAD2D).

El **objeto de estudio** de la investigación se centra en la modelación de entidades geométricas, teniendo como **campo de acción**, la computación gráfica.

Dado el problema expresado se define el siguiente **objetivo general:**

Desarrollar un componente para la modelación de árboles escalonados en el Sistema CAD 2D utilizando tecnologías de código abierto.

Partiendo del objetivo planteado anteriormente se formulan los siguientes **Objetivos específicos:**

- Elaborar el marco teórico conceptual que sustente la investigación.
- Desarrollar una propuesta de solución.
- Implementar y probar los resultados de la solución propuesta.

Para dar cumplimiento a los objetivos específicos planteados se proponen las siguientes **tareas de investigación:**

- Asimilación de los principales conceptos y tecnologías que se requieren para el desarrollo del componente.
- Elaboración del fundamento teórico de la investigación.
- Descripción detallada de los diferentes aspectos que conforman el componente.
- Obtención de requisitos mediante el acelerador de diseño para la modelación de árboles escalonados del sistema Autodesk Inventor.

- Definir y modelar el 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 para la modelación de árboles escalonados.
- Realización de pruebas al componente implementado.

Para una mejor organización del proceso y el trabajo de desarrollo se formularon las siguientes **preguntas de investigación:**

- ¿Cuáles serán las tecnologías necesarias para modelar entidades geométricas en 3 dimensiones?
- ¿Cómo visualizar secciones cónicas y cilíndricas haciendo uso de tecnologías para el modelado en 3 dimensiones?
- ¿Cómo se podría modelar troncos de cono utilizando herramientas de modelación 3D?
- ¿Cuáles serían las funciones necesarias para la unión de secciones continuas de tipos cilíndricas y cónicas?
- ¿Cómo realizar modificaciones a las secciones que conforman el árbol escalonado?

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

Métodos teóricos:

El método **Analítico-Sintético** se utilizó para estudiar y revisar la documentación sobre los sistemas de modelación de árboles escalonados ya existentes, así como para analizar lo referente a las tecnologías de modelado 3D; obteniendo los elementos necesarios que permitan la elaboración de una solución viable para el problema de investigación.

Métodos empíricos:

El método de **Observación** permite adquirir información necesaria y puede utilizarse en cualquier fase de la investigación, además que posibilita obtener la solución del problema desde diferentes puntos de vista.

1.2 Los sistemas CAD

Los sistemas CAD son programas computacionales para crear representaciones gráficas de objetos físicos ya sea en segunda o tercera dimensión (2D o 3D). El software CAD puede ser especializado para usos y aplicaciones específicas. Es ampliamente utilizado para la animación computacional y efectos especiales en películas, publicidad y productos de diferentes industrias, donde el software realiza cálculos para determinar una forma y tamaño óptimo para una variedad de productos y aplicaciones de diseño industrial (1).

En el diseño industrial y de productos, el CAD es usado principalmente para la creación de modelos de superficie o sólidos en 3D, o bien, dibujos de componentes físicos basados en vectores en 2D. Sin embargo, también se utiliza en los procesos de ingeniería desde el diseño conceptual, a través de fuerza y análisis dinámico de ensambles hasta la definición de métodos de manufactura. Estas herramientas le permiten al ingeniero analizar interactiva y automáticamente las variantes de diseño, para encontrar el diseño óptimo en la manufactura mientras se minimiza el uso de prototipos físicos. Dentro de sus grandes beneficios se encuentran los siguientes (1):

1. Menores costos de desarrollo de productos, aumento de la productividad, mejora en la calidad del producto y un menor tiempo de lanzamiento al Mercado.
2. Mejor visualización del producto final, los sub-ensambles parciales y los componentes en un sistema CAD agilizando el proceso de diseño.
3. Ofrece gran exactitud de forma que se reducen los errores.
4. Brinda una documentación más sencilla y robusta del diseño, incluyendo geometría, dimensiones, lista de materiales y otros.
5. Permite una reutilización sencilla de diseños de datos y mejores prácticas.

1.3 Árboles escalonados

Los árboles escalonados son elementos que se utilizan en la transmisión del movimiento circular en las maquinarias y equipos de muy diversas industrias. Se denomina árbol al órgano giratorio de la máquina, que cumple con las funciones de sostener a los elementos acoplados a él y de transmitir un par de torsión simultáneamente (2).

1.4 Soluciones CAD privativas y de software libre

En el mundo existen diversas soluciones CAD para el modelado en 3D que cuentan con disímiles herramientas para la modelación. Dentro de estas se destacan las soluciones privativas y libres con grandes brechas entre ambas, siendo las privativas acreedoras de buenos mecanismos para el diseño y las soluciones libres aunque se encuentran al alcance de la industria nacional poseen desventajas sustanciales como el poco dinamismo y las inconsistencias en sus herramientas al ser productos poco desarrollados. A continuación se expondrá un análisis de cada una de estas soluciones y se hará referencia a sus características fundamentales así como a sus desventajas.

1.5 Productos de Dassault Systèmes

Dassault Systemes es una de las empresas más prestigiosas en la actualidad que cuenta con disímiles sistemas informáticos sobre todo para la modelación, simulación e interactividad 3D.

Dentro de sus productos para la modelación mecánica tienen CATIA[1] y Solid Works, de los cuales se hablará a continuación, exponiéndose en cada caso sus características principales.

1.5.1 CATIA

CATIA es una aplicación para el modelado gráfico que tuvo su primera aparición en el año 1967. Desde sus primeros momentos este producto se reconoció por poseer una notable calidad y exigencia con sus desarrolladores debido a la seriedad de su objetivo inicial, pero no fue hasta el año 1981 en que decidieron fundar un grupo integrado de desarrollo que sería el llamado Dassault Systemes. Dicho producto inició su labor con solo 15 miembros y tecnología CAD/CAM adquirida por IBM para el dibujo 2D y 3D, para dicha fecha también serían firmados contratos con la misma IBM para marketing, ventas y soporte solidificando su afiliación con la misma. También fueron firmados acuerdos de colaboración con las empresas BMW, Honda y Mercedes Benz, líderes en la industria de automoción (3). Durante el año 82 se realizaría el primer lanzamiento de CATIA Versión 1 como módulo complementario para el diseño 3D, la modelización de superficies y la programación de NC (Numerical Control).

Durante el transcurso de su historia CATIA ha sido declarado internacionalmente como herramienta de vanguardia en el área de la automoción, aviación, entre disímiles productos para la industria lo que le ha concedido la adquisición de recursos tecnológicos actuales como Enginuity, Intercim, *International Business Machines* **(**IBM**)** PLM, Geemsoft, Enginius Software, ICEM, Apriso, ACIS, entre otras (4). Técnicamente CATIA es un sistema que provee de soporte para la creación de productos desde la etapa del diseño hasta la producción y el análisis de los mismos, además de presentar dentro de sus disímiles características componentes de simulación que permiten la prueba de los diseños, la corrección de errores y la innovación tecnológica. 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++. Proporciona las herramientas necesarias para la modelación en 3D de árboles escalonados por lo que se podría considerar como una solución pero no resulta viable por las siguientes razones:

- Su licencia es privativa y el costo de adquisición es alto.
- El bloqueo económico prohíbe su adquisición y distribución.
- Código cerrado, por lo que imposibilita la adquisición y reutilización de código.
- Es un sistema mono plataforma compatible solo con versiones de Microsoft Windows.

[1] Computer-Aided Three Dimensional Interactive Applications.

1.5.2 SolidWorks

SolidWorks es un software para modelado mecánico en 3D, desarrollado en la actualidad por SolidWorks Corp., una filial de *Dassault Systèmes* (5), 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. 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. Este producto cuenta en la actualidad con un sistema de certificación el cual avala a los diseñadores con categorías que acreditan internacionalmente a quien las posee, ejemplo de estas certificaciones son las siguientes:

- CSWA (*Certified SolidWorks Associate*): se enfoca a temas sobre el manejo básico de las herramientas del software.
- CSWP (*Certified SolidWorks Professional*): evalúa habilidades avanzadas de diseño de partes y ensambles, al haber obtenido más conocimiento y la destreza en el manejo de las herramientas, los ingenieros pueden especializarse en diferentes módulos, como el dibujo, chapa metálica, superficies, piezas soldadas y herramientas de moldes.
- CSWE (*Certified SolidWorks Expert*) con esta se demuestra un conocimiento completo de todas las áreas del software.

Desventajas y restricciones de uso:

- Su licencia es privativa y el costo de adquisición es alto.
- Código cerrado, por lo que imposibilita la adquisición y reutilización de código.
- Es un sistema mono plataforma siendo solamente compatible con versiones de Microsoft Windows.

1.6 Soluciones de AutoDesk Incorporated

Esta compañía está dedicada al desarrollo y soporte de software para el diseño en 2D y 3D en industrias de manufacturas, infraestructuras, construcción, medios y entretenimiento y datos transmitidos vía inalámbrica.

Dicha empresa actualmente cuenta con una infraestructura poderosa que está conformada por divisiones de desarrollo para casi todos los ámbitos de la vida cotidiana presentando Soluciones de la manufactura (MSD), Soluciones Arquitectura, Ingeniería y Construcción Civil (AEC), División de Medios y Entretenimiento (M&E), Soluciones de plataforma y negocios emergentes

(PSEB) incluyendo soluciones geoespaciales y de plantas, contenido y búsqueda así como los Laboratorios Autodesk, Autodesk Consulting y la División de Servicios basados en locación (LBS) (6).

1.6.1 AutoCAD

Este sistema fue uno de los primeros sistemas para el modelado y dibujo técnico que dio a luz en el año 1982, año en el cual se efectuó su primer reléase COMDEX *Trade Show* de Las Vegas en noviembre pero los primeros en adquirir AutoCAD debieron esperar al siguiente mes para instalar el nuevo programa. Si bien las utilidades de AutoCAD 1.0 eran muy elementales, permitían mucho más que representar gráficamente coordenadas de puntos. Por ejemplo, ya ofrecía *layers*, texto y hasta un menú de comandos, todo ello con muchas limitaciones. Los *layers* no eran nombrados por el usuario y la cantidad posible era ilimitada. El menú lateral, único hasta 5 años después, sólo permitía acceder a 40 comandos.

Dentro de los avances de este sistema se pueden tener una larga lista de liberaciones de producto, marcada por ser una herramienta que desde un principio tuvo incluso a los más asiduos a este producto en shock con la salida constante de nuevas características que lo han convertido un sistema potente y útil para la mayoría de las necesidades de esos momentos. Permite el trabajo en equipo y minimiza cualquier posibilidad de errores. Se destacan opciones para la edición de dibujos en 3D, modelación de estructuras tanto simples como complejas, apoyadas en el dibujo paramétrico que viene siendo como una extensión misma de la mesa de dibujo tan utilizada por los diseñadores permitiendo dibujar de manera rápida, ágil y sencilla, sin las desventajas que tiene el dibujo cuando se hace a mano. Permite intercambiar información no solo por papel, sino mediante archivos, y esto representa una mejora en rapidez y efectividad a la hora de interpretar diseños, sobretodo en el campo de las tres dimensiones. Cuenta con herramientas para gestión de proyectos donde se puede compartir información de manera eficaz e inmediata. Esto es muy útil sobretodo en ensamblajes, contrastes de medidas y otras situaciones. Muy útil para el acabado y la presentación de un proyecto o plano, ya que tiene herramientas para que el documento en papel sea perfecto, tanto en estética, como, lo más importante, en información, que ha de ser muy clara. Cuenta con una avanzada gamma de funcionalidades para el dibujo en 2D, se podrían citar los dimensionados que están divididos por lineales, angulares, diametrales, radiales, ordinales, entre otros, además de estilos de acotación, manipulación de variables, estilos de texto, alineamiento, caracteres especiales, funciones de importación y exportación de archivos según su utilidad y composición y ploteo de planos. Mientras que para el modelado en 3D cuenta con funciones para descripción analítica de la volumetría, contornos y dimensiones de un

objeto. Posee herramientas para la obtención de vistas, secciones, perspectivas, detalles, modelados de superficies, creación de modelos tridimensionales mediante el uso de la extrusión, primitivos y sólidos de revolución. Permite la aplicación de operaciones booleanas para sumar, restar e interceptar partes de modelos, además de funcionalidades para la creación de piezas mecánicas. Utiliza como filosofía el dibujo paramétrico por lo que no cuenta con un sistema automatizado para la generación de árboles escalonados, problema que encontró solución en su homólogo Autodesk Inventor (7).

Desventajas y restricciones de uso:

- Código cerrado, que imposibilita la reutilización de código para nuevas soluciones.
- Solo disponible en la plataforma Windows.
- Por ser una aplicación CAD para el dibujo paramétrico no presenta aceleradores de diseño que permitan la generación de árboles escalonados.

1.6.2 Autodesk Inventor

Autodesk Inventor es otro de los sistemas desarrollados por la multinacional Autodesk para el dibujo en 2D y el modelado en 3D. Autodesk Inventor es un software reconocido a nivel internacional por sus amplias capacidades de edición, que hacen posible el dibujo digital de planos para mecánica y la recreación de imágenes en 3D; es uno de los programas más usados por arquitectos, ingenieros, diseñadores industriales y otros. Tiene como una de sus grandes ventajas contar con aceleradores de diseño, que tienen como objetivo agilizar el proceso de diseño de piezas conjuntas y complejas, además traen incluido un paquete de funcionalidades para la simulación, análisis y prueba de los mismos. Ejemplos de estos aceleradores son el generador de engranajes cilíndricos y el de árboles escalonados, que por su complejidad de diseño pueden en programas que no cuenten con este soporte demorar un tiempo considerable para su confección. El componente para la generación de árboles escalonados cuenta con múltiples funciones como el generador de diseño, generación de cálculos físicos, y de análisis gráfico, estas funciones proveen grandes detalles del modelado, útil en los procesos de análisis y diseño del producto, mientras que el generador de engranajes cilíndricos provee incluso de un sistema de simulación que permite el chequeo de la calidad del diseño al testear su funcionamiento.

A continuación se muestra una imagen del componente para la modelación de árboles escalonados de la herramienta Autodesk Inventor.

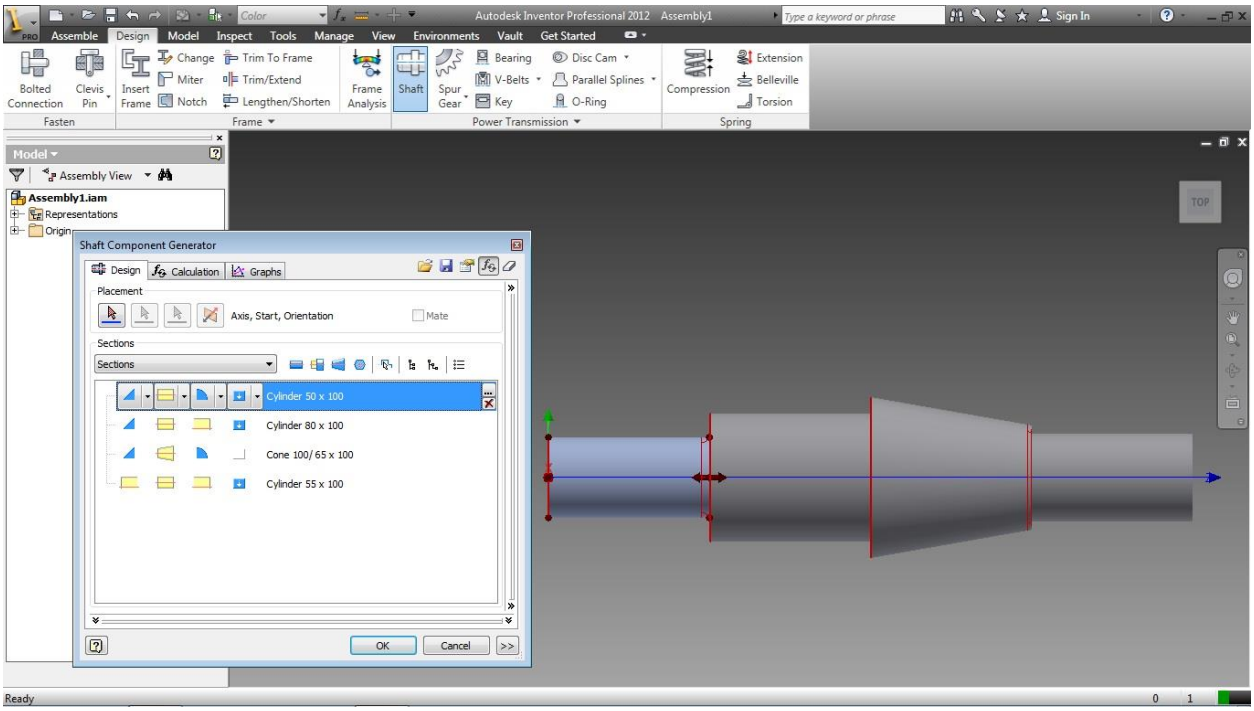


Figura 1. Generador de árboles escalonados de Autodesk Inventor.

Este sistema no solo tiene su popularidad por sus funciones sino por contar con soporte para una amplia colección de extensiones como DWG, DWF, DWFx, PDF, BMP, GIF, IGES, IPT, JPEG, JT, PNG, SAT, STEP, TIFF, STL, además de poseer extensiones compatibles para otros sistemas CAD como CATIA V5, Parasolid, Pro-Engineer. Podría considerarse como una opción viable para el país solo que presenta las siguientes restricciones:

- Código cerrado, que imposibilita la reutilización de código para nuevas soluciones.
- Solo disponible en sistemas de Microsoft Windows.
- Altos costos adquisitivos.
- Bloqueo norteamericano que prohíbe la distribución de este producto.

1.7 Solid Edge de PLM Siemens

Presentado en 1996, inicialmente fue desarrollado por la empresa Intergraph considerándose como uno de los primeros entornos basados en CAD para el sistema operativo Windows NT, posteriormente comprado por Siemens AG. Inicialmente contaba con un *kernel* de modelado geométrico llamado ACIS, pero fue cambiado posteriormente a Parasolid. El núcleo Parasolid es desarrollado actualmente por Siemens PLM software y es usado ampliamente como el motor geométrico de otras herramientas CADs como SolidWorks, IronCAD, MoldFlow, entre otros. Solid Edge es un sistema CAD para el modelado geométrico en 3D con estilo de trabajo paramétrico,

además de permitir el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros (1).

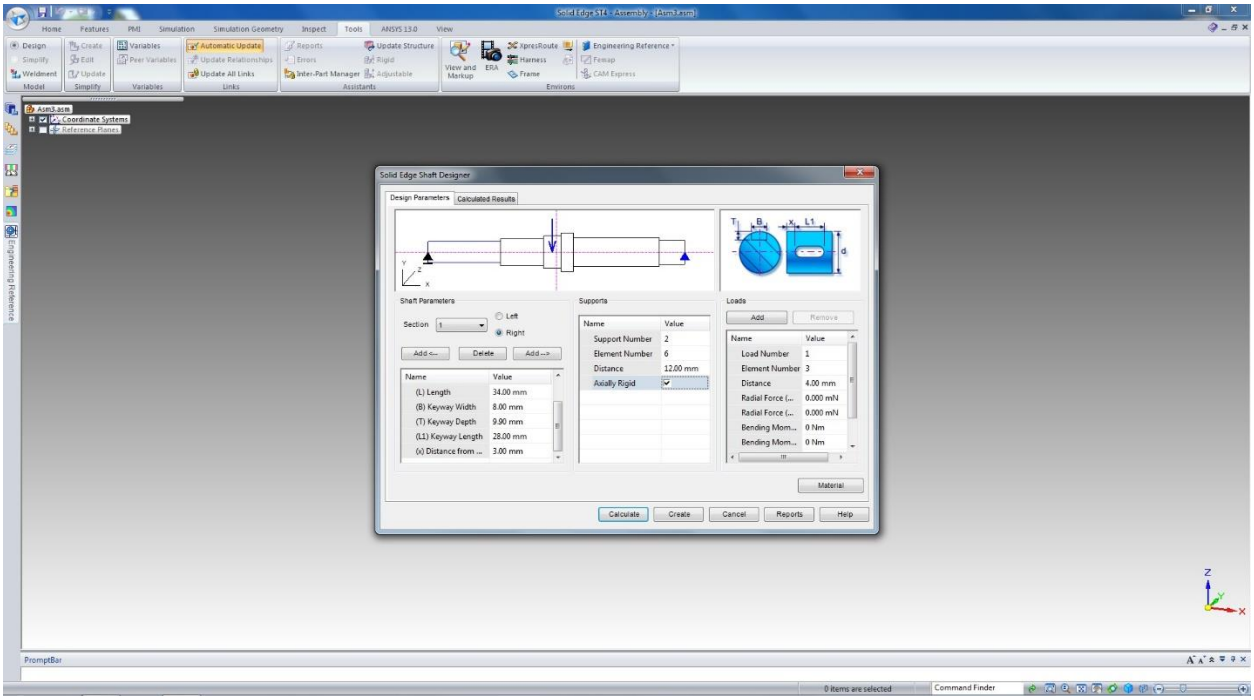


Figura 2. Generador de árboles escalonados en Solid Edge.

Desventajas y restricciones de uso:

- Código cerrado.
- Solo disponible en sistemas Windows.
- Altos costos adquisitivos.
- Bloqueo económico.

1.8 Software libre para el CAD

Estos sistemas permiten ampliamente la modificación, distribución e incluso la reutilización de su código fuente en otras aplicaciones por lo cual representan un pilar fundamental en el desarrollo de nuevas aplicaciones. A continuación se presentan lo que podrían ser en un futuro soluciones para el problema de la modelación automatizada de árboles escalonados y constituir una herramienta de reemplazo para los actuales sistemas propietarios.

1.8.1 FreeCad

Este sistema es uno de los más estables en cuanto a la cantidad de funcionalidades que posee para el diseño y el modelado 3D, permite la creación y modelación de entidades con tendencias similares a las de sistemas como Autodesk Inventor y CATIA para mencionar algunos, aunque

una de sus principales limitantes es la interfaz, también se podría señalar que el diseño de piezas se realiza de manera tradicional, similar al proceso en una mesa de dibujo, por lo que conformar entidades como árboles escalonados sería algo trabajoso y requeriría de mucho más tiempo. Es una aplicación de código abierto que puede ser modificado y utilizado de manera gratuita por toda la comunidad, además de estar basado en lenguaje C++ y utiliza como tecnología de modelado el *framework* de OpenCascade (OCAF, por sus siglas en inglés). Es multiplataforma por funcionando en sistemas operativos Windows y Linux. Posee una interfaz gráfica totalmente desarrollada con el *framework* de QT y un visor basado en Open Inventor, propiciando un renderizado rápido de las texturas en 3D. Cuenta con soporte para la exportación e importación a formatos tradicionales de otras aplicaciones como STEP, IGES, OBJ, STL, DXF, SVG, STL, DAE, IFC u OFF, NASTRAN, VRML además de la extensión nativa de FreeCad .fcstd y con un alto nivel de compatibilidad de cargar modelos de otros sistemas en dependencia de la complejidad de los mismos. Presenta las funcionalidades de *Undo* y *Redo* permitiendo recuperar un estado anterior y posterior en caso de errores (8).

Desventajas de uso:

- Es una herramienta paramétrica que no presenta aceleradores de diseño.

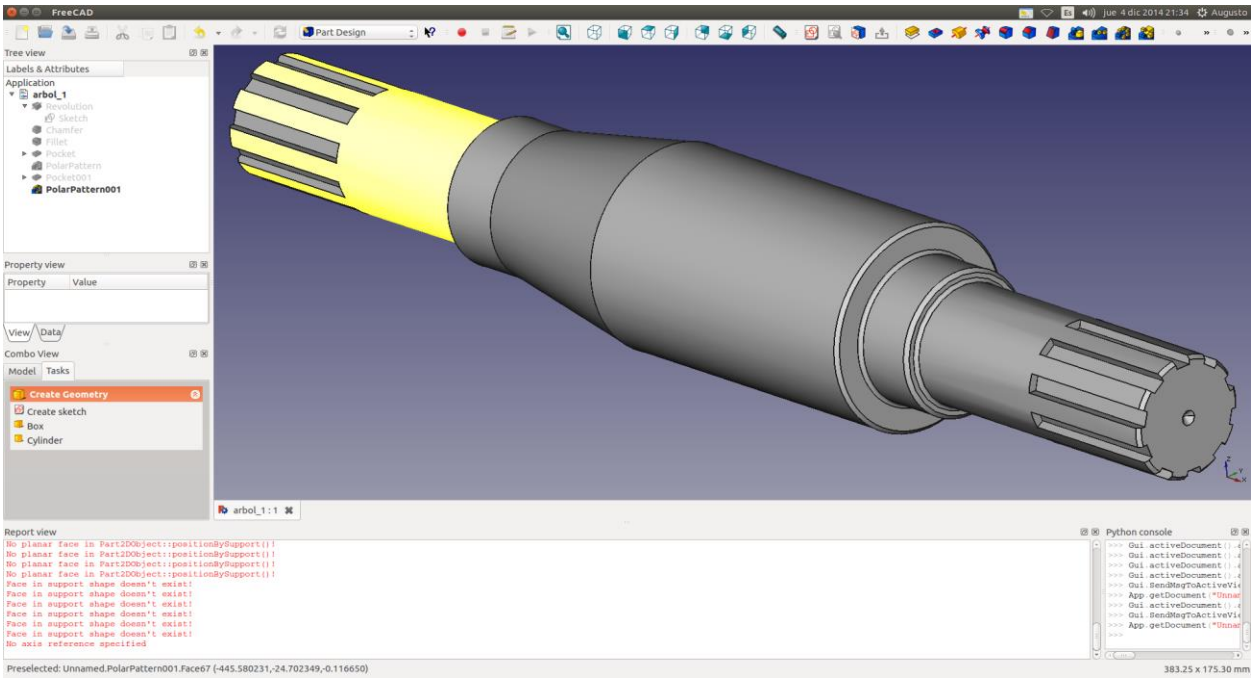


Figura 3. Árbol escalonado modelado paramétricamente en FreeCad.

1.8.2 LibreCAD

Libre CAD es una aplicación informática CAD de código libre para el diseño en 2D. La misma es multiplataforma funcionando en sistemas como GNU/Linux, Mac OS X, Solaris y Microsoft Windows. Gran parte de la interfaz y de los conceptos sobre su uso son similares a los del sistema AutoCAD, logrando que la misma sea más agradable para personas que tenga una larga trayectoria en el uso de estos sistemas CAD comerciales. Además cuenta con el uso del formato DXF de manera interna lo que proporciona que los diseños puedan ser importados y visualizados en sistemas propietarios como AutoCAD. Una de las principales limitaciones es que este sistema solo está concebido para el diseño en 2D lo que no proporciona una vista de modelado y por consiguiente de aceleradores de diseño como el de árboles escalonados (9).

Desventajas de uso:

- Es un sistema para el dibujo en dos dimensiones por lo que no es aplicable para el modelado en 3D.
- No cuenta con aceleradores de diseño para la modelación automatizada.
- Es un sistema totalmente paramétrico.

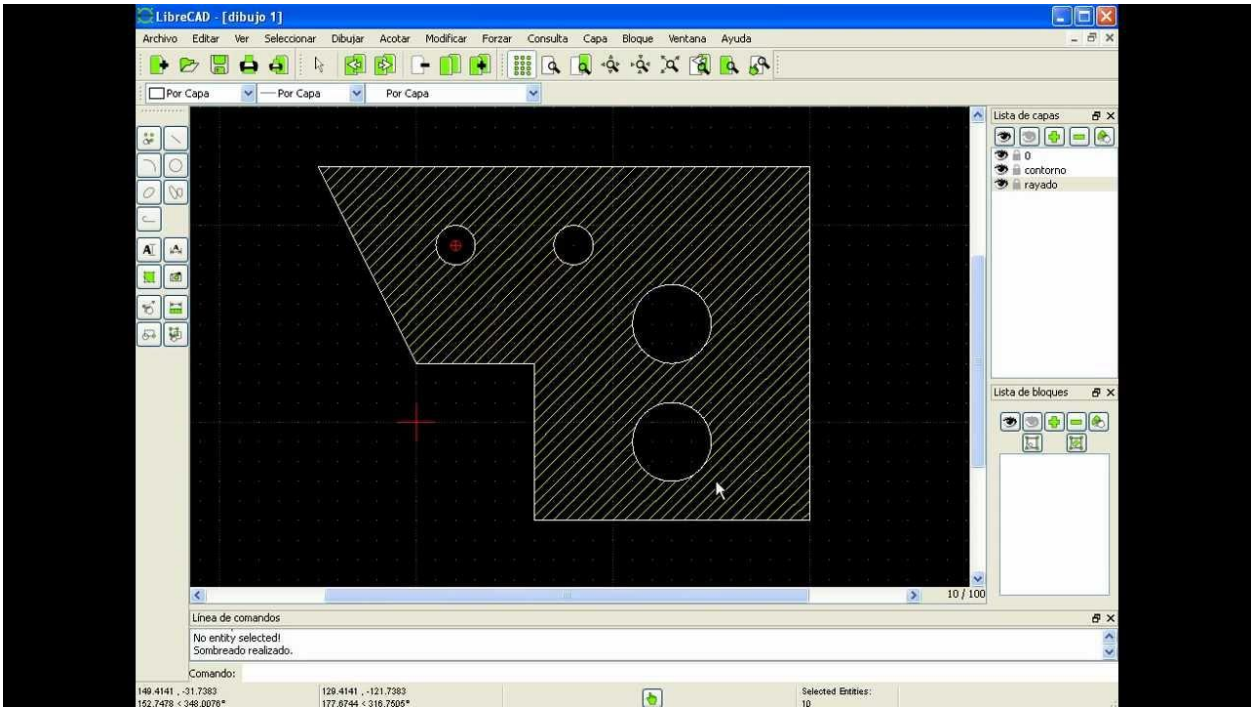


Figura 4. Dibujo 2D en LibreCAD.

1.9 Metodologías y herramientas de desarrollo

El proceso de desarrollo de software es sin dudas una tarea ardua y que incluye una serie de procesos y pautas que deben ser seguidas para lograr como resultado un sistema sólido y a la

altura de las exigencias de la sociedad actual. Como ayuda para esto ha surgido desde hace un tiempo las metodologías de desarrollo de software. Estas metodologías imponen un proceso disciplinado sobre el desarrollo de software, con el fin de hacerlo más predecible y eficiente. Se basan en una combinación de los modelos de procesos genéricos (cascada, evolutivo, incremental, entre otras), además de definir dentro de su marco de trabajo, artefactos, roles, actividades, junto con prácticas y técnicas recomendadas, que se une para la elaboración de un sistema robusto y eficaz (10).

1.9.1 Metodología SCRUM

SCRUM está considerada como una metodología ágil de desarrollo de software. Para la misma se aplicaron principios del proceso de control industrial aplicados a empresas como Microsoft, Borland, entre otras de igual renombre. Dentro de sus rasgos particulares se exponen los siguientes (11):

- Los equipos están auto dirigidos y auto organizados.
- El proceso de desarrollo consta de iteraciones de hasta 30 días, pero pueden ser más frecuentes.
- Se realiza el planeamiento directo con el cliente al inicio de cada iteración del proceso de desarrollo.

SCRUM al igual que todas las metodologías tanto ágiles como tradicionales presenta sus roles característicos como el SCRUM Master, el propietario de SCRUM, el equipo de desarrollo, el cliente, el manager y los usuarios. Está diseñado para equipos pequeños que no deberían superar los diez integrantes aunque su número ideal es siete, en el caso de equipos muy grandes que desean aplicar esta metodología, por lo general lo que se hace es dividir el grupo en subgrupos o lo que se denomina como SCRUMs de SCRUM, además cuenta con un ciclo de vida bastante dinámico haciendo que esta metodología sea excelente para el desarrollo rápido de aplicaciones.

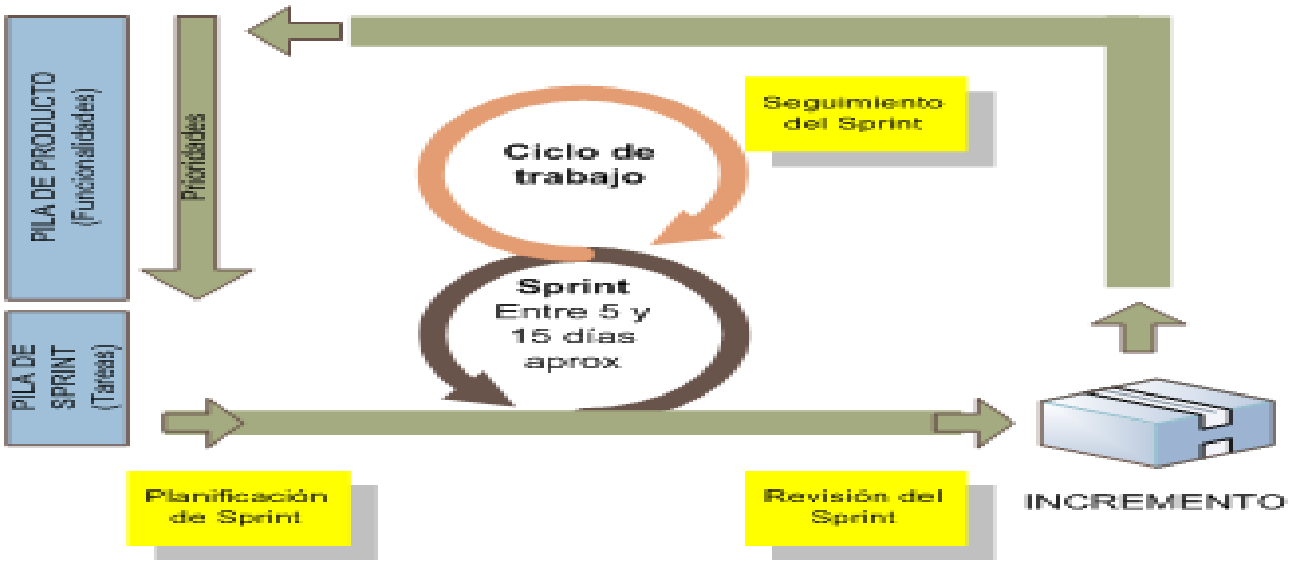


Figura 5. Ciclo de vida de la metodología SCRUM.

1.9.2 Metodología de Programación Extrema (XP)

La metodología Programación Extrema (XP, por sus siglas en inglés) es una variante de las metodologías ágiles que por sus características ha ganado gran aceptación por parte de la comunidad de programadores. Los fundamentos de esta metodología, según su creador Kent Beck son la de mejorar la comunicación, buscar la simplicidad, buscar retroalimentación y que siempre hay que proceder con valentía. Una de las herramientas más importantes en la metodología XP es el desarrollo orientado a pruebas, haciendo uso de las pruebas unitarias como eje de todo desarrollo. Las iteraciones suelen ser muy cortas y se promueve a los programadores buscar soluciones y experiencia con ellas, programar sin miedo a descomponer el sistema (12).

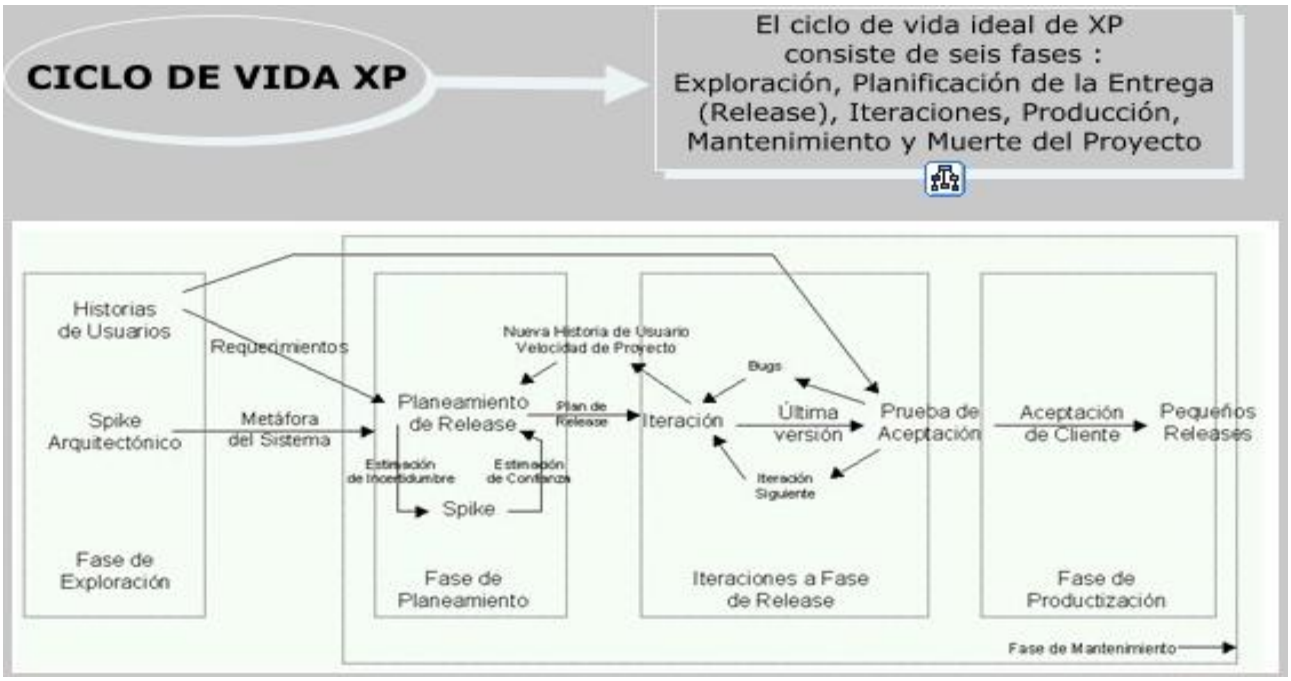


Figura 6. Ciclo de vida de la metodología XP.

1.9.3 Metodología SCRUM con XP

Basándose en el poder de gestión de SCRUM y los principios ágiles para el desarrollo de XP, se tomó como metodología de trabajo una mezcla de ambas, tomando de cada una las características y principios más relevantes para el proyecto, con el fin de lograr una metodología ágil pero fuerte para el desarrollo, capaz de realizar una buena gestión y organización del trabajo. De las mismas se tomarán los siguientes artefactos y principios que se muestran a continuación (13).

Artefactos y características tomadas de SCRUM:

- **Sprint Backlog:** Son subtareas en las que se divide la pila de producto realizándose en los llamados Sprints (13).
- **Sprint Goal:** Se definen después de cada iteración los resultados que se lograron en cada una.
- **Estimación de esfuerzo:** Cada historia de usuario de la pila del producto es estimada en lo que se denomina puntos de historia y los mismos se corresponden a días*personas ideales. Además puede ser calculada mediante criterio de expertos (13).

Artefactos y principios tomados de XP:

- **Historias de usuario:** Las historias de usuario, son descripciones cortas de una necesidad de un cliente del software que se quiera desarrollar. Su uso es común cuando se aplican marcos de trabajo ágiles, tales como Scrum o XP (14).

- **TDD(*Test Driven Development*):** Este es uno de los principios más importantes que se aplican de esta metodología, consiste en el desarrollo basado en pruebas, donde a cada funcionalidad se le realiza un diseño de prueba antes de ser desarrollado (13).
- **Desarrollo incremental:** Desarrollo llevado de lo simple a lo complejo, no queriendo desarrollar todo de un golpe sino ir perfeccionándolo poco a poco hasta lograr el resultado esperado (13).
- **Estandarización del código:** Proceso donde el equipo de trabajo define el estándar de código a seguir para lograr una comprensión general por parte de todos en el código (13).
- **Integración Continua:** Proceso para garantizar una compatibilidad entre todos los componentes de la aplicación. Se basa en la realización de pruebas constantes para detectar posibles errores e incompatibilidades (13).

1.10 Lenguaje de programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue fabricar una extensión del lenguaje de programación C, con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, siendo un lenguaje de alto nivel que está considerado potente al poder trabajar tanto en bajo como en alto nivel. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores) y de poder crear nuevos tipos que se comporten como tipos fundamentales. También presenta una biblioteca estándar muy poderosa. Es un lenguaje que con su estructura y diseño es considerado más eficiente que sus contrapartidas Java y C#. Por estas características se decidió tomarlo como lenguaje de programación además de que la mayoría de los *frameworks* de desarrollo para las aplicaciones de modelado 3D están basadas en este lenguaje, posibilitando una compatibilidad y adaptabilidad para el componente a desarrollar (15).

1.11 *Framework* de desarrollo

Un *framework* de desarrollo o marco de trabajo es una estructura conceptual y tecnológica de soporte que posee artefactos o módulos de software concretos y pueden servir de base para la organización y desarrollo de un producto informático determinado. Normalmente incluyen soporte de programas, bibliotecas, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto mediante un estilo común (16).

1.11.1 ACIS

El modelador ACIS es un *kernel* o motor geométrico para el modelado tridimensional. ACIS es utilizada en sistemas CAD, de animación 3D y astilleros al proveer todo lo necesario para el modelado en 3D. Posee una arquitectura abierta, desarrollada en lenguaje C++ y orientada a objetos lo cual le permite poseer capacidades que lo hacen más robusto y rápido. Con ACIS se pueden construir aplicaciones mixtas al integrar modelación de mallas de tipo *Wireframe* y modelado de sólidos, topologías con o sin variedad, además de una colección bastante extensa de operaciones geométricas (17).

Esta tecnología está dividida principalmente en tres categorías fundamentales:

Modelado de sólidos

- Extrusiones/Revoluciones/Barridos de conjuntos de curvas en 2D para crear superficies complejas o sólidos.
- Ajuste de superficies a conjuntos de curvas.
- Generación de patrones de formas repetidas.
- Sólidos huecos y engrosamiento de superficies.
- Doblado interactivo, torceduras, estiramientos y deformaciones de curvas, superficies y sólidos.
- Operaciones Booleanas de intersección/unión/diferencia de cualquier combinación de curvas, superficies o sólidos.

Gestión de modelado de sólidos

- Es posible adjuntar datos definidos por el usuario a cualquier nivel de un modelo.
- Rastrear cambios de geometría y topología.
- Modelado de sub-regiones de un sólido por medio de topología celular.
- Manejo de historia de construcción de modelo de manera independiente con capacidad de rehacer/deshacer cambios.

Modelador ACIS con extensiones

- Modelado de sólidos deformables.
- Cobertura avanzada.
- Descomposición de características.
- Remoción de líneas ocultas (tecnología 3D PHL V5), basado en tecnología CATIA V5.

Es una tecnología propietaria y siendo necesario adquirir una licencia para poder desarrollar productos con la misma, además de poseer un código cerrado lo que hace imposible la modificación y su distribución. Es multiplataforma por lo cual es posible su uso en plataformas de software libre (17).

1.11.2 Parasolid

Parasolid es una de las mejores tecnologías de modelado 3D actualmente disponible en el mundo. Esta tecnología posee recursos imbatibles de modelado 3D permitiendo manipular más modelos complejos y soportar niveles más altos de automatización en el modelado, administrar datos de modo preciso y consistente, y ofrecer la interoperabilidad necesaria para facilitar la integración confiable de datos 3D. Es una tecnología propietaria por lo que la adquisición de su código se encuentra inaccesible para libre distribución (18).

1.11.3 Tecnología de modelado OpenCascade

La tecnología OpenCascade es un paquete de desarrollo de software para implementar aplicaciones de modelado en 3D, es libre de costo y de código abierto. Además incluye una completa librería de clases desarrolladas en lenguaje C++ que proveen de servicios para la modelación y visualización de sólidos, además de clases pre implementadas para proveer de un desarrollo ágil. Cuenta con soporte multiplataforma siendo extensible indistintamente para cualquier sistema tanto UNIX como Windows y MAC (19).

1.11.4 *Framework* seleccionado

Tomando en cuenta lo anteriormente expuesto sobre los distintos *frameworks* de modelación tridimensional existentes y teniendo en cuenta que en el proyecto Sistema CAD 2D trabaja actualmente con la tecnología OpenCascade se decidió tomar este *framework* como el ideal para el desarrollo del componente para la generación automatizada de árboles escalonados ya que se estaría logrando así una total compatibilidad entre lo ya desarrollado y lo que se quiere desarrollar, además de tener en cuenta los factores anteriormente expuestos sobre el mismo.

1.12 Gestor de código fuente Doxygen

Doxygen es una herramienta utilizada para documentar el código fuente de aplicaciones basadas en C++, soporta otros lenguajes estándar como son C, C#, PHP, Java, Python, Fortran, entre otros (20). Este gestor puede ser utilizado para extraer documentación de archivos fuente de clases, métodos, además de ser útil para generar diagramas de herencia, de colaboración, per-

mitiendo generarlos de forma automática. Doxygen fue desarrollado para funcionar en plataformas como Mac OS y Linux y Windows, cuenta con licencia libre que esta accesible para su libre distribución en repositorios de estos sistemas. Actualmente es el sistema de gestión documental utilizado en el proyecto Sistema CAD2D por lo que será el gestor a utilizar en la futura documentación.

1.13 Entorno de desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es un programa informático compuesto por un conjunto de herramientas para programar, contando como mínimo con un editor, compilador, intérprete y depurador de uno o varios lenguajes de programación (21).

1.13.1 QtCreator

Qt Creator es un excelente IDE multiplataforma para desarrollar aplicaciones en C++ de manera sencilla y rápida (22). Como su nombre lo indica, está basado en la librería Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

El IDE de Qt presenta diversas ventajas técnicas ya que posee un *framework* que es utilizado por otras herramientas como FreeCAD que garantizaría una compatibilidad con estas herramientas para su reutilización en el proyecto.

1.14 Herramienta para el modelado

Las herramientas para la Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. Las mismas brindan ayuda a los analistas, ingenieros de software y desarrolladores. Permiten además el modelado de los sistemas mediante diferentes diagramas y generación de código a partir de éstos, y viceversa. La herramienta CASE seleccionada para el modelado de aplicación es *Visual Paradigm for* Unified Modeling Language (UML por sus siglas en inglé+s) (23).

1.14.1 Herramienta Visual Paradigm para el modelado UML

Es una herramienta profesional multiplataforma que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Soporta todos los diagramas UML y el Diagrama Entidad-Relación ayudando a una rápida construcción de aplicaciones con calidad y a un menor coste. Su interoperabilidad entre diagramas permite exportar diagramas de un modelo a otro con mucha facilidad. Pueden encontrarse múltiples usuarios trabajando sobre el mismo proyecto gracias a su integración con SVN (software empleado para el control de versiones). Produce documentación del sistema en formato PDF, HTML y MS Word. Genera código en una amplia gama de lenguajes, entre los que se encuentra Java. Posee una interfaz de uso intuitivo y profesional para el trabajo y se puede modelar en varios idiomas (24).

1.15 Conclusiones Parciales

En el presente capítulo se realizó un estudio sobre las soluciones existentes, así como las herramientas a utilizar para la implementación del componente. Siguiendo la línea base que define el propio sistema, se determinó utilizar como lenguaje de programación a **C++**, haciendo uso del *framework* de **Qt**, el IDE **QtCreator** y seleccionando como tecnología para el modelado tridimensional las librerías del *framework* **OCAF**. Como metodología de desarrollo se propuso utilizar una mezcla de Scrum con XP tomando de ambas los principios y artefactos más significativos, siendo la misma, adecuada según las características del equipo de trabajo y el corto plazo de entrega del producto que requirieron de técnicas ágiles de desarrollo.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

El objetivo principal de este capítulo es describir las funcionalidades del sistema que darán paso a la conformación de las Historias de Usuarios a implementar. Para dar cumplimiento a este objetivo haciendo uso de la metodología Scrum con XP se abarcarán las dos primeras etapas de planificación y diseño con el fin de conocer el alcance, estimar los tiempos de entrega de cada sprint y los artefactos que se generarán durante el posterior proceso de implementación.

2.1 Propuesta

Durante el proceso de desarrollo se pretende realizar una herramienta que funcione como sostén en los proyectos de diseñadores mecánicos, conformando una vía no paramétrica que acelere el proceso de diseño brindando funcionalidades para la modelación de árboles escalonados de una manera semi-automatizada. El componente contará con tres pestañas principales:

- **Diseño**: Brinda las herramientas para el diseño de la pieza.
- **Cálculo**: Realiza el proceso de cálculos físicos y matemáticos de la pieza.
- **Gráfica**: Muestra representaciones físicas del modelo mediante gráficas y modelos.

2.2 Modelo de dominio para el componente

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés que pueden mostrar asociaciones entre las clases conceptuales y sus atributos. Utilizando la notación UML, dicho modelo se representa con un conjunto de diagramas de clases en los que no se define ninguna operación (25).

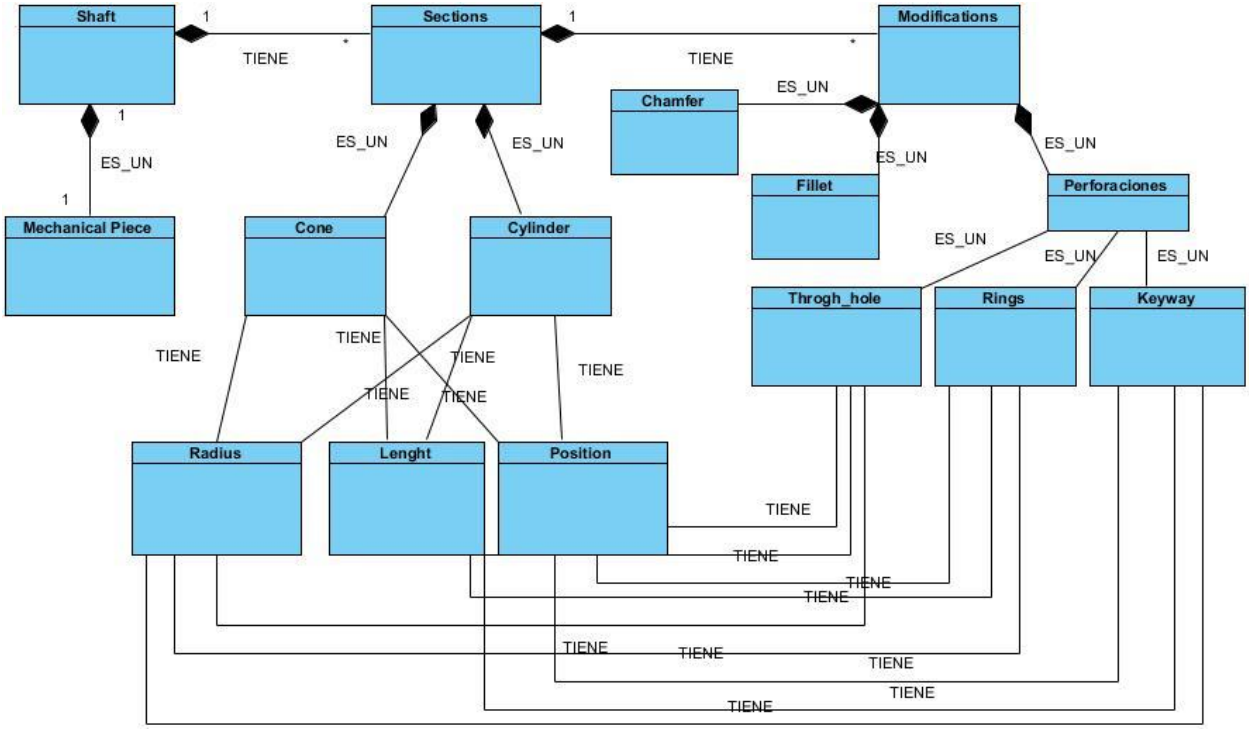


Figura 7. Modelo de dominio de la aplicación.

En el diagrama de dominio se aprecian los principales conceptos y las relaciones existentes entre cada uno, para lograr un mejor entendimiento del negocio. Dentro del mismo se representa Shaft como el árbol escalonado, siendo este una pieza mecánica compuesta por secciones que pueden ser cilíndricas o cónicas. Las secciones presentadas contienen restricciones de radio, largo y posición en el espacio. A las mismas se le pueden aplicar tantas modificaciones como se deseen que pueden ser de tipo achaflanado, fileteado, y perforaciones. Las perforaciones pueden ser anillas, agujeros diametrales, y chaveteras, siendo estos los elementos más significativos para el negocio según el cliente.

2.3 Etapa de planificación

Durante esta etapa se inicia el ciclo de vida de la metodología ágil SCRUM en la cual se recopilan todos los requisitos funcionales y no funcionales del componente, elaborando las historias de usuario que posteriormente serán implementadas por orden de prioridad durante los Sprints o iteraciones definidas por el equipo de proyecto y se realizará la estimación para la confección de la herramienta.

2.3.1 Roles que interviene en el sistema.

Tabla 1. Roles que intervienen en la aplicación.

Roles del sistema	
Rol	Descripción
Usuario	Es cualquier persona que utilice el sistema.

2.3.2 Lista de reservas del producto

En la metodología XP se define la lista de reservas del producto compuesta por los aspectos funcionales con los que debe contar la aplicación para satisfacer las necesidades requeridas por el cliente. Dentro de los requerimientos dados por el cliente se determinaron 6 aspectos funcionales que serán expuestos a continuación:

RF1: Gestionar sección cónica.

RF1.1 Insertar sección cónica.

RF1.2 Modificar sección cónica.

RF1.3 Eliminar sección cónica.

RF2: Gestionar sección cilíndrica.

RF2.1 Insertar sección cilíndrica.

RF2.2 Modificar sección cilíndrica.

RF2.3 Eliminar sección cilíndrica.

RF3: Gestionar inserción de anillas en las secciones cilíndricas.

RF3.1 Insertar anillas en sección cilíndrica.

RF3.2 Eliminar anillas en sección cilíndrica.

RF4: Gestionar agujeros diametrales en las secciones cilíndricas.

RF4.1 Insertar agujeros diametrales en sección cilíndrica.

RF4.2 Eliminar agujeros diametrales en sección cilíndrica.

RF5: Gestionar chaveteras en secciones cilíndricas.

RF5.1 Insertar chaveteras en secciones cilíndricas.

RF5.2 Eliminar chaveteras en secciones cilíndricas.

RF6: Fusionar secciones.

2.3.3 Historias de usuario

Las Historias de Usuario (HU) representan una breve descripción del comportamiento del sistema, descrito en un lenguaje natural de fácil entendimiento para el cliente y el desarrollador. Además se emplean para hacer estimaciones de tiempo y el plan de lanzamientos, reemplazando así un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Tabla 2. HU 1. Insertar sección cónica.

Historia de Usuario	
Funcionalidad: RF1.1	
Número: 1	**Nombre:** Insertar sección cónica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Alta.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Funcionalidad: 1	
Descripción: Se deben insertar secciones cónicas en el visor al presionar los botones de inserción de la interfaz principal. Figura 10	
Validación: Al presionar el botón de cono, este debe permitir insertar tantas entidades como se desee de este tipo de sección.	
Prototipo de Interfaz de Usuario (IU): 	

Tabla 3. Modificar sección cónica.

Historia de Usuario	
Funcionalidad: RF1.2	
Número: 2	**Nombre:** Modificar sección cónica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Medio.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Se debe poseer un área de trabajo que permita cambiar las dimensiones del cono en pantalla así como orientar su tronco de cono.	
Validación: El usuario puede determinar la dirección de los troncos de cono, así como sus dimensiones mediante una caja de dialogo.	
Prototipo de IU:	

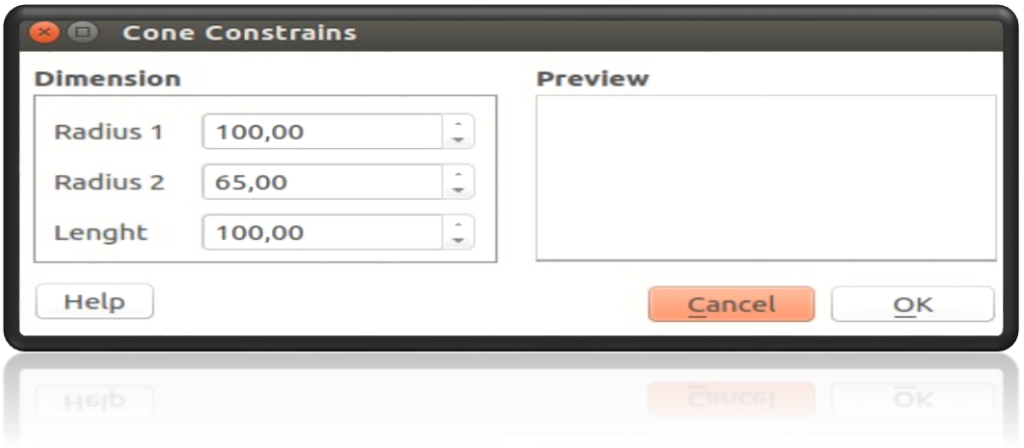


Tabla 4. HU 3. Eliminar sección cónica

Historia de Usuario	
Funcionalidad: RF1.3	
Número: 3	**Nombre:** Eliminar sección cónica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Se debe poseer un área de trabajo que permita eliminar el cono en pantalla.	
Validación: El usuario puede mediante un botón de la interfaz eliminar la sección cónica seleccionada.	
Prototipo de IU:	

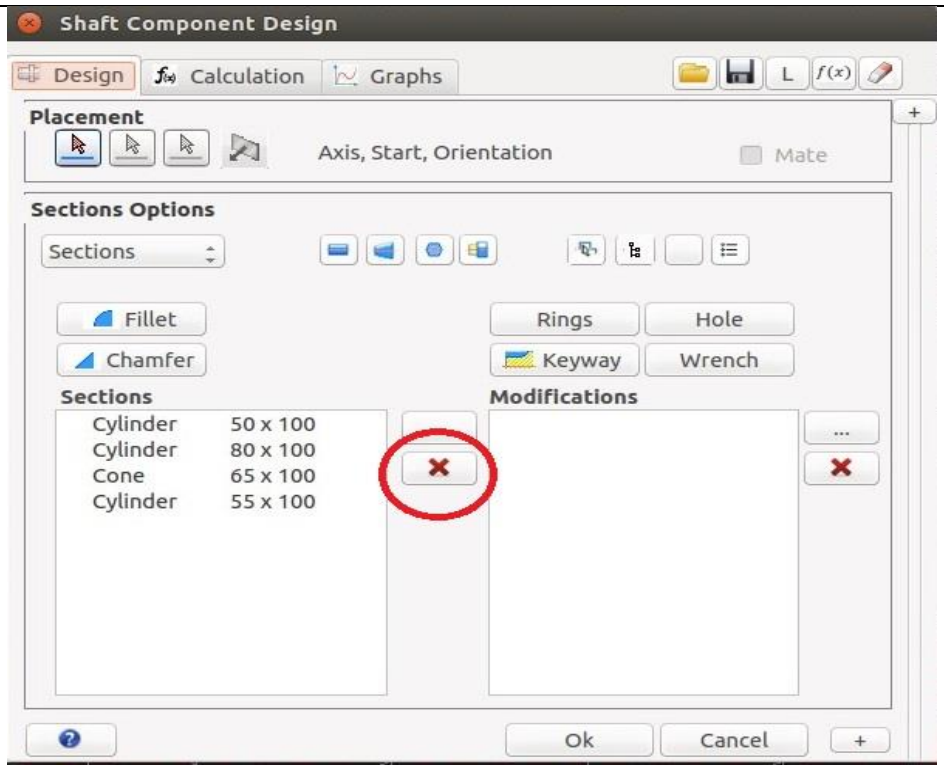


Tabla 5. HU 4. Insertar sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF2.1	
Número: 4	**Nombre:** Insertar sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Alto.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede ser capaz de insertar tantas secciones cilíndricas como desee.	
Validación: Al presionar el botón de cilindro se inserta una sección cilíndrica en la pieza principal.	
Prototipo de IU:	

Tabla 6. HU 5. Modificar sección cilíndrica.

Historia de Usuario
Funcionalidad: RF2.2

Número: 5	**Nombre:** Modificar sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Medio.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede modificar las dimensiones de cualquier sección cilíndrica seleccionada.	
Validación: Al presionar el botón de gestión se debe mostrar las características de la sección cilíndrica seleccionada y permitir modificar sus dimensiones.	
Prototipo de IU: 	

Tabla 7. HU 6. Eliminar sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF2.3	
Número: 6	**Nombre:** Eliminar sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Medio.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede eliminar cualquier sección cilíndrica seleccionada.	
Validación: El usuario puede mediante un botón de la interfaz eliminar la sección cilíndrica seleccionada.	
Prototipo de IU:	

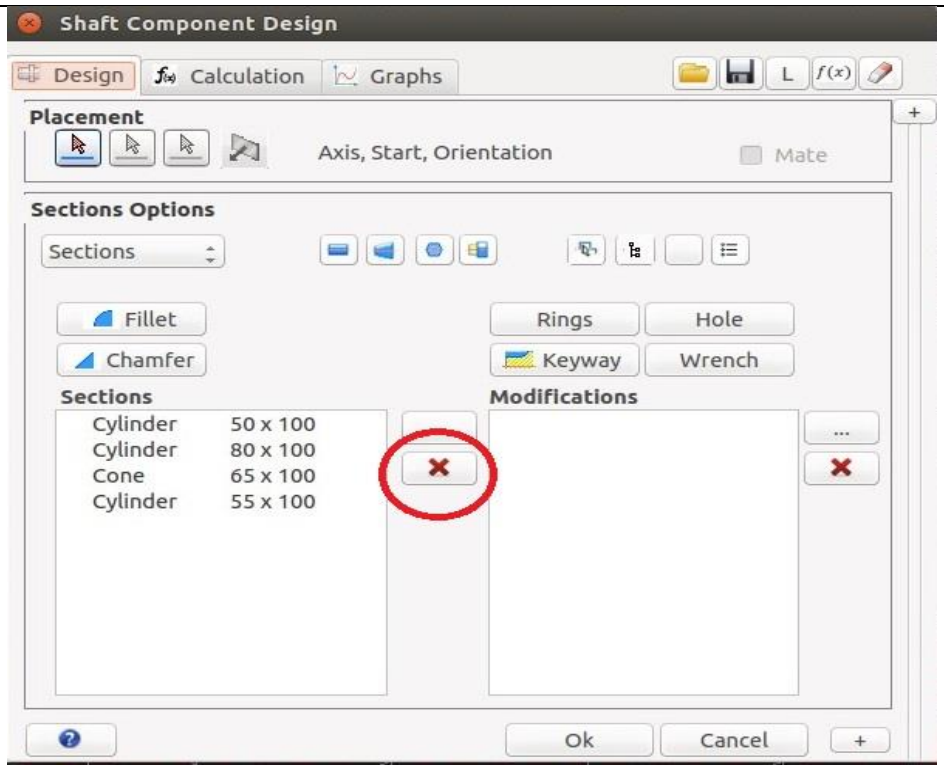


Tabla 8. HU 7. Insertar anilla en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF3.1	
Número: 7	**Nombre:** Insertar anillas en sección cilíndrica.
Prioridad del Negocio: Alta	Riesgo en desarrollo: Medio.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Mediante una caja de dialogo se debe definir la posición con respecto al inicio de la sección seleccionada, diámetros y ancho de la anilla que se va a insertar.	
Validación: Al presionar el botón de inserción de anillas se muestra una caja de dialogo que permite establecer todas las propiedades de la misma dada la sección seleccionada.	
Prototipo de IU:	

Tabla 9. HU 8. Eliminar anilla en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF3.2	
Número: 8	**Nombre:** Eliminar anilla en sección cilíndrica.
Prioridad del Negocio: Alta	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede eliminar la anilla seleccionada.	
Validación: Al presionar el botón de eliminar se elimina la anilla de la sección cilíndrica.	
Prototipo de IU:	

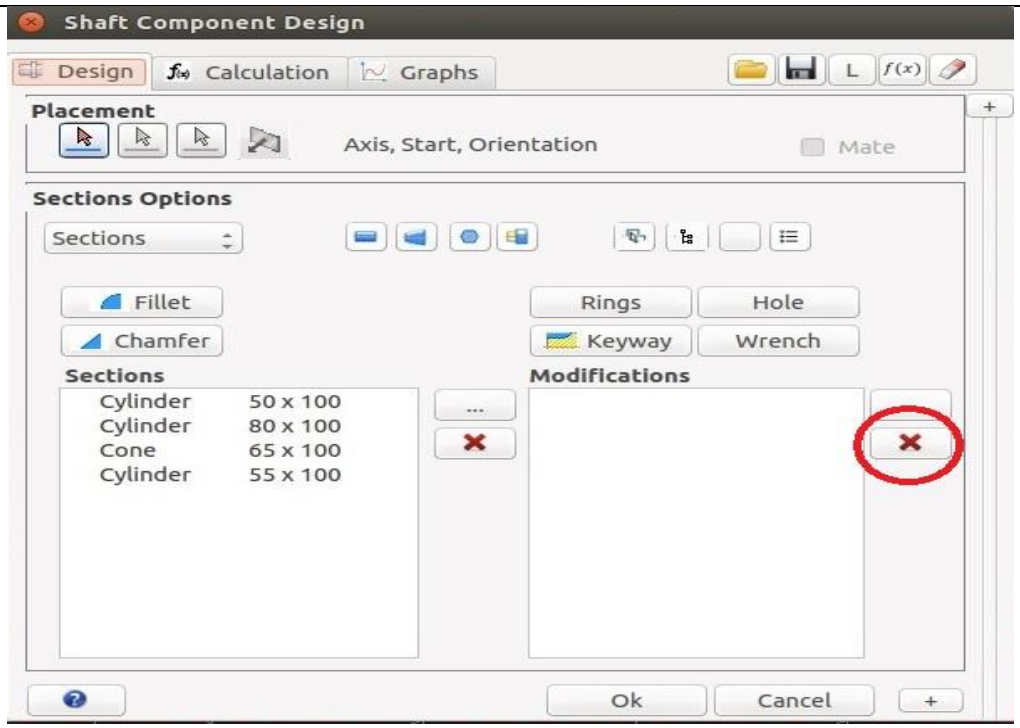


Tabla 10. HU 9. Insertar agujero diametral en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF4.1	
Número: 9	**Nombre:** Insertar agujero diametral en sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Mediante una caja de dialogo se debe definir las características del agujero diametral que se va a insertar en la sección seleccionada.	
Validación: Al presionar el botón de inserción de agujeros diametrales se muestra una caja de dialogo que permite definir las características del agujero a insertar en cuanto a su diámetro, posición con respecto al inicio de la sección, entre otras características.	
Prototipo de IU:	

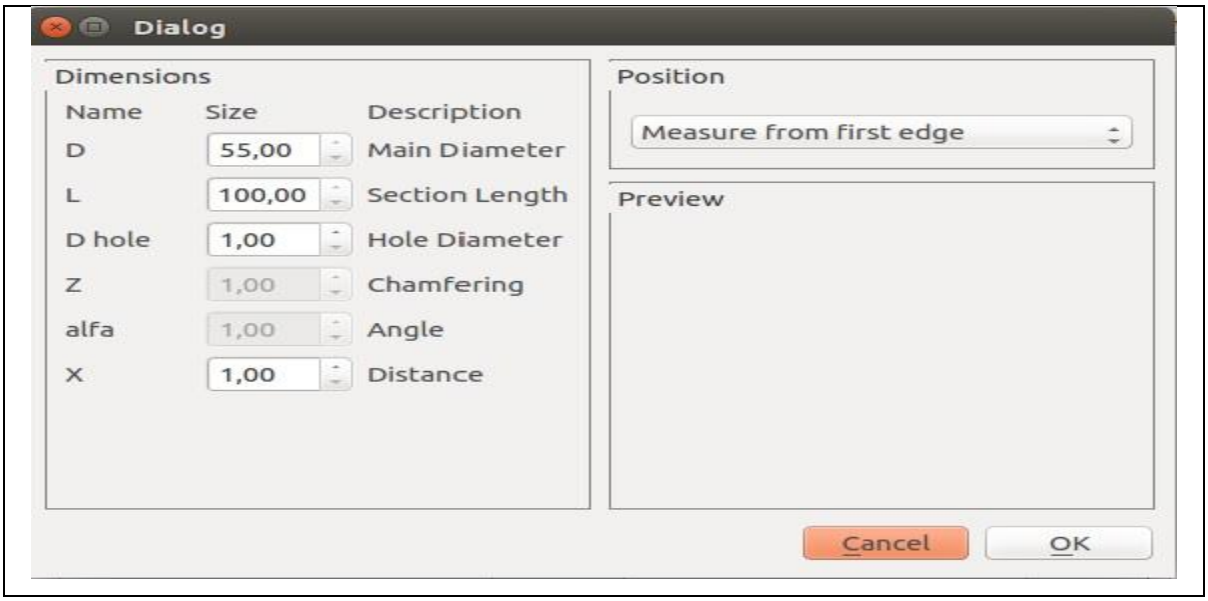


Tabla 11. HU 10. Eliminar agujero diametral en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF4.2	
Número: 10	**Nombre:** Eliminar agujero diametral en sección cilíndrica.
Prioridad del Negocio:	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede eliminar el agujero diametral seleccionado.	
Validación: Al presionar el botón de eliminar se elimina el agujero diametral seleccionado por el usuario.	
Prototipo de IU:	

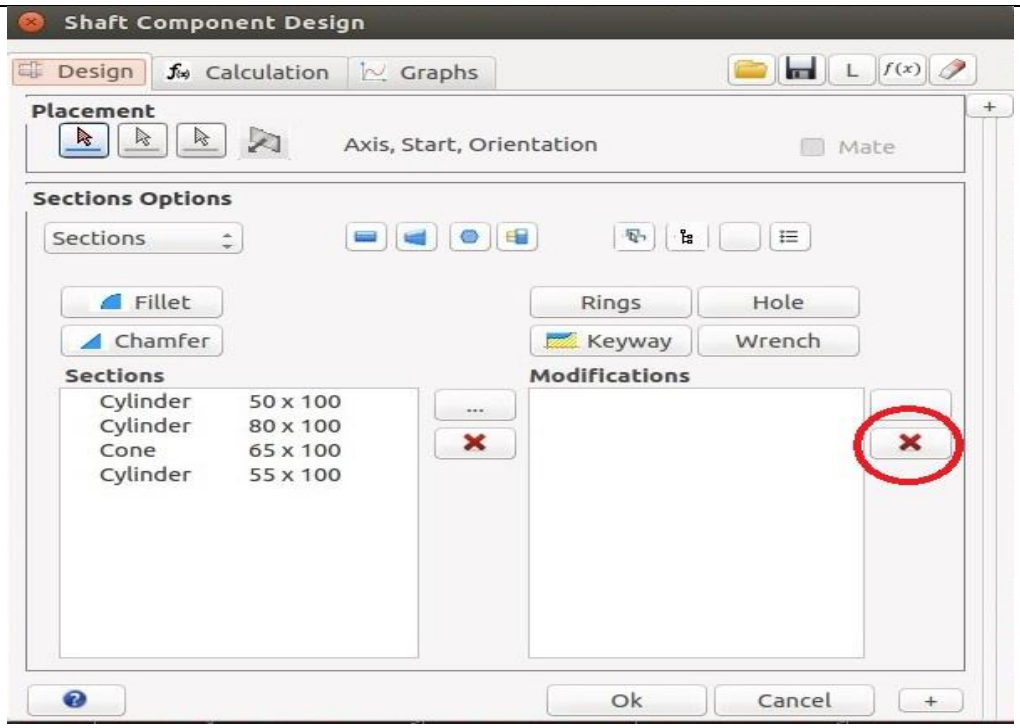


Tabla 12. HU 11. Insertar chaveteras en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF5.1	
Número: 11	**Nombre:** Insertar chaveteras en sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Alto.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Mediante una caja de dialogo se debe definir la cantidad de chaveteras que se van a insertar en la sección seleccionada, así como sus características.	
Validación: Se muestra una caja de dialogo en la cual se muestran las posibles características que puede tener la chavetera, la cantidad, longitud entre otras opciones.	
Prototipo de IU:	

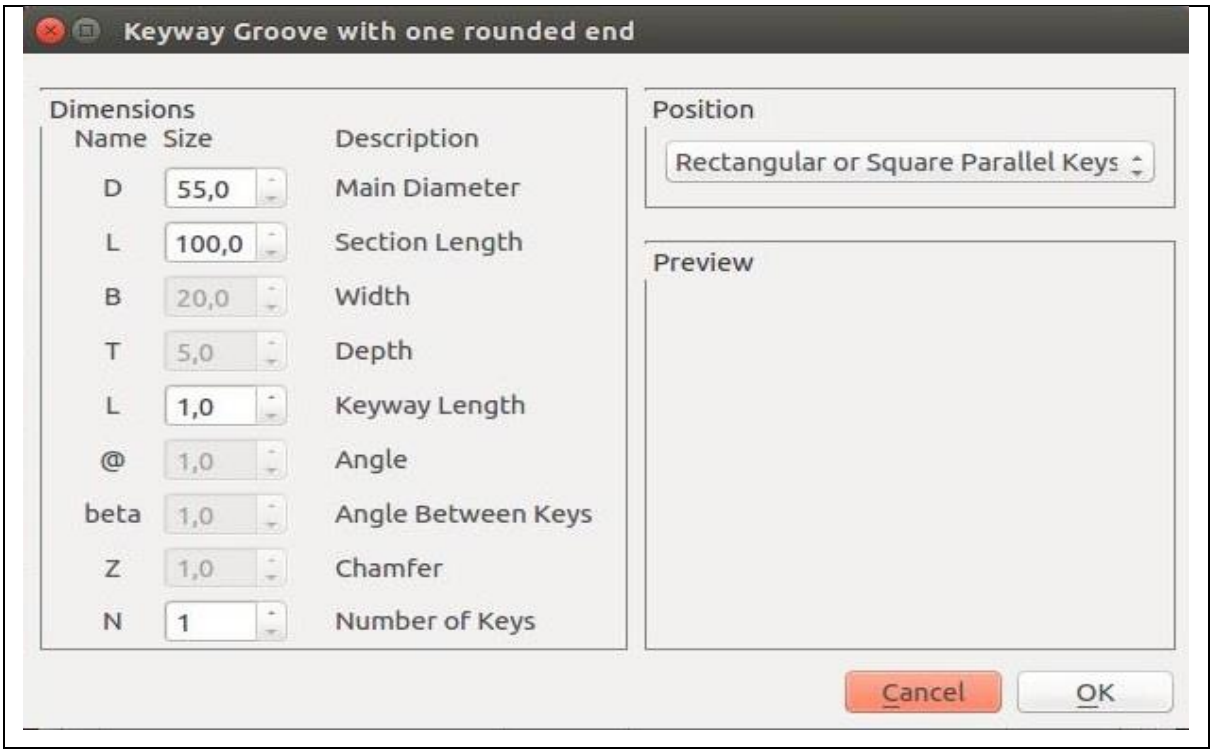


Tabla 13. HU 12. Eliminar chaveteras en sección cilíndrica.

Historia de Usuario	
Funcionalidad: RF5.2	
Número: 12	**Nombre:** Eliminar chaveteras en sección cilíndrica.
Prioridad del Negocio: Alta.	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: El usuario puede eliminar las chaveteras en cualquier sección cilíndrica de la pieza.	
Validación: Se eliminar la chavetera seleccionada al presionar el botón eliminar.	
Prototipo de IU:	

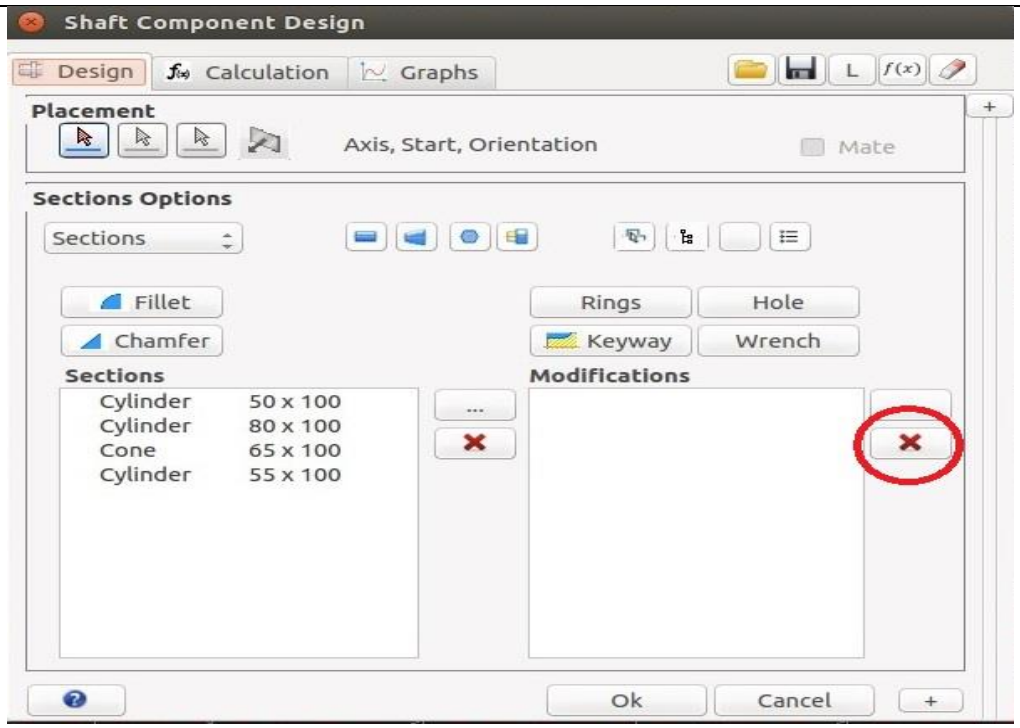
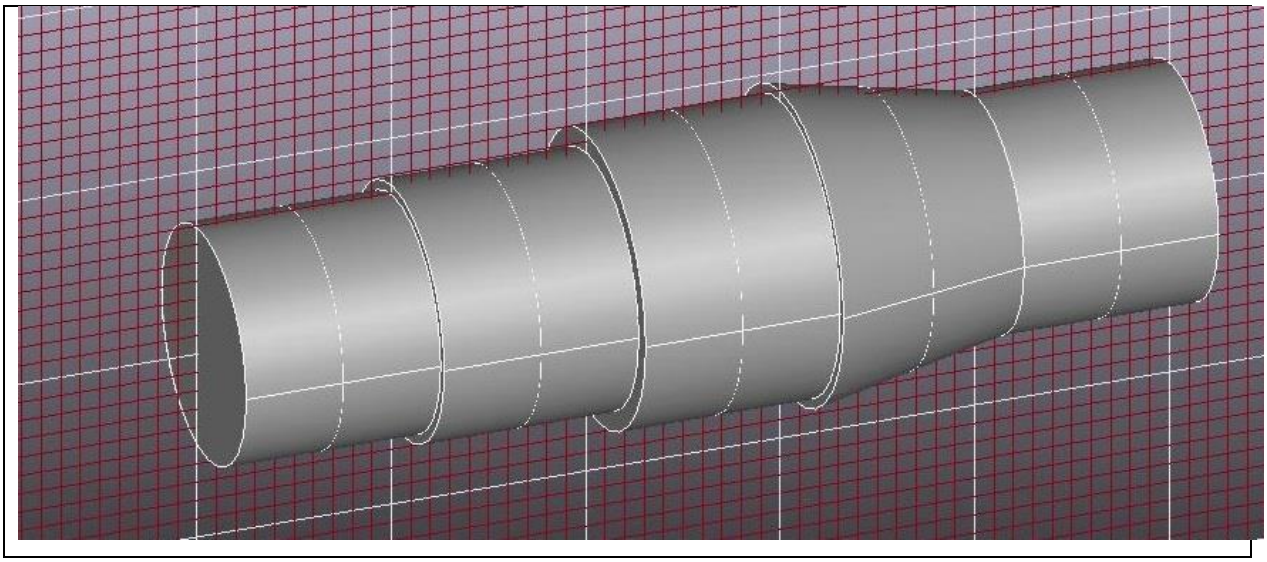


Tabla 14. HU 13. Fusionar secciones.

Historia de Usuario	
Funcionalidad: RF6	
Número: 13	**Nombre:** Fusionar secciones.
Prioridad del Negocio: Alto.	Riesgo en desarrollo: Bajo.
Tiempo estimado (días):	
Programador: Abel Fernández Ferrer.	
Descripción: Una vez terminado el diseño del árbol se debe fusionar la pieza en una sola entidad.	
Validación: Al determinar las características del árbol se termina el diseño y se muestra solamente la pieza generada con todas las modificaciones aplicadas.	
Prototipo de IU:	



2.3.4 Aspectos no funcionales del producto

Los **requisitos no funcionales** son aspectos técnicos con características y restricciones que debe cumplir el software.

1. **Software**

- El componente es compatible sobre plataformas GNU principalmente en plataformas como Debian, Ubuntu y Nova, ambos en versiones actualizadas.
- Se requieren las librerías básicas disponibles en los repositorios de Linux: libtk-dev, libtcl-dev, libxmu-dev, freeglut3-dev, libopencascade-dev, asymptote.
- Debe contar con las librerías del paquete **OCE, Python** y **EIGEN** desplegadas en el directorio ***opt*** de Linux.

2. **Hardware**

- Memoria RAM: Aproximadamente 512Mb para un funcionamiento óptimo de la aplicación, 128mb de requerimiento mínimo.
- Espacio mínimo en disco duro: 1Gb.
- Microprocesador: Requerimiento Mínimo: Intel Pentium III a 1.0Ghz y como requerimiento óptimo Intel Pentium IV o superior.

3. **Portabilidad**

- El componente debe ser compatible con otras plataformas como Salome y FreeCAD.

4. **Aspectos Legales**

- El componente una vez terminado deberá ser de código abierto y bajo licencia GPL.

5. **Aspecto o interfaz externa**

- La aplicación debe poseer una interfaz con características similares a las utilizadas por los sistemas Autodesk Inventor y Solid Edge para garantizar una familiaridad por parte de los usuarios finales.

6. Políticos Culturales

- La interfaz debe estar desarrollada en idioma inglés.

2.3.5 Estimación

Las estimaciones del esfuerzo para implementar la duración de cada sprint permiten tener una medida bastante real de la velocidad de progreso del proyecto y brindan una guía razonable a la cual ajustarse. En este caso basándose en la experiencia del equipo de desarrollo se estimó de manera empírica de acuerdo al criterio de expertos.

Tabla 15. Estimaciones por sprint.

Sprint	Historia de usuario	Tiempo estimado (días)
1	Gestionar secciones Cónicas.	9
2	Gestionar secciones Cilíndricas	9
3	Gestionar inserción de anillas en las secciones cilíndricas.	8
4	Gestionar agujeros diametrales en las secciones cilíndricas. Fusionar secciones.	10
5	Gestionar chaveteras en las secciones cilíndricas.	8

2.4 Etapa de 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 modela el sistema y para que soporte todos los requerimientos, 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, además ofrece una vista previa de cómo se debe estructurar el componente para lograr un funcionamiento óptimo y estable.

2.4.1 Arquitectura de software basada en capas

El estilo basado en capas define su estructura 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 (26).

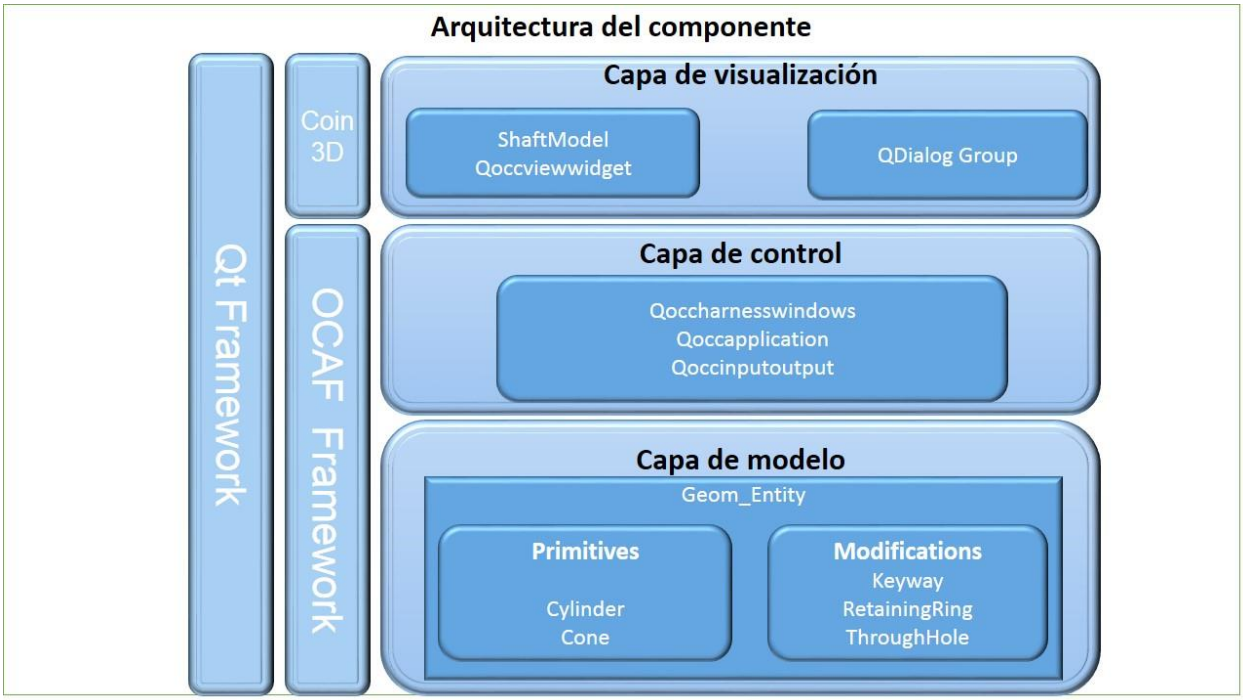


Figura 8. Arquitectura definida en el componente árboles escalonados.

Dentro del componente se han establecido 3 capas principales denominados *Control*, *Modeling Data* y *Visualization*, estas conforman todas las funciones necesarias para el correcto funcionamiento de la aplicación. La capa de control contiene la clase **ShaftControl** como clase controladora principal, siendo esta la encargada de recibir y enviar las instrucciones necesarias para todas las funciones del componente. La capa de modelo contiene las clases geométricas en la cual se tiene a **Geom_Entity** como clase abstracta, todas las demás clases heredaran sus propiedades, además de poseer el método polimórfico BuiltGeometry () que genera las entidades geométricas de acuerdo al tipo que estas sean. Por último la capa de visualización cuenta con todas las interfaces requeridas por el componente en cuestión.

Dentro de las relaciones existentes todos los paquetes implementan de una manera u otra el *framework* de Qt ya que la aplicación está desarrollada sobre esta plataforma. Las librerías de Coin3D serán las que darán soporte para la representación de las entidades e implementará el visor principal de la aplicación, estableciendo solo sus relaciones con el paquete de visualización del componente, mientras que la capa de modelo y control implementarán las funciones del *framework* de OpenCascade para las funciones de persistencia de datos, funciones ***deshacer*** y ***rehacer***, además de exportación e importación de datos.

2.4.2 Patrones de diseño

Un patrón de diseño según Larman no es más que "…una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas" (25).

Se clasifican en patrones GRASP y GOF. Los patrones GRASP describen principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Mientras que los patrones GOF se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

2.4.3 Patrones GRASP

GRASP es un acrónimo que significa *General Responsibility Assignment Software Patterns*. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos para identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (27). Se evidencia el uso de este en clase **QoccHarnesswindows** en los métodos **Built_Cone ()** y **Built_Cilinder ()** que se encargan de la creación de instancias de conos y cilindros.
- **Experto:** Asigna una responsabilidad al experto en información, es decir a la clase que tiene la información necesaria para realizar la responsabilidad (27). En el paquete **ModelingData** la clase **Cylinder** contiene la información necesaria para mediante el método **Make_Cylinder ()** construir un cilindro.
- **Controlador:** Es el responsable de controlar todo el flujo de información entre todas las clases del sistema y es un objeto que no pertenece a la interfaz de usuario (27). La clase controladora **QoccHarnesswindows** es la encargada de invocar la vista principal mediante el método **ShaftGenerator ().**
- **Bajo Acoplamiento:** Asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. La clase controladora principal es la única que accede a todas las clases del modelo e invoca a las vistas de modo tal que ninguna vista se relaciona con el modelo y viceversa (27).
- **Polimorfismo:** El polimorfismo viene siendo una vía de abstracción en la cual una misma funcionalidad tiene diferente implementación o efecto distinto en la aplicación en dependencia del objeto que lo implemente (27) (28). Este patrón es utilizado en el paquete *ModelingData* en el método **BuiltGeometry ()** de la clase **Geom_Entity** el cual es implementado de forma distinta en cada una de sus clases hijas.

- **Alta Cohesión:** Este patrón resuelve el problema de asignar una responsabilidad de manera que la cohesión permanezca alta. La cohesión es una medida de la fuerza con que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión (29).En el diseño del componente se tuvo en cuenta este patrón.

2.4.4 Diagrama de clases del componente

Los Diagramas de Clases del Sistema describen la arquitectura y las relaciones entre cada una de estas clases. Es la vista más conceptual y detallada de la solución. A continuación se representa el diagrama de clases que fue concebido por la arquitectura seleccionada y que representan la base del componente propuesto.

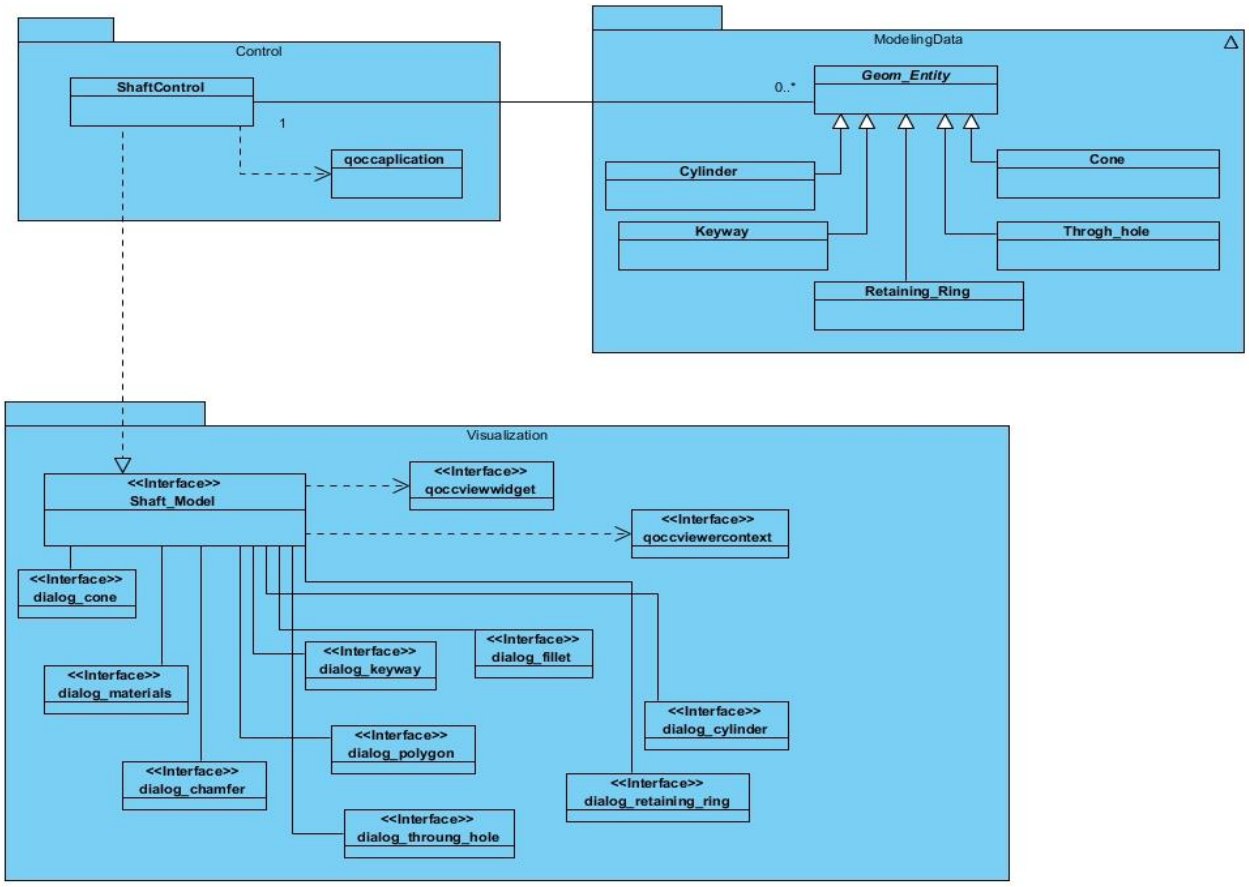


Figura 9. Diagrama de clases del componente

En el diagrama de clases propuesto se identifican tres paquetes de clases principales los cuales responden a la arquitectura definida para la elaboración del componente. Este diagrama contiene

el paquete ***Control***, donde está contenida la clase **ShaftControl** como clase principal y tiene la función de comunicar y ejecutar la inserción de secciones geométricas haciendo uso de las vistas y el modelo así como manejar las funciones de gestión de modificaciones. El paquete de modelo contiene todas las entidades necesarias para la creación de un árbol escalonado basado en conos, cilindros y sus modificaciones de tipo chaveteras, agujeros diametrales y anillas. Dentro de ***Visualization*** se encuentran todas las clases interfaz que garantizan una comunicación fluida entre el usuario y el componente.

2.5 Conclusiones Parciales

En este capítulo se ejecutaron las tareas y se generaron los artefactos definidos para la etapa de planificación según el marco de trabajo de XP conformándose las Historias de Usuario a partir de las necesidades funcionales del cliente, además de definir la arquitectura y los patrones de diseño a aplicar para la confección de la solución propuesta. Con este análisis previo, se determinó la estimación de entregas mediante el criterio de expertos atendiendo a la complejidad de las funcionalidades definidas y se prepararon las condiciones para el posterior proceso de implementación siguiendo las especificaciones de los artefactos desarrollados.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

En el presente capítulo se exponen los componentes que integran las etapas de implementación y prueba de la aplicación. Siguiendo las etapas que presentan las metodologías de desarrollo **SCRUM** y **XP**, el proceso de implementación en las mismas se realiza de forma iterativa realizándose pruebas de funcionalidad al final de cada iteración obteniendo así una nueva versión del producto. Finalmente se procede a realizar de conjunto con el cliente las pruebas de aceptación del sistema, verificando que se cumplen todas las funcionalidades especificadas.

3.1 Estándar de codificación utilizado

Los estándares de codificación se utilizan durante la fase de implementación de un software para darle una estructura al código facilitando su comprensión y mantenimiento. La metodología *Extreme Programming* exige el uso de este principio de programación para que el código sea legible y exista una buena comunicación entre programadores, de tal forma que se logre disminuir los riesgos de introducir errores que no son detectados por lo compiladores, reduciendo el tiempo y coste de las actividades de refinación y pruebas necesarias para la detección y corrección de los mismos (14). A continuación se muestra el estándar a utilizar en la implementación:

Tabla 16. Estándar de desarrollo.

Definición de Objetos, Clases, funciones y atributos	
Descripción	Ejemplo

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

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

por la normativa CamelCase-lowerCamel-Case.	
Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato> atributo; <Tipo dato> atributoNombreCompuesto;
Definición de parámetros dentro de las funciones y constructores de clases	
Descripción	Ejemplo
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.	<Tipo dato retorno> funcion(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Los identificadores de los parámetros dentro de los constructores de las clases deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCase-lowerCamelCase.	Clase(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Definición de expresiones	
Descripción	Ejemplo

Para una mejor comprensión en la lectura y legibilidad del código los operadores binarios exceptuando los punteros, función de llamado a miembros, escritura de un arreglo y paréntesis de una función se escribirán con un espacio entre ellos.	x + y;
	x == y;
	idFuncion.miembro();
	idFuncion->miembro();
	array[];

Definición de estructuras de control y bucles	
Descripción	Ejemplo

Las estructuras de control y los bucles estarán definidos de igual manera en ambos casos siguiendo el estándar determinado por el *framework* de Qt.	Para las estructuras if, else, if else :
	
	<estructura control>(condición){
	tarea a ejecutar
	}

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

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

3.2 Plan de Sprints

La pila del sprint, es la lista que descompone las funcionalidades de la pila del producto en las tareas necesarias para construir un incremento o una parte completa y operativa del producto. Cada tarea de la pila del sprint tiene asignada una persona, y la indicación del tiempo necesario para terminarla. 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. Es también una herramienta de soporte para la comunicación directa del equipo (13).

3.2.1 Sprint 1

Durante esta primera iteración se pretende comenzar a trabajar en base a los requisitos fundamentales que el cliente necesita para el negocio en cuestión. En la misma se desarrollará la funcionalidad Gestionar Secciones Cónicas, una de las estructuras fundamentales para la modelación de árboles escalonados.

Tabla 17. Sprint 1. RF Gestionar secciones cónicas. Tarea 1

Sprint: 1	Inicio: 6/4/2015	Fin: 7/4/2015	Duración (días): 1	**Funcionalidad**: Gestionar secciones cónicas.
Número tarea	Nombre tarea		Responsable	Estado
1	Crear interfaz de gestión para sección cónica.		Abel Fernández Ferrer.	Completado
Descripción: Crear una interfaz que permita gestionar las dimensiones de la sección cónica seleccionada.				

Tabla 18. Sprint 1. RF Gestionar secciones cónicas. Tarea 2

Sprint: 1	Inicio: 8/4/2015	Fin: 10/4/2015	Duración (días): 3	**Funcionalidad**: Gestionar secciones cónicas.
Número tarea	Nombre tarea		Responsable	Estado
2	Programar métodos para la actualización del visor.		Abel Fernández Ferrer.	Completado
Descripción: Se provee un método para actualizar la vista de cada sección cónica modificada repintando la misma en el visor con las nuevas características.				

Tabla 19. Sprint 1. RF Gestionar secciones cónicas. Tarea 3

Sprint: 1	Inicio: 13/4/2015	Fin: 13/4/2015	Duración (días): 1	**Funcionalidad**: Gestionar secciones cónicas.
Número tarea	Nombre tarea		Responsable	Estado
3	Gestionar excepciones en el sistema para los campos de entrada.		Abel Fernández Ferrer.	Completado
Descripción: Gestionar que las entradas solo sean números y no cadenas de caracteres.				

Tabla 20. Sprint 1. RF Gestionar secciones cónicas. Tarea 4

Sprint: 1	Inicio: 13/4/2015	Fin: 14/4/2015	Duración (días): 2	**Funcionalidad**: Gestionar secciones cónicas.
Número tarea	Nombre tarea		Responsable	Estado

4	Integrar con el componente.	Abel Fernández Ferrer.	Completado
Descripción: Se integra la vista creada con la interfaz principal del componente.			

Tabla 21. Sprint 1. RF Gestionar secciones cónicas. Tarea 5

Sprint: 1	Inicio: 15/4/2015	Fin: 16/4/2015	Duración (días): 2	**Funcionalidad**: Gestionar secciones cónicas.
Número tarea	Nombre tarea		Responsable	Estado
5	Ejecutar pruebas.		Abel Fernández Ferrer.	Completado
Descripción: Se ejecutan las pruebas a las funcionalidades implementadas durante la iteración.				

3.2.2 Sprint 2

Durante esta iteración se planificó la implementación de las funcionalidades para la administración de las secciones cilíndricas correspondiente a la RF 2. Durante esta iteración se dará por concluida una primera parte indispensable para la modelación del componente árboles escalonados, terminando con las estructuras básicas necesarias para su confección.

Tabla 22. Sprint 2. RF Gestionar secciones cilíndricas. Tarea 6

Sprint: 2	Inicio: 17/4/2015	Fin: 17/4/2015	Duración (días): 1	**Funcionalidad**: Gestionar secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
6	Crear interfaz de gestión para sección cilíndrica.		Abel Fernández Ferrer.	Completado
Descripción: Se brinda una interfaz con campos que permitan modificar la sección cilíndrica seleccionada en cuanto a su radio y longitud.				

Tabla 23. Sprint 2. RF Gestionar secciones cilíndricas. Tarea 7

Sprint: 2	Inicio: 20/4/2015	Fin: 22/4/2015	Duración (días): 3	**Funcionalidad**: Gestionar secciones cilíndricas.

Número tarea	Nombre tarea	Responsable	Estado
7	Crear métodos para la actualización de la vista al ejecutar las modificaciones.	Abel Fernández Ferrer.	Completado
Descripción: Se crea un método en la clase controladora que permita la actualización del visor principal al ejecutar las modificaciones en la sección seleccionada.			

Tabla 24. Sprint 2. RF Gestionar secciones cilíndricas. Tarea 8

Sprint: 2	Inicio: 23/4/2015	Fin:24/4/2015	Duración (días): 2	**Funcionalidad**: Gestionar secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
8	Integrar con el componente.		Abel Fernández Ferrer.	Completado
Descripción: Se integra con la interfaz principal del componente.				

Tabla 25. Sprint 2. RF Gestionar secciones cilíndricas. Tarea 9

Sprint: 2	Inicio: 27/4/2015	Fin: 27/4/2015	Duración (días): 1	**Funcionalidad**: Gestionar secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
9	Ejecutar de pruebas.		Abel Fernández Ferrer.	Completado
Descripción: Ejecutar pruebas a las funciones y la vista implementada.				

3.2.3 Sprint 3

Durante la tercera iteración se implementarán las funciones para la inserción de anillas en las secciones cilíndricas dándole el *release* a la funcionalidad para la inserción de anillas quedando solo 3 iteraciones para dar por concluido el producto.

Tabla 26. Sprint 3. RF Gestionar inserción de anillas en las secciones cilíndricas. Tarea 10

Sprint: 3	Inicio: 28/4/2015	Fin: 28/4/2015	Duración (días): 1	**Funcionalidad**: Gestionar inserción de anillas en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
10	Crear interfaz para la inserción de anillas en las secciones cilíndricas.		Abel Fernández Ferrer.	Completado
Descripción: Crear un panel que permita definir las características de las anillas a insertar en las secciones cilíndricas. Las mismas tendrán características de radio y radio interior, ancho y posición respecto al borde de la sección.				

Tabla 27. Sprint 3. RF Gestionar inserción de anillas en las secciones cilíndricas. Tarea 11

Sprint: 3	Inicio: 29/4/2015	Fin: 1/5/2015	Duración (días): 3	**Funcionalidad**: Gestionar inserción de anillas en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
11	Crear la entidad anilla en el modelo.		Abel Fernández Ferrer.	Completado
Descripción: Construir la clase anilla en el paquete *Modeling Data* para la posterior inserción en las secciones cilíndricas. En la misma se definen sus características en cuanto radio inferior y exterior, posición y ancho de la misma.				

Tabla 28. Sprint 3. RF Gestionar inserción de anillas en las secciones cilíndricas. Tarea 12

Sprint: 3	Inicio: 4/5/2015	Fin: 6/5/2015	Duración (días): 3	**Funcionalidad**: Gestionar inserción de anillas en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
12	Implementar métodos para la inserción de anillas en el visor principal de la aplicación.		Abel Fernández Ferrer.	Completado
Descripción: En el controlador del componente se implementarán los métodos necesarios para el posicionamiento de las anillas en la sección cilíndrica seleccionada.				

Tabla 29. Sprint 3. RF Gestionar inserción de anillas en las secciones cilíndricas. Tarea 13

Sprint: 3	Inicio: 7/5/2015	Fin: 7/5/2015	Duración (días): 1	**Funcionalidad**: Gestionar inserción de anillas en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
13	Ejecutar pruebas a la funcionalidad.		Abel Fernández Ferrer.	Completado
Descripción: Se le realizan pruebas a la funcionalidad mediante ejecuciones de la misma sobre las secciones cilíndricas mediante valores aleatorios para las anillas.				

3.2.4 Sprint 4

Durante esta iteración se implementarán las funciones para la inserción de agujeros diametrales en las secciones cilíndricas insertadas, garantizándose las opciones para la manipulación y posicionamiento de las mismas. Dicha funcionalidad debe permitir al usuario determinar tantos agujeros estime conveniente mostrándose un control de los mismos en la lista de modificaciones de la interfaz principal.

Tabla 30. Sprint 4. RF Gestionar agujeros diametrales en secciones cilíndricas. Tarea 14

Sprint: 4	Inicio: 8/5/2015	Fin: 8/5/2015	Duración (días): 1	**Funcionalidad**: Gestionar agujeros diametrales en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
14	Crear interfaz para la inserción de agujeros diametrales en las secciones cilíndricas.		Abel Fernández Ferrer.	Completado.
Descripción: Se genera una interfaz principal donde se muestren los campos para inserción de métricas como el radio, la posición y el ángulo del agujero diametral con respecto a la sección.				

Tabla 31. Sprint 4. RF Gestionar agujeros diametrales en secciones cilíndricas. Tarea 15

Sprint: 4	Inicio: 11/5/2015	Fin: 13/5/2015	Duración (días): 3	**Funcionalidad**: Gestionar agujeros diametrales en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
15	Crear entidad agujero diametral en el paquete de modelo.		Abel Fernández Ferrer.	Completado
Descripción: Dentro del paquete de modelado se creará una clase llamada ThroughHole la cual contiene las características de radio, posición y ángulo.				

Tabla 32. Sprint 4. RF Gestionar agujeros diametrales en secciones cilíndricas. Tarea 16

Sprint: 4	Inicio:14/5/2015	Fin: 16/5/2015	Duración (días): 3	**Funcionalidad**: Gestionar agujeros diametrales en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
16	Implementar métodos para la inserción de agujeros.		Abel Fernández Ferrer.	Completado
Descripción: En la clase controladora del componente se implementarán las funciones para la inserción de los agujeros en la sección cilíndrica seleccionada.				

Tabla 33. Sprint 4. RF Gestionar agujeros diametrales en secciones cilíndricas. Tarea 17

Sprint: 4	Inicio:18/5/2015	Fin:18/5/2015	Duración (días): 1	**Funcionalidad**: Gestionar agujeros diametrales en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
17	Ejecutar pruebas de funcionalidad.		Abel Fernández Ferrer.	Completado
Descripción: Probar mediante valores aleatorios que la funcionalidad esté libre de errores.				

Tabla 34. Sprint 4. RF Fusionar secciones. Tarea 18

Sprint: 4	Inicio: 19/5/2015	Fin: 19/5/2015	Duración (días): 1	**Funcionalidad**: Fusionar secciones.
Número tarea	Nombre tarea		Responsable	Estado
18	Implementar funcionalidad para fusionar secciones.		Abel Fernández Ferrer.	Completado
Descripción: En la clase controladora del componente se creara un método para el evento de fusión el cual permitirá fusionar todas las secciones y transformarlas en una sola.				

Tabla 35. Sprint 4. RF Fusionar secciones. Tarea 19

Sprint: 4	Inicio:20/5/2015	Fin: 20/5/2015	Duración (días): 1	**Funcionalidad**: Fusionar secciones.
Número tarea	Nombre tarea		Responsable	Estado
19	Ejecutar pruebas que validen la fusión de las secciones.		Abel Fernández Ferrer.	Completado
Descripción: Se debe desarrollar un modelo básico de árbol para comprobar que al terminar el diseño este fusiona todas las secciones de árbol y las transforma en una sola sección.				

Los datos del próximo sprint son expuestos en los anexos del documento.

3.3 Descripción de la solución

La solución es, en esencia, un componente para automatizar la modelación de árboles escalonados, en el que se pueden identificar dos componentes principales: la interfaz gráfica del componente y las funcionalidades con las que está conectada; la primera fue diseñada con el QtDesigner, mientras que las funcionalidades están encapsuladas en una clase controladora denominada **ShaftControl** con 23 métodos definidos, todo ello ubicado en el paquete Control.

En la figura se muestra el diseño de la interfaz principal:

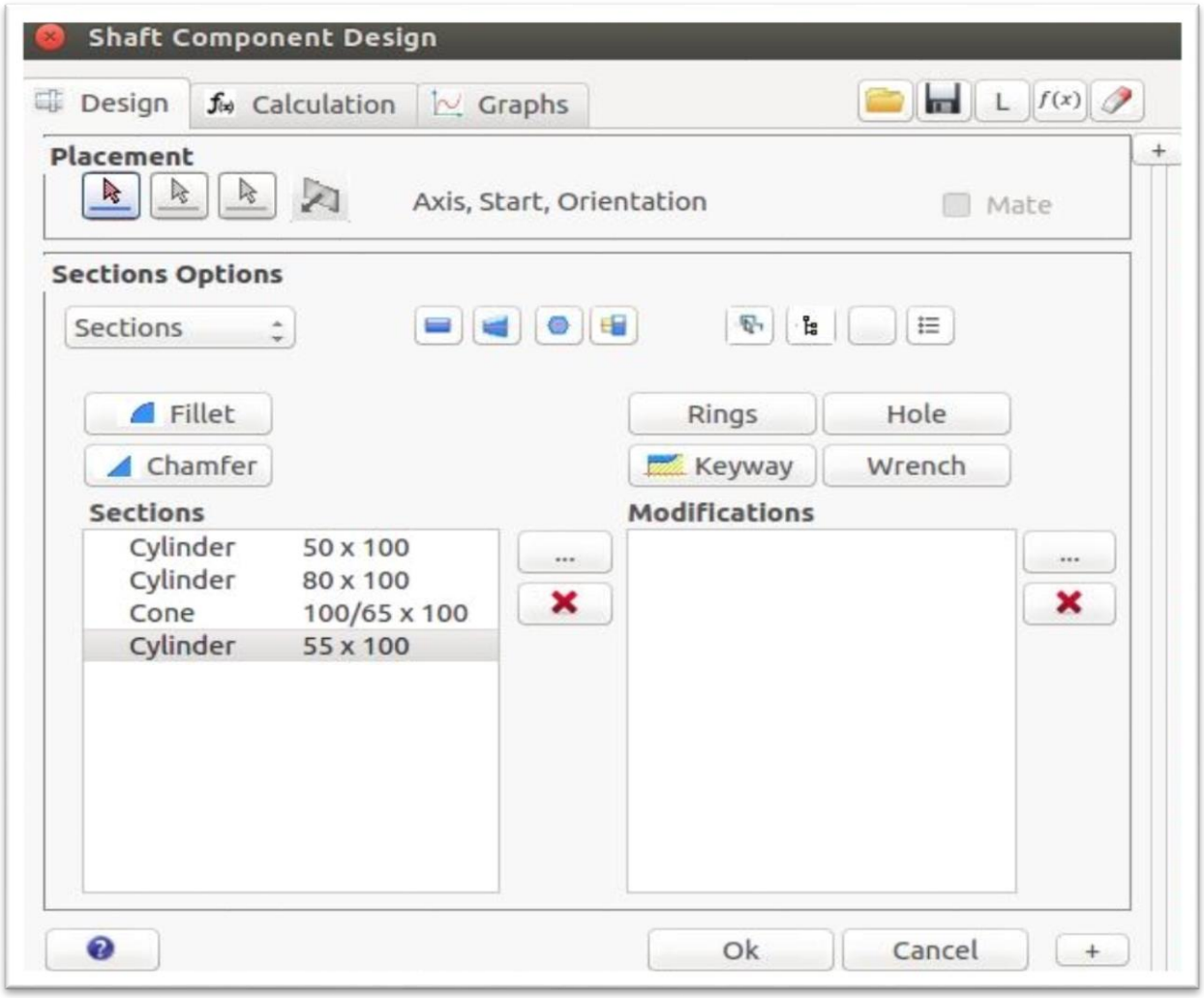


Figura 10. Interfaz principal del componente

En la misma se puede apreciar la pestaña ***design*** donde se muestran tres áreas de trabajo, solo se hablará de la segunda pues representa la de mayor importancia para el cliente.

Section Options:

- En esta sección se muestran los botones para la inserción de secciones cilíndricas, cónicas, poligonales y una opción para la división de cilindros en dos partes iguales.
- Más abajo a la izquierda se encuentra el área en la cual se almacenan las secciones insertadas y a la derecha las modificaciones aplicadas a cada una de las secciones insertadas.
- En las partes derechas de cada lista de secciones o modificaciones aparecen botones para gestionar las características de las entidades seleccionadas y uno inferior para eliminar las mismas.

Como parte del proceso de implementación fueron de gran importancia el uso de ciertas funcionalidades que propiciaron el correcto funcionamiento del componente así como de sus requisitos, las cuales serán referenciadas seguidamente.

Built_Cone (): Este método es el encargado de la modelación geométrica de los troncos de cono para luego visualizarlos como una sección del árbol.

Built_Cilinder (): En este se construye un objeto de tipo cilindro el cual pasa a ser parte de las secciones de árbol.

InsertKeyway (): Este método provee de las funciones necesarias para la creación de chaveteras en las secciones cilíndricas.

Insert_Retaining_Ring (): Este método inserta anillas en las secciones definidas por el usuario.

3.3.1 Librerías para la modelación:

Para la modelación de cada una de las secciones que conforman el árbol se utilizaron una serie de librerías que permitieron la posterior visualización en el visor principal de la aplicación así como la manipulación de los datos.

Tipos de datos: Son los estándares que se deben seguir, estos son retornados para la visualización y almacenamiento de datos propios de la aplicación.

- ***TopoDS_Shape.hxx***: Esta librería es la principal utilizada para los formatos de retornos siendo 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 que es equivalente al dominio de números reales.
- ***Standard_Integer.hxx*:** Formato numérico equivalente al dominio de enteros.
- ***BRepPrimAPI_MakeCone.hxx***: Librería que provee de las herramientas necesarias para la construcción de troncos de cono.
- ***BRepPrimAPI_MakeCylinder.hxx***: Posee las características para la modelación de secciones cilíndricas.

3.3.2 Librerías para la visualización de entidades en el área de trabajo

Para la definición y visualización de cada una de las secciones se utilizaron diferentes librerías del paquete **OpenCascade** que permitieron el posicionamiento de estas estructuras en el espacio definido por el usuario garantizando la correcta modelación y visualización de la pieza. A continuación se exponen las mismas.

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

3.3.3 Funciones para el corte, fusión y fresado de secciones

Durante el proceso de modelado el usuario tiene la posibilidad de agregar tantas secciones estime convenientes para el árbol escalonado que este diseñando y aplicar las operaciones de fresado (cortes y agujeros a la pieza) una vez terminado este. Para la aplicación de estos elementos se generaron diferentes entidades que sirven para crear este proceso de fresado como son las anillas, agujeros diametrales, chavetas, entre otras que aún se encuentran en desarrollo. Cada uno de estos elementos requirió de librerías y funciones ya disponibles en OpenCascade, estas serán mencionadas a continuación.

- ***BRepAlgoAPI_Cut.hxx***: Contiene los métodos necesarios para realizar la operación de diferencia booleana entre dos entidades geométricas en el espacio. De gran utilidad ya que permitió la ejecución de fresado en las secciones del árbol escalonado.
- ***BRepAlgoAPI_Fuse.hxx***: Posee los métodos de suma booleana entre dos entidades en el espacio. Se utilizó para la fusión de las secciones del árbol escalonado.
- ***BRepAlgoAPI_Common.hxx:*** Brinda las funciones de espacios comunes entre dos entidades geométricas y retorna las áreas comunes entre las dos.

3.4 Etapa de pruebas

Uno de los pilares fundamentales para garantizar la calidad del sistema es la ejecución de pruebas durante el proceso de producción de software. Unas de las prácticas más aplicadas son las pruebas de caja negra y caja blanca, comprobándose el comportamiento de entrada-respuesta en el caso de la primera y la inmersión directamente en el código en el caso de las pruebas de caja blanca. Para el desarrollo de la aplicación se tomó como paradigma de la metodología XP el desarrollo basado en pruebas o *Test Driven Development* (TDD, por sus siglas en inglés), aplicándose las pruebas de aceptación, que consisten en casos de prueba que corroboran la compatibilidad entre lo que se desarrolló y los deseos reales del cliente (30).

3.4.1 Pruebas de aceptación (*Assert Test Driven Development*)

Los test de aceptación o de cliente son el criterio escrito de que el software cumple los requisitos de negocio que el cliente demanda. Son ejemplos escritos por los dueños de producto. Es el punto de partida del desarrollo en cada iteración, la conexión perfecta entre Scrum y XP; allá

donde una se queda y sigue la otra (30). Estas pruebas se realizan al final, antes del despliegue del sistema y generalmente lo realizan los usuarios finales.

A continuación se exponen las pruebas de aceptación realizadas a los RF definidos en la documentación:

Tabla 36. CP Insertar sección cónica.

Caso de prueba de aceptación	
Historia de usuario: Insertar sección cónica.	Id. de prueba:1
Nombre del caso de prueba: CP insertar sección cónica.	
Descripción del caso de prueba: Permite insertar secciones cónicas en la pieza principal.	
Condiciones de ejecución: Debe tener la interfaz de Shaft en la pestaña de diseño.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Presionar el botón de sección cónica.	
Resultado esperado: Se inserta la sección cónica en la pieza principal mostrándose en el visor.	
Evaluación de la prueba: Satisfactoria.	

Tabla 37. CP Modificar sección cónica.

Caso de prueba de aceptación	
Historia de usuario: Modificar sección cónica.	Id. de prueba:2
Nombre del caso de prueba: CP modificar sección cónica.	
Descripción del caso de prueba: Permite gestionar las dimensiones en cuanto radio y longitud de una sección cónica seleccionada por el usuario.	
Condiciones de ejecución: La sección debe estar insertada.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección cónica que se desea modificar. ▪ Presionar el botón de modificación o aplicar doble clic sobre la sección. ▪ Modificar las dimensiones deseadas. ▪ Aplicar los cambios.	
Resultado esperado: Se modifican las dimensiones de la sección cónica.	
Evaluación de la prueba: Satisfactoria.	

Tabla 38. CP Eliminar sección cónica.

Caso de prueba de aceptación	
Historia de usuario: Eliminar sección cónica.	Id. de prueba:3
Nombre del caso de prueba: CP eliminar sección cónica.	
Descripción del caso de prueba: Se elimina la sección cónica seleccionada.	
Condiciones de ejecución: La sección debe estar insertada.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección a eliminar. ▪ Presionar el botón eliminar.	
Resultado esperado: Se elimina la sección seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Los restantes casos de pruebas están expuestos en los anexos de la documentación.

3.4.2 Pruebas unitarias

La realización de pruebas unitarias antes de comenzar la codificación es fundamental en la metodología de desarrollo XP. Estas pruebas deben tener una estructura que permita ejecutarlas repetidas veces y con facilidad. Los requisitos funcionales en XP se expresan como historias de usuario, donde el equipo de desarrollo evalúa cada historia y las divide en tareas. Cada tarea representa una característica diferente del sistema y se puede diseñar entonces una prueba de unidad para verificar la implementación descrita en esa tarea.

Para realizar las pruebas unitarias del componente implementado se utilizó el conjunto de funcionalidades brindado por el *framework* de Qt llamado “Qtestlib”.

En la siguiente figura se muestra un ejemplo de las pruebas unitarias realizadas durante el proceso de implementación utilizando la clase **Qtest** del framework **Qt**.

```
********* Start testing of QtestTest *********
Config: Using QTest library 4.8.6, Qt 4.8.6
PASS   : QtestTest::initTestCase()
PASS   : QtestTest::testConeClass()
PASS   : QtestTest::testCylinderClass()
PASS   : QtestTest::testConeSets()
PASS   : QtestTest::testCylinderSets()
PASS   : QtestTest::testKeywayClass()
PASS   : QtestTest::testThroughHoleClass()
PASS   : QtestTest::testRingsClass()
PASS   : QtestTest::testConeBuilt()
PASS   : QtestTest::testCylinderBuilt()
PASS   : QtestTest::testKeywayBuilt()
PASS   : QtestTest::testRingsBuilt()
PASS   : QtestTest::cleanupTestCase()
Totals: 13 passed, 0 failed, 0 skipped
********* Finished testing of QtestTest *********
```

Figura 11. Ejemplo de pruebas unitarias realizadas

3.5 Resultados de las pruebas efectuadas

El siguiente gráfico muestra el resumen del resultado de las pruebas realizadas.

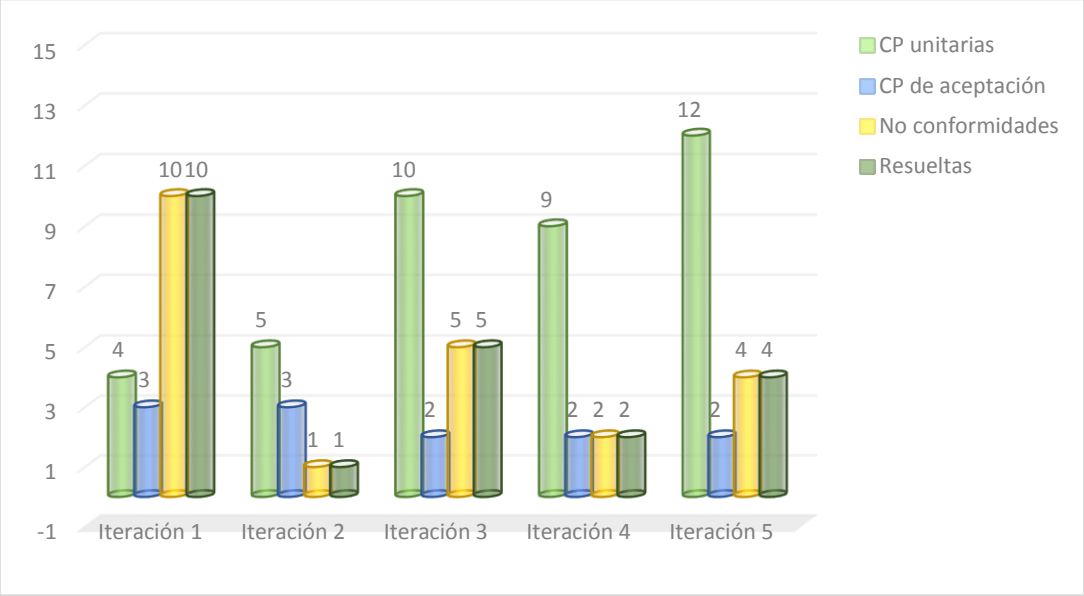


Figura 12. Resultados pruebas unitarias y de aceptación del proceso de desarrollo.

Durante el proceso de desarrollo del componente se realizaron una serie de pruebas unitarias y de aceptación que sostuvieron el proceso de desarrollo y guiaron la implementación de las funcionalidades. Durante el primer sprint se realizaron 4 caso de pruebas unitarias y 3 de aceptación dando como resultados 10 no conformidades, estas fueron resueltas para la salida de la primera versión; para la segunda iteración fueron efectuadas 5 casos de pruebas unitarias y 2 de aceptación, arrojando 1 no conformidad que fue resuelta posteriormente. Durante el tercer sprint se programaron 10 pruebas unitarias y 2 de aceptación dando como resultado 5 deficiencias siendo

resultas una vez encontradas. En el transcurso de la 4 iteración se realizaron 9 pruebas unitarias enfocadas principalmente al análisis del código para la construcción del agujero diametral, además de sus respectivas pruebas de aceptación, dando como resultados 2 no conformidades las que fueron resueltas. En la última entrega y quinta iteración fueron conformadas 12 pruebas unitarias al código y 2 de aceptación dando como resultados 4 no conformidades que fueron resueltas. Al resolver todas las no conformidades encontradas en cada iteración permitió hacer entrega de un producto libre de errores para su posterior liberación.

3.6 Conclusiones parciales

En este capítulo fueron abordados los temas referentes a la implementación, validación y prueba de la solución, quedando plasmado el estándar de código utilizado, las tareas llevadas a cabo durante las iteraciones realizadas y las pruebas de caja blanca y caja negra efectuadas, siguiendo el desarrollo guiado por pruebas propuesto por la metodología XP, concluyendo el proceso con resultados satisfactorios dándole solución a las no conformidades identificadas.

CONCLUSIONES GENERALES

Durante el transcurso de la investigación y el proceso de desarrollo han sido logrados los objetivos propuestos arribando a las siguientes conclusiones:

- El desarrollo del componente representa una herramienta que de ser llevada a la práctica simplificará el diseño de estas estructuras por parte de los diseñadores mecánicos.
- Representa hasta el momento una solución única de su tipo en el país y en el campo del software libre y de código abierto.
- El componente generado muestra una arquitectura y estructura permitiendo ser integrable con otras plataformas que implementen el lenguaje OpenCascade.

RECOMENDACIONES

A partir del trabajo realizado y luego de haber analizado los resultados obtenidos se sugieren las siguientes recomendaciones:

- Implementación de los módulos de cálculo y graficado del componente.
- Inclusión de una biblioteca con estándares para la obtención de resultados más precisos en los diseños.
- Publicación de los resultados en las comunidades de software libre como parte del proceso de liberación del código de la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. PLM Siemens. CAD/Diseño asistido por Computadora. [En línea] PLM SIEMENS, 2015. http://www.plm.automation.siemens.com/es_mx/plm/cad.shtml.

2. Rodriguez Hernández, Orlando y Corugedo Mendez, Angel. *Dibujo Aplicado para Ingenieros.* La Habana : s.n., 2005. Vol. II. 959-258-905-4 tomo 2.

3. Systemes, Dassault. CATIA-Dar forma al mundo que nos rodea . [En línea] Dassault Systemes, 2015. http://www.3ds.com/es/productos-y-servicios/catia/.

4. Dassault Systemes. Home page- Dassault Systemes. [En línea] Dassault Systemes, 2015. http://www.3ds.com/es.

5. Systemes, Dassault. SolidWorks. [En línea] Dassault Systemes, 2015. http://www.solidworks.es/.

6. Corp., Autodesk. Autodesk.es. [En línea] Autodesk, 2015. http://www.autodesk.es/products/autocad.

7. —. Autodesk.es. [En línea] Autodesk, 2015. http://www.autodesk.es/products/inventor/.

8. Freecad. FreeCAD Documentation. [En línea] Freecad, 6 de 12 de 2014. http://www.freecadweb.org.

9. LibreCAD. LibreCAD Home Page. [En línea] LibreCAD, 2015. http://librecad.org/cms/home.html.

10. Universidad de Murcia. Universidad de Murcia. [En línea] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

11. ProcesosdeSoftware. [En línea] 2015. http://procesosdesoftware.wikispaces.com/METODOLOGIA+SCRUM.

12. ProcesosdeSoftware. [En línea] 2015. http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

13. Kniberg, Henrik. *SCRUM y XP desde las Trincheras.* s.l. : InfoQ, 2007. 978-1-4303-2264-1.

14. Beck, Kent. *Extreme Programing.* Massachusetts : s.n., 2000. 0201616416.

15. Cplusplus.com. [En línea] 2015. http://www.cplusplus.com/info/.

16. ALEGSA.com.ar. [En línea] 2015. http://www.alegsa.com.ar/Dic/framework.php.

17. Corp, Spatial. Spatial | 3D Software Components for 3D Modeling, 3D Data Exchange, and 3D Visualization. [En línea] Spatial Corp., 2015. http://www.spatial.com/.

18. VectorWORKS. VectorWORKSCR.com. [En línea] VectorWorks, 2009. http://www.vectorworksca.com/vectorworks/parasolid.html.

19. OpenCascade. OpenCascade Technology. [En línea] OPEN CASCADE S.A.S, 2015. http://www.opencascade.org/occt/.

20. Doxygen. Doxygen. [En línea] http://www.stack.nl/~dimitri/doxygen/.

21. Vigo : Union de Webmaster Libre. [En línea] 2015. https://sites.google.com/site/vigomiciudad/otras-cosa-intersantes/ide-entornos-de-desarrollo-integrado.

22. Qt Company. Cross-platform application. [En línea] 2015. http://doc.qt.io/qtcreator/.

23. Departamento de Sistemas Informáticos y Computación. Metodología y Tecnología de la Programación. [En línea] http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

24. Software Design Tools for Agile Teams, with UML, BPMN and More. [En línea] Visual Paradigm, 2015. http://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

25. Larman, Craig. *UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.* Prentice Hall : s.n., 1999.

26. Reinoso y N. Kiccillof. *Estilos y Patrones en la Estrategia de Arquitectura Microsoft.* Buenos Aires-Argentina : s.n., 2004.

27. Grosso, Andres. Prácticas de Software. [En línea] 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

28. Ingenieria del Software. *Fase de diseño 2.*

29. L, Craig. *UML y patrones.* s.l. : Prentice Hall., 2004.

30. Beas, Jose Manuel y Ble Jurado, Carlos . Desarrollo Dirigido por Tests de Aceptación (ATDD). *Diseño Agil con TDD.* 2010.

31. Iglesias, Arabel Moráguez. monografias.com. [En línea] octubre de 2005. http://monografias.com.

32. Medina, Dr.Augusto Cesar. *Herramientas utilizadas por diseñadores.* La Habana, febrero de 2015.

33. Systemes, Dassault. Dassault Systemes Corporation. [En línea] Dassault Systemes, 2015. http://www.3ds.com/products-services/catia/.

34. *Arquitectura de Software: Estilos y patrones.* Sandra Almeira, Adriana y Perez Cavenago, Vanina. 2007.

35. SOLID WORKS-INSPIRACIÓN PARA LA INNOVACIÓN . [En línea] Dassault Systèmes, 2015. http://www.3ds.com/es/productos-y-servicios/solidworks/.

36. PLM-Siemens. CAD-Diseño Asistido por Computadora. [En línea] PLM-SIEMENS, 2015. http://www.plm.automation.siemens.com/es_sa/plm/cad.shtml.

37. Oracle Corporation. NetBeans. [En línea] Oracle Corporation, 2015. https://netbeans.org/index_es.html.

38. Hosting, Qt Project. Qt Project. [En línea] Digia, 2014. https://qt-project.org/wiki/Category:Tools::QtCreator_Spanish.

39. IBM. IBM Developer Works. [En línea] IBM, 26 de 11 de 2012. http://www.ibm.com/developerworks/ssa/library/os-ecov/.

40. *Patrones de diseño.* Theoktisto, Víctor. Venezuela : s.n., 2000.

41. Gamma, E. *Design Patterns: Elements of Reusable Object-Oriented Software. .* s.l. : Addison-Wesley Professional. 416, 1994.

42. Larman, Craig. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educación, 2003. ISBN 9788420534381.

43. Reynoso, Carlos y Kiccillof, Nicolás. *Estilos y Patrones en la Estrategia de Microsoft.* Buenos Aires : UNIVERSIDAD DE BUENOS AIRES, 2004.

GLOSARIO DE TÉRMINOS

BMP: *Bit Mapped Image* (Imagen de mapa bit). Es un formato de mapa bit nativo de Microsoft Windows.

DWG: Es un formato de archivo informático de dibujo computarizado derivado de la palabra inglesa "*drawing*"

DXF: *Drawing Exchange File. Es un* archivo de intercambio para la importación-exportación entre sistemas CAD.

GIF: *Graphics Interchange Format*.

IGES: *Initial Graphics Exchange Specification* (*Especificación de Intercambio Inicial de Gráficos*). Es un sistema de archivo informático que define un formato neutral de datos que permite el intercambio digital de información entre sistemas de diseño asistido por computadora (CAD).

JPEG: *Joint Photographic Experts Group*, Grupo Conjunto de Expertos en Fotografía, es el nombre de un comité de expertos que creó un estándar de compresión y codificación de archivos e imágenes fijas.

PNG: *Portable Network Graphics* (PNG) por sus siglas del inglés es un formato gráfico basado en un algoritmo de compresión sin pérdida para *bitmaps* no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

SAT: Es un formato de archivo informático de tipo CAD para ACIS.

STEP: *Standard for the Exchange of Product model data (STEP) es una ISO* 10303 denominada así por ser un estándar para el intercambio de datos entre sistemas CAD, CAM, CAE, PDM/EDM y otros sistemas CAx. STEP soporta modelos de diseño mecánico, eléctrico, análisis y manufactura, con información adicional específica de varias industrias tales como automotriz, aeroespacial, construcción de edificios, barcos, aceite y combustibles, plantas de proceso y otros.

SVG: Gráficos Vectoriales Redimensionables (del inglés *Scalable Vector Graphics*) o SVG son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en formato XML.

TIFF: (*Tagged Image File Format*) es un formato de archivo informático para imágenes.

UML: Lenguaje Unificado de Modelado, es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

ANEXOS

Anexo 1: Iteración 5 realizada durante el proceso de desarrollo del componente.

Tabla 39. Sprint 5. RF Gestionar chaveteras en las secciones cilíndricas. Tarea 20

Sprint: 5	Inicio: 21/5/2015	Fin: 22/5/2015	Duración (días): 2	**Funcionalidad**: Gestionar chaveteras en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
20	Diseñar interfaz para inserción de chaveteras.		Abel Fernández Ferrer.	Completado
Descripción: Se creara una interfaz que permita la inserción desde 1 hasta 4 chaveteras en cualquier sección cilíndrica del diseño definiendo en la misma la longitud y el ancho de la chavetera.				

Tabla 40. Sprint 5. RF Gestionar chaveteras en las secciones cilíndricas. Tarea 21

Sprint: 5	Inicio: 23/5/2015	Fin: 26/5/2015	Duración (días): 4	**Funcionalidad**: Gestionar chaveteras en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado
21	Implementar la clase chavetera en el paquete de modelo que responda a la cantidad de chavetas orientada por el usuario.		Abel Fernández Ferrer.	Completado
Descripción: Se creara una interfaz que permita la inserción desde 1 hasta 4 chaveteras en cualquier sección cilíndrica del diseño definiendo en la misma la longitud y el ancho de la chavetera.				

Tabla 41. Sprint 5. RF Gestionar chaveteras en las secciones cilíndricas. Tarea 22

Sprint: 5	Inicio: 27/5/2015	Fin: 28/5/2015	Duración (días): 2	**Funcionalidad**: Gestionar chaveteras en las secciones cilíndricas.
Número tarea	Nombre tarea		Responsable	Estado

22	Ejecutar pruebas a la inserción de chaveteras.	Abel Fernández Ferrer.	Completado
Descripción: Mediante la interfaz de inserción de chaveteras se probaran valores aleatorios para validar la calidad de la funcionalidad implementada.			

Anexo 2: Casos de pruebas de aceptación.

Tabla 42. CP Modificar sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Modificar sección cilíndrica.	Id. de prueba:5
Nombre del caso de prueba: CP modificar sección cilíndrica.	
Descripción del caso de prueba: Permite modificar las dimensiones en cuanto radio y longitud de una sección cilíndrica seleccionada por el usuario.	
Condiciones de ejecución: La sección debe estar insertada.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección cilíndrica que se desea modificar. ▪ Presionar el botón de modificación o aplicar doble clic sobre la sección. ▪ Modificar las dimensiones deseadas. ▪ Aplicar los cambios.	
Resultado esperado: Se modifican las dimensiones de la sección cilíndrica.	
Evaluación de la prueba: Satisfactoria.	

Tabla 43. CP Eliminar sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Eliminar sección cilíndrica.	Id. de prueba:6
Nombre del caso de prueba: CP eliminar sección cilíndrica.	
Descripción del caso de prueba: Permite eliminar secciones cilíndricas de la pieza principal.	
Condiciones de ejecución: La sección debe estar insertada.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño.	

▪ Seleccionar la sección cilíndrica que se quiere eliminar. ▪ Presionar el botón eliminar.
Resultado esperado: Se elimina la sección cilíndrica seleccionada eliminándola de la pieza principal.
Evaluación de la prueba: Satisfactoria.

Tabla 44. CP insertar anilla en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Insertar anilla en sección cilíndrica.	Id. de prueba:7
Nombre del caso de prueba: CP insertar anilla en sección cilíndrica.	
Descripción del caso de prueba: Permite insertar anillas en las secciones cilíndricas.	
Condiciones de ejecución: Debe existir al menos una sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección cilíndrica a la cual se le desea insertar la anilla. ▪ Presionar el botón insertar anilla. ▪ Seleccionar las características de la anilla deseada. ▪ Aplicar los cambios.	
Resultado esperado: Inserta una anilla con las características insertadas en la sección cilíndrica seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla 45. CP eliminar anilla en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Eliminar anilla en sección cilíndrica.	Id. de prueba:8
Nombre del caso de prueba: CP eliminar anilla en sección cilíndrica.	
Descripción del caso de prueba: Permite eliminar anillas en las secciones cilíndricas de la pieza principal.	
Condiciones de ejecución: Debe existir la anilla en la sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la anilla que se quiere eliminar. ▪ Presionar el botón eliminar.	

Resultado esperado: Se elimina la anilla de la sección cilíndrica.
Evaluación de la prueba: Satisfactoria.

Tabla 46. CP insertar agujero diametral en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Insertar agujero diametral en sección cilíndrica.	Id. de prueba: 9
Nombre del caso de prueba: CP insertar agujero diametral en sección cilíndrica.	
Descripción del caso de prueba: Permite insertar agujeros diametrales en las secciones cilíndricas.	
Condiciones de ejecución: Debe existir al menos una sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección cilíndrica a la cual se le desea insertar el agujero diametral. ▪ Presionar el botón agujero diametral. ▪ Seleccionar las características del agujero diametral deseado. ▪ Aplicar los cambios.	
Resultado esperado: Inserta un agujero diametral con las características insertadas en la sección cilíndrica seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla 47. CP eliminar agujero diametral en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Eliminar agujero diametral en sección cilíndrica.	Id. de prueba:10
Nombre del caso de prueba: CP eliminar agujero diametral en sección cilíndrica.	
Descripción del caso de prueba: Permite eliminar los agujeros diametrales de las secciones cilíndricas.	
Condiciones de ejecución: Debe existir el agujero diametral en la sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar el agujero diametral que se quiere eliminar. ▪ Presionar el botón eliminar.	
Resultado esperado: Se elimina el agujero diametral de la sección cilíndrica.	

Evaluación de la prueba: Satisfactoria.

Tabla 48. CP insertar chaveteras en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Insertar chaveteras en sección cilíndrica.	Id. de prueba:11
Nombre del caso de prueba: CP insertar chaveteras en sección cilíndrica.	
Descripción del caso de prueba: Permite insertar chaveteras en las secciones cilíndricas.	
Condiciones de ejecución: Debe existir al menos una sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la sección cilíndrica en la cual se desea insertar la(s) chavetera(s). ▪ Presionar el botón de insertar chavetera. ▪ En la caja de dialogo seleccionar las características deseadas por el usuario. ▪ Aplicar los cambios.	
Resultado esperado: Se inserta la cantidad de chavetas seleccionadas en la sección cilíndrica.	
Evaluación de la prueba: Satisfactoria.	

Tabla 49. CP eliminar chavetera(s) en sección cilíndrica.

Caso de prueba de aceptación	
Historia de usuario: Eliminar chavetera(s) en sección cilíndrica.	Id. de prueba:12
Nombre del caso de prueba: CP eliminar chavetera(s) en sección cilíndrica.	
Descripción del caso de prueba: Permite eliminar la(s) chavetera(s) de las secciones cilíndricas.	
Condiciones de ejecución: Debe(n) existir chavetera(s) en la sección cilíndrica.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Seleccionar la(s) chavetera(s) que se quiere(n) eliminar. ▪ Presionar el botón eliminar.	
Resultado esperado: Se elimina(n) la(s) chavetera(s) de la sección cilíndrica.	
Evaluación de la prueba: Satisfactoria.	

Tabla 50. CP Fusionar secciones

Caso de prueba de aceptación	
Historia de usuario: Fusionar secciones.	Id. de prueba:13
Nombre del caso de prueba: CP fusionar secciones.	
Descripción del caso de prueba: Permite fusionar todas las secciones de la pieza y tratarla como una sola.	
Condiciones de ejecución: Debe existir al menos una sección.	
Entrada / Pasos de ejecución: ▪ Situarse en la pestaña de diseño. ▪ Realizar el diseño de un árbol. ▪ Presionar el botón ok.	
Resultado esperado: Se cierra la interfaz principal y se muestra el árbol como una sola sección.	
Evaluación de la prueba: Satisfactoria.	